UNIVERSIDADE D
COIMBRA

Rafael Alexandre Rodrigues Neves

SALA-Z

SISTEMA PARA PROCURA E GESTÃO DE SALAS DE EVENTOS MUSICAIS

VOLUME 1

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Engenharia de Software orientada pelo Professor Fernando José Barros Rodrigues da Silva e pelo Engenheiro Arnaldo Moura e apresentada Faculdade de Ciências e Tecnologia / Departamento de Engenharia Informática.

Junho de 2021

Esta página foi intencionalmente deixada em branco.

Agradecimentos

Quero começar por agradecer o apoio incondicional da minha família e, em particular, dos meus pais que sempre foram o meu maior suporte durante o meu percurso académico.

De seguida, quero agradecer a toda a empresa Grama pela incrível experiência que me proporcionaram e pelo modo acolhedor com que me aceitaram. Quero agradecer especialmente ao meu orientador de estágio Arnaldo Moura e à Susana Lourenço por serem os principais responsáveis pelo meus sucesso no decorrer do estágio curricular. Em adição, quero agradecer a toda a equipa que esteve presente ou contactou com o projeto do Sala-Z e que me ajudou sempre que precisei.

Quero também agradecer ao orientador Fernando Barros por ser um apoio constante durante o estágio curricular e sempre ajudar em todos os aspectos curriculares.

Por fim, quero agradecer a todos os meus amigos que sempre me apoiaram durante o meu percurso académico.

Esta página foi intencionalmente deixada em branco.

Resumo

Em virtude de haver uma proximidade entre o mundo musical e alguns membros da empresa Grama, soube-se que existe a necessidade por parte dos artistas musicais e dos gerentes de estabelecimentos que conduzem eventos musicais de haver uma ferramenta que, de alguma forma, ajude o processo de angariação, condução e monitorização destes eventos e do negócio pessoal em si.

Após a empresa de software, Grama, lançar a primeira versão da plataforma Sala-Z em 2019 com o objetivo de facilitar a procura de locais de espetáculos por parte de artistas musicais, em 2020, num contexto de estágio curricular, a Grama decidiu recriar e evoluir esta plataforma para, não só listar locais de espetáculos mas, também, possibilitar que os gerentes destes tenham uma ferramenta de gestão do seu estabelecimento e de eventos associados.

No presente relatório de estágio, será exposto todo o processo que o aluno e a empresa tiveram durante o decorrer do ano curricular 2020/2021. Serão analisadas as políticas de trabalho e de organização de eventos dentro dos locais de espetáculos com o auxílio de possíveis clientes que, durante o estágio, ajudaram a equipa a criar um produto que se adequa às suas necessidades.

Assim, após serem estudadas ferramentas que, de alguma forma, concorrem ao Sala-Z, foi desenhada e criada uma ferramenta moderna, inovadora e interessante que visa auxiliar os diversos estabelecimentos a gerir os seus espaços e eventos musicais. Seguindo diversas técnicas e princípios de engenharia de software, o aluno conseguiu terminar o estágio curricular com uma plataforma inovadora que poderá ser utilizada por diversos gerentes de eventos musicais num futuro próximo.

Palavras-Chave

Gestão de eventos musicais, *Venues*, *Web Application*, *TypeScript*, *Java*, Micro-serviços, Engenharia de *Software.***-

Esta página foi intencionalmente deixada em branco.

Abstract

Due to the proximity between the musical world and some members of Grama, it was learned that there was a need on part of musical artists and venue managers who conduct musical events to have a tool that, in some way, helps the process of raising, conducting and monitoring musical events and the personal business itself.

As such, after the software company, Grama, launched the first version of Sala-Z in 2019 with the objective of facilitating the search for venues by musical artists, in 2020, in context of a curricular internship, Grama decided to recreate and evolve this platform in order to, not only list venues, but also providing their managers with a tool to manage their establishment and associated events combating the need they currently have.

In the current internship report, the entire process that the student and the company had during the course of the curricular year 2020/2021 will be exposed. The work and event organization policies within the venues will be analyzed with the help of potential customers who, during the internship, helped the team to create a product that suits their needs.

Thus, after studying tools that, somehow, compete with Sala-Z, a modern, innovative and interesting tool was designed and created to help the various establishments manage their spaces and events. Following several software engineering techniques and principles, the student managed to finish the internship with an innovative plataform that can be used by venue managers in a near future.

Keywords

Musical events management, Venues, Web Application, TypeScript, Java, Microservices, Software Engineering.

Esta página foi intencionalmente deixada em branco.

Conteúdo

Esta página foi intencionalmente deixada em branco.

Acronyms

AWS *Amazon Web Services*. 17, 21, 35, 36, 38–40

EC2 *Amazon Elastic Compute Cloud*. 35, 36, 38

ELB *Elastic Load Balancing*. 36, 38

ER Entidade-Relacionamento. 42

MVP *Minimum Viable Product*. 28

RDS *Amazon Relational Database Service*. 37

RGPD Regulamento Geral sobre a Proteção de Dados. 35

S3 *Amazon Simple Storage Service*. 36

SES *Amazon Simple Email Service*. 37

SO Sistema Operativo. 23

TI Tecnologias de informação. 1

Esta página foi intencionalmente deixada em branco.

Lista de Figuras

Esta página foi intencionalmente deixada em branco.

Lista de Tabelas

Esta página foi intencionalmente deixada em branco.

Capítulo 1

Introdução

O presente relatório foi desenvolvido no âmbito da disciplina de estágio curricular do Mestrado de Engenharia Informática com especialização em Engenharia de *Software* da Faculdade de Ciências e Tecnologia da Universidade de Coimbra no ano lectivo 2020/2021.

Todo o estágio foi acompanhado pelo professor Fernando Barros do departamento de Engenharia Informática da Universidade de Coimbra, assim como, pelo engenheiro de *software* Arnaldo Moura, da empresa responsável pelo presente estágio.

1.1 Contexto

O estágio curricular teve uma duração de dois semestres e foi realizado na empresa de desenvolvimento de *software*, Grama. Fundada em Janeiro de 2017, a empresa está localizada atualmente em Coimbra, Portugal e o núcleo da empresa tem vários anos de experiência em desenvolvimento de projetos complexos de Tecnologias de informação (TI), nomeadamente, para algumas das maiores operadoras de telecomunicações da Europa e América do Sul.

Foi proposto pela empresa que o aluno desenvolvesse uma segunda versão da plataforma de procura e gestão de salas de eventos denominada, Sala-Z.

O Sala-Z, na sua versão 1.0, é uma plataforma que permite encontrar locais de espectáculo (mais usualmente tratados como *venues)*, classificados por localização, detalhes técnicos, tipo e capacidade, colmatando assim a dificuldade que os artistas e promotores de eventos sentem para encontrar e contactar estas salas de espetáculos para os seus eventos. Permite, também, que os donos das *venues* submetam a informação necessária a ser incluída nos resultados de pesquisa que, posteriormente, será recebida pelos administradores através do *backoffice*, onde conseguirão rever e aprovar o conteúdo, bem como adicionar, editar e remover dados existentes.

1.2 Objetivos

O Sala-Z é um projeto interno e antigo da empresa (Secção 3.2), porém, nunca foi totalmente finalizado. Assim, a proposta realizada pela empresa passa por recriar o antigo *software* no seu todo, de modo a que possa servir como ferramenta de gestão das próprias *venues* e dos eventos que estão associados às mesmas.

Assim, o objetivo será desenvolver a segunda versão da aplicação *web* de forma a que disponibilize aos gerentes das *venues* uma forma fácil e conveniente de gerir a(s) sua(s) sala(s) de espetáculo(s) de forma a conseguirem organizar cada evento e recursos associados, assim como, conseguirem ter uma tradução

visual de estatísticas monetárias da(s) *venue(s)* pelas quais estão responsáveis.

1.3 Estrutura

O presente documento está estruturado em 8 capítulos e o objetivo de cada um é apresentar, detalhadamente, cada uma das etapas de trabalho vividas pelo aluno:

- No **capítulo 1** é dada uma introdução a todo o relatório, apresentando o contexto e a estrutura do documento.
- O objetivo do **capítulo 2** é analisar e apresentar a metodologia de trabalho adoptada na empresa Grama e, por consequência, pelo estagiário e apresentar o planeamento temporal do trabalho realizado ao longo do ano curricular, assim como, o plano de testes usado dentro da empresa.
- O **capítulo 3** é dedicado ao estado de arte. Neste, são analisados temas que estão ligado ao âmbito do trabalho, começando por uma explicação do que são as *venues*, que tipos existem e como funcionam. De seguida, é abordada a primeira versão do Sala-Z, estudando as funcionalidades que esta fornece, assim como, que funcionalidades é que poderão ser usadas para auxiliar o desenvolvimento da sua nova versão.

  Em adição, é realizado um estudo sobre os concorrentes existentes com o objetivo de comparar as plataformas já existentes e o Sala-Z conseguindo, por sua vez, entender de que modo o mesmo se destaca dos produtos já disponíveis no mercado.

  Visto que foi dado ao estagiário a hipótese de escolher entre várias ferramentas de desenvolvimento, será também apresentado o estudo realizado sobre as tecnologias, justificando as escolhas tomadas.

  Por fim será feita uma breve análise dos riscos existentes no desenvolvimento do produto, assim como, riscos que poderão afetar o sucesso da plataforma.
- No **capítulo 4** irão ser expostas todas as *user stories* escritas no decorrer do primeiro semestre e, consequentemente, todos os requisitos funcionais e não funcionais da segunda versão do Sala-Z.
- No **capítulo 5** irá ser exposta a arquitetura do *software* segundo o modelo C4.
- Por sua vez, no **capítulo 6**, irá ser apresentada toda a fase de desenvolvimento e requisitos desenvolvidos, assim como, serão expostos algumas das funcionalidades mais importantes da aplicação desenvolvida no 2º semestre.
- Por consequência, o **capítulo 7** tem como objetivo apresentar a fase de testes e todos os testes realizados após o terminar do desenvolvimento da aplicação
- Por fim, será feita uma breve conclusão do relatório no **capítulo 8**.

Capítulo 2

Metodologia e Planeamento

2.1 Metodologia

Durante o desenvolvimento de qualquer produto, é sempre importante que a equipa responsável determine, antes de qualquer desenvolvimento, o método de trabalho que irão utilizar para alcançar o objetivo pretendido.

Como tal, a presente secção tem como objetivo expor a metodologia de trabalho seguida na empresa Grama no decorrer do estágio curricular.

2.1.1 *Agile*

Durante o desenvolvimento do presente trabalho, foi usada uma **metodologia ágil**. *Agile* é um método de trabalho que, ao contrário de metodologias tradicionais como *waterfall*, é baseado num desenvolvimento iterativo e rápido, entregando sequencialmente funcionalidades novas, com qualidade e de acordo com as necessidades do cliente.

Em 2001, com a necessidade de repensar a forma como os produtos eram desenvolvidos, pois estes demoravam muito tempo até terem um resultado visível, surgiu o **Manifesto Ágil**. Neste, um conjunto de *developers* reuniu quatro valores que sumarizam a filosofia de uma metodologia ágil [8], são estes:

- **Indivíduos e interacções** mais do que processos e ferramentas.
- **Software funcional** mais do que documentação abrangente.
- **Colaboração com o cliente** mais do que negociação contratual.
- **Responder à mudança** mais do que seguir um plano.

2.1.2 *Scrum*

Por fim, a equipa optou por uma metodologia ágil baseada em ***scrum***. Em adição aos valores existentes numa metodologia ágil, a metodologia *scrum* é assente em mais três fundamentos [1] :

- **Transparência**: cada membro da equipa tem que ter acesso a toda a informação do produto.
- **Inspeção**: validações regulares são essenciais para reajustar o projeto.

- **Desvios**: a implementação de novas medidas é necessária quando a inspeção mostra desvios dos resultados esperados.

Assim, para cumprir os valores descritos em cima, a metodologia ágil *scrum* é estruturada em vários períodos temporais (normalmente de 2 a 4 semanas) denominados ***sprints*** onde a equipa desenvolve um número de tarefas presentes numa lista de tudo o que se sabe ser necessário no produto. Esta lista é gerida pelo *product owner* e chama-se, ***backlog***.

Por fim, no final de cada *sprint*, existe uma nova ***release***, ou seja, uma nova versão do produto com as novas funcionalidades desenvolvidas durante a presente *sprint*.

Roles

Para agilizar o processo de gestão do produto, existem três papéis (ou *roles*) fundamentais e complementares numa metodologia *scrum*.

São estes:

- **Scrum Master**: atua como líder da equipa e ajuda toda a equipa com quaisquer dificuldades.
- **Product Owner**: elemento responsável pela ligação entre a área de negócio e a área técnica. É também responsável por criar as *user stories* e manter o *backlog* atualizado.
- **Equipa de Desenvolvimento**: responsável por desenvolver as funcionalidades listadas no *backlog*.

No contexto do estágio, o aluno esteve incluído na equipa de desenvolvimento em conjunto com um *designer* responsável por desenhar os elementos visuais da plataforma e, apesar do aluno não ser o *product owner*, teve contacto com a criação das *user stories* como forma de aprendizagem e facilidade na escrita do presente relatório.

Cerimónias

Por fim, a metodologia *scrum* também define três cerimónias essenciais para o sucesso de cada *sprint* (Fig. 2.1).

Em primeiro lugar, no início de cada *sprint*, a equipa deverá reunir-se para delinear que tarefas deverão ser implementadas na mesma, tendo em conta a prioridade da tarefa e o tempo que esta irá consumir.

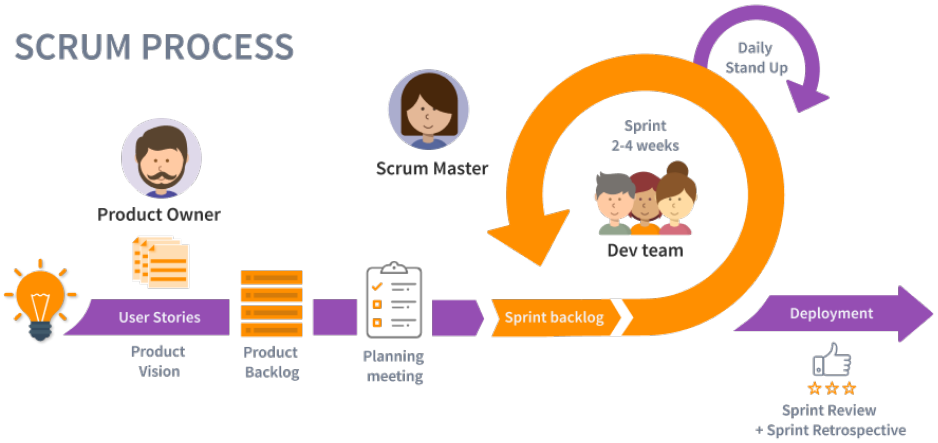


Figura 2.1: Metodologia ágil, *Scrum* [1]

De seguida, durante o recorrer da *sprint*, a equipa deverá reunir-se diariamente para partilhar com os restantes membros os avanços conseguidos no dia anterior e o planeamento para o dia atual. Esta reunião, chamada ***daily stand-up***, foi realizada todos os dias durante o segundo semestre para haver um acompanhamento rigoroso do processo de desenvolvimento, porém, durante o decorrer do 1º semestre, foi feita semanalmente. Esta decisão foi tomada por não haver a necessidade de reuniões diárias, visto que, a equipa é pequena e as tarefas previstas para o 1º semestre não necessitaram de um acompanhamento diário. Em adição, durante o 1º semestre, o aluno apenas trabalhou com a empresa em tempo parcial, pelo que, não se encontrou na empresa todos os dias da semana.

Finalmente, dado o fim da *sprint*, a equipa deverá, mais uma vez, reunir-se para fazer uma retrospectiva da mesma de modo a incentivar a melhoria contínua da equipa.

2.1.3 Ferramentas e comunicação

De modo a haver um conhecimento geral do trabalho realizado pela equipa e do estado atual do projeto, é importante para uma equipa de desenvolvimento ter uma rotina de partilha de informação ativa e constante, para tal, a Grama incentiva sempre o uso de diversas ferramentas de comunicação, complementadas com reuniões semanais entre toda a equipa.

Assim, no decorrer do primeiro semestre, foi utilizado o ***Trello*** como um quadro ***Kanban*** (Fig.2.2) para criar e manter o *backlog*, delinear tarefas para a *sprint*, associar essas mesmas tarefas a algum elemento da equipa e, por fim, listar todas as tarefas concluídas.

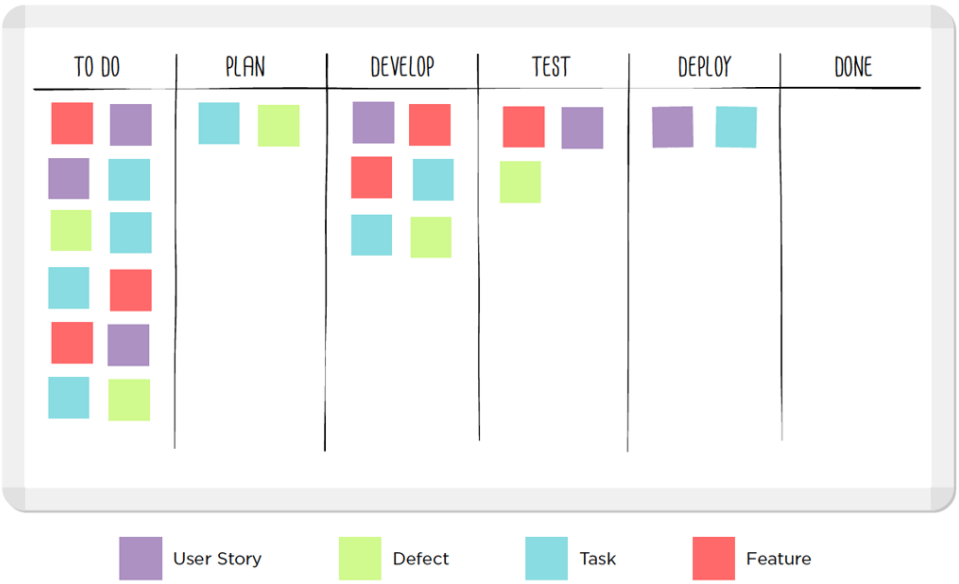


Figura 2.2: Quadro *Kanban* [2]

De forma a conhecer uma ferramenta nova e inovadora, a equipa tomou a decisão de, no decorrer do segundo semestre, substituir a ferramenta ***Trello*** pela ferramenta ***Linear***. Apesar do objetivo de ambas as plataformas ser semelhante (servir como um quadro *Kanban* para organização das tarefas), a nova ferramenta contém um conjunto de funcionalidades bastante apelativas que a ferramenta *Trello* não oferece, como por exemplo, uma análise gráfica do esforço efetuado ao longo do *sprint* (estes gráficos foram utilizados, inclusive, na secção [2.2.2]), vários outros gráficos de progressão, inúmeras novas formas de classificar um *issue*, entre outras.

Em adição, foram utilizadas diversas ferramentas de comunicação como o ***Slack*** para enviar/receber mensagens de elementos da equipa ou mesmo da empresa no geral, o ***Webmail*** para troca de documentos ou marcação de reuniões e o ***Discord*** ou ***Google Meetings*** como forma de comunicação ativa em ambiente remoto.

Por fim, uma vez por mês, realizaram-se reuniões entre o estagiário e ambos os orientadores que serviram de apoio ao aluno e eventuais correcções ao trabalho efectuado até à data.

2.1.4 Fluxo de trabalho

Durante todo o período de desenvolvimento foi utilizado um fluxo baseado em ***gitflow*** (Fig. 2.3) cujo objetivo é repartir as funcionalidades em *branches* de trabalhos distintos de forma a organizar o fluxo de trabalho e a facilitar os *code reviews* futuros.

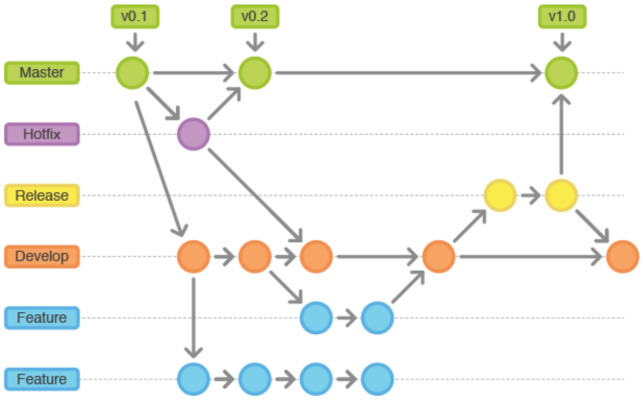


Figura 2.3: Organização comum *gitflow* [3]

Assim, durante o período de desenvolvimento, o aluno teve um fluxo de trabalho mais simplificado do que o apresentado anteriormente (Fig. 2.3), sendo que, o *branch master* e o *branch develop* foram fundidos de forma a apenas haver um *branch* principal, o *develop*. A partir deste, foram criados novos *branches* por cada funcionalidade nova que, por sua vez, foram repartidos em diferentes tarefas (ver Fig. 2.4).

Em suma, podemos repartir fluxo de trabalho nos seguintes passos:

1. Criação de um *branch* a partir do *branch develop*, para implementar uma nova funcionalidade. Esta poderá ter uma ou mais tarefas associadas que correspondem a um novo *branch*.
2. Após a funcionalidade estar completa será feito um *merge-request* para o *branch develop* que será avaliado por um outro membro da equipa, geralmente e se possível, um elemento sénior.
3. Caso o *merge-request* não seja aprovado, o código terá de ser alterado para estar de acordo com o pedido durante a avaliação do mesmo.
4. Caso o *merge-request* seja aprovado, o *branch develop* será atualizado com a nova funcionalidade.
5. No final do *sprint* será feito um *deployment* com o conteúdo presente no *branch* pai (*develop*).

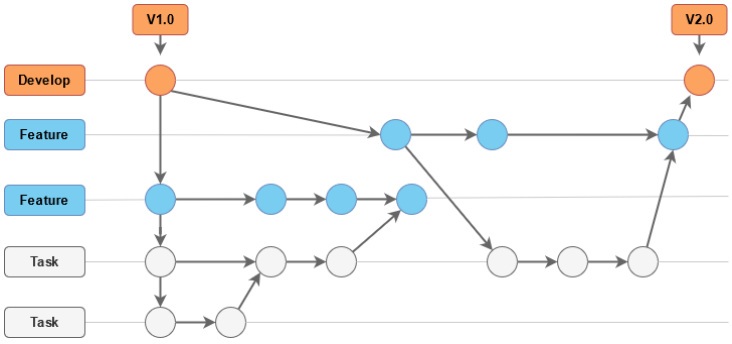


Figura 2.4: Organização *gitflow* do Sala-z

2.2 Planeamento

Antes de começar cada um dos semestres, foi criado um diagrama de *Gantt* para, assim, estruturar o trabalho e determinar *deadlines* das tarefas delineadas para cada um dos mesmos.

Em adição, de modo a haver um termo comparativo entre o planeamento e o decorrer real dos semestres, no final dos mesmos, foi analisado o trabalho feito para concluir se o semestre correu de acordo com o planeamento e, caso contrário, perceber o que aconteceu de negativo servindo de aprendizagem para ocasiões futuras.

Por fim, foi estruturado o plano de testes a levar a cabo no decorrer do segundo semestre.

2.2.1 1º Semestre

Apesar do projeto seguir uma metodologia ágil, o primeiro semestre serviu para entender o conceito da plataforma e o problema existente nesta área específica da cultura, assim como, serviu para planear detalhadamente o processo de desenvolvimento levado a cabo no segundo semestre.

Planeamento inicial

Numa fase inicial do presente estágio, foi elaborado um *Gantt Chart* de modo a planear o primeiro semestre. Neste, o aluno baseou-se no plano proposto pela empresa, assim como, em reuniões em que o tema foi abordado. Assim, o seguinte diagrama foi elaborado (Fig 2.5):

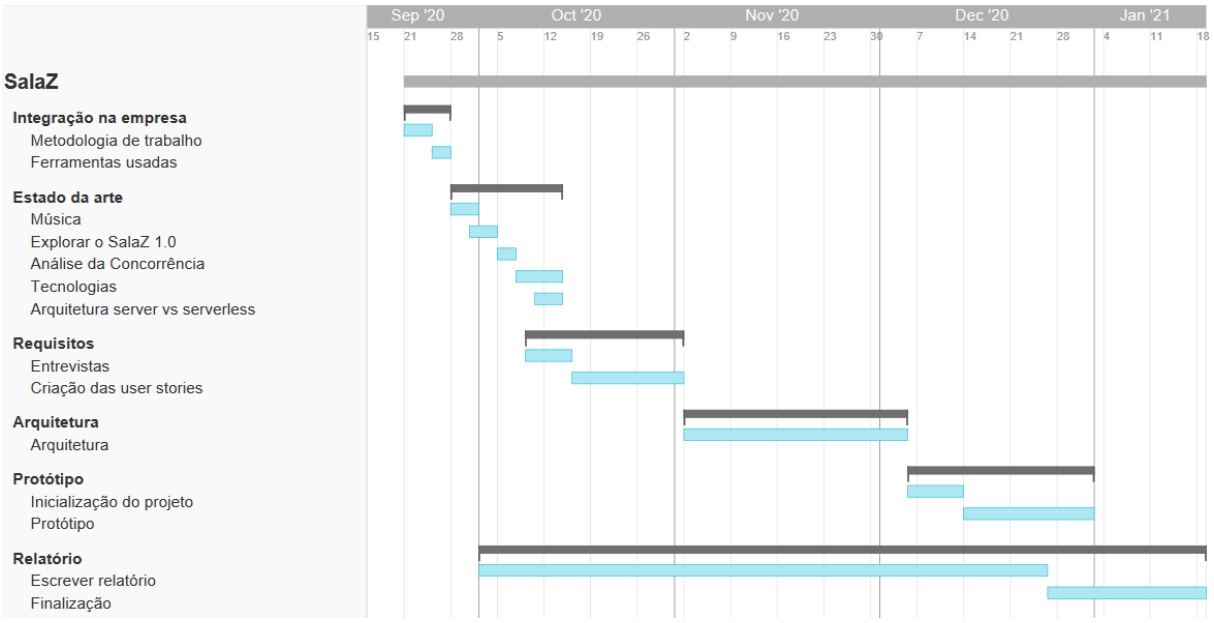


Figura 2.5: Planeamento inicial do 1º Semestre

Cronologia real

De modo a visualizar as alterações que surgiram no decorrer do semestre contra o plano realizado no início do mesmo, o aluno criou outro *Gantt Chart* com o decorrer real do semestre. Assim, o resultado foi o seguinte (Fig 2.6):

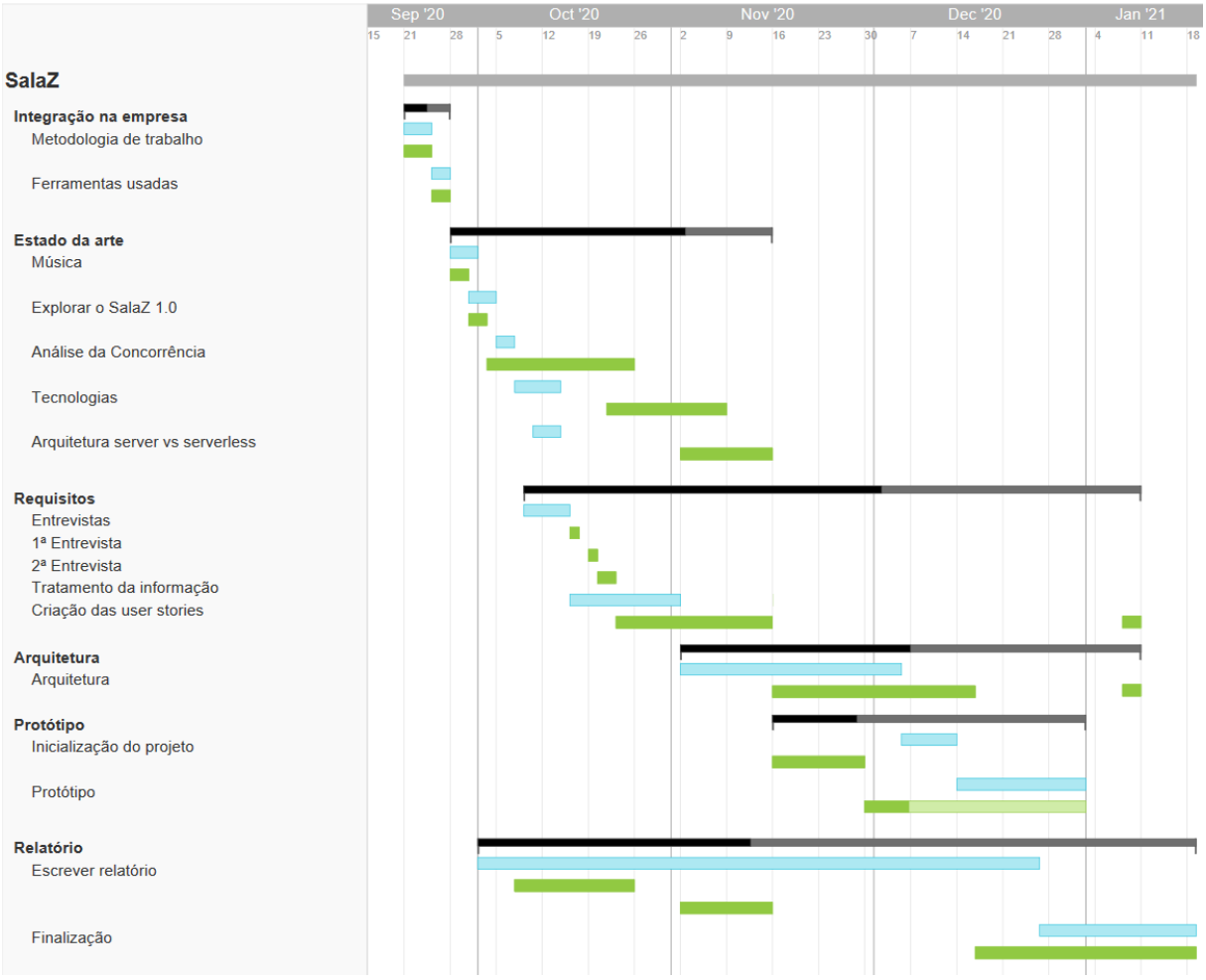


Figura 2.6: Comparação entre o plano inicial (azul) e o decorrer real (verde)

Conclusões

Podemos concluir que, pela análise dos diagramas, houve alterações e atrasos no que toca ao plano elaborado no início do semestre.

Em primeiro lugar, devido à pandemia e à pouca disponibilidade dos possíveis clientes, as entrevistas começaram mais tarde do que o planeado, pelo que, todo o processo de levantamento de requisitos também sofreu atrasos. Este atraso serviu, no entanto, para consolidar o estudo da concorrência.

Em segundo lugar, devido ao facto de ter sido dada uma grande liberdade ao estagiário para escolher as tecnologias a usar no desenvolvimento do produto, o mesmo demorou mais tempo do que o planeado a estudar a melhor solução.

Apesar do facto da fase de requisitos ter-se atrasado em relação ao plano inicial, a equipa conseguiu remediar esse atraso, sendo que, apenas o protótipo foi deixado incompleto. Este factor não apresenta um problema pois, apesar do protótipo não ter sido acabado, o objetivo deste era o estagiário conseguir adquirir conhecimentos e experiência com as tecnologias a utilizar no semestre seguinte.

Por fim, houve um alteração nos requisitos na fase final do 1º semestre, o que levou a uma alteração do documento de requisitos e da arquitetura.

2.2.2 2º Semestre

O segundo semestre teve como objetivo desenvolver a plataforma no seu todo. Para tal, o semestre foi repartido em períodos temporais de duas semanas (*sprints*). Cada *sprint* começou por ser planeado, ou seja, foi feita uma reunião geral onde foram delineadas as tarefas que deveriam ser desenvolvidas durante as duas semanas seguintes. Após o *sprint* acabar foi feita uma retrospectiva onde foram abordados os aspectos positivos e negativos da *sprint* de tal forma a haver uma evolução progressiva da equipa e do método de trabalho.

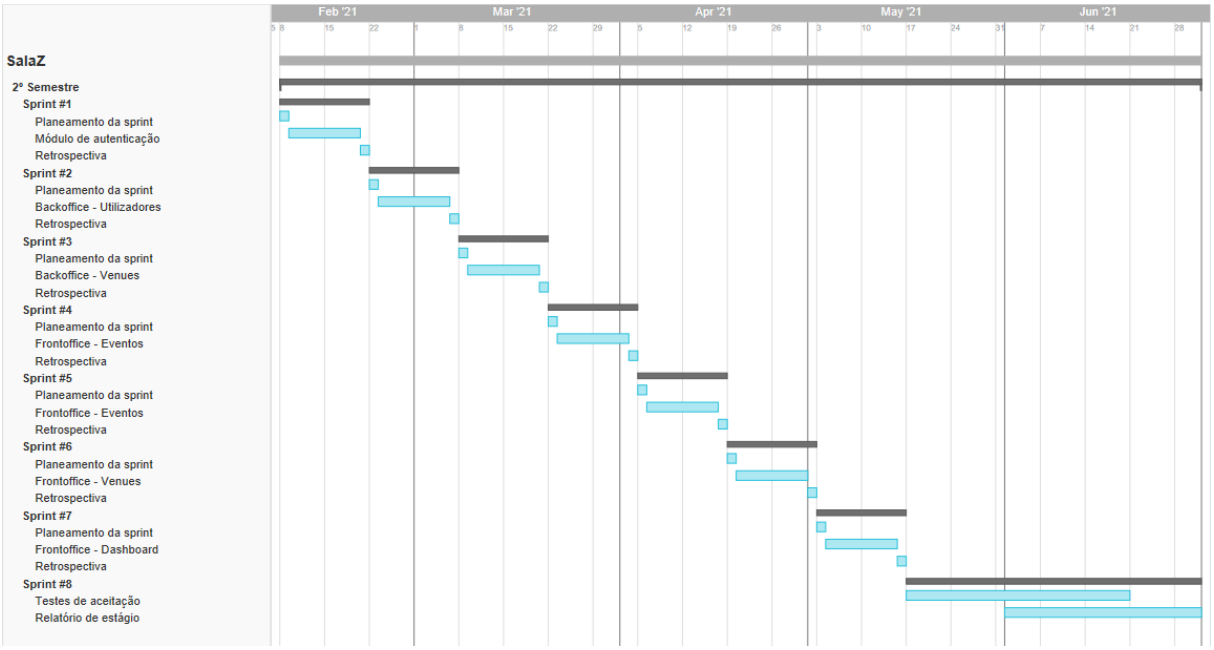


Figura 2.7: Gantt Chart - 2º Semestre 1ª iteração

Cronologia real

Assim, o segundo semestre foi repartido em 7 *sprints* (chamados "ciclos"na plataforma ***Linear***) de desenvolvimento e um *sprint* apenas dedicado aos testes e conclusão dos testes de aceitação. Assim, no inicio de cada *sprint* de desenvolvimento, foi feito um planeamento *poker* onde, para cada tarefa, é atribuído um valor que irá corresponder ao esforço estimado que a tarefa terá. Cada estimativa é adquirida depois de todos os elementos da equipa concordarem com a mesma.

Este valor poderá ser:

1 - Tarefa que demora 1 hora

2 - Tarefa que demora meio dia

3 - Tarefa que demora um dia

5 - Tarefa que demora 3/4 dias

Em adição, como é possível visualizar no conjunto de figuras abaixo (Fig.2.11(a) a Fig.2.11(g)), cada um dos períodos temporais é representado por um gráfico que apresenta o esforço dedicado ao longo do tempo. Para tal são apresentadas as seguintes informações em cada um dos gráficos:

Scope - Apresentado com uma linha a cinzento, o *scope* representa o esforço delineado, geralmente, no início de cada *sprint*. O objetivo de cada *sprint* será, então, atingir esta linha no final do

mesmo. Em adição, o *scope* poderá sofrer pequenas alterações ao longo do *sprint* pois, por vezes, a equipa sentiu a necessidade de remover ou adicionar tarefas estimadas por motivos diferenciados (e.g: demasiado esforço, algum membro ia de férias, entre outros).

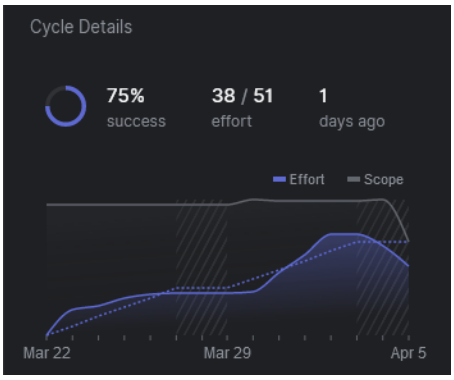
Effort - Apresentado com uma sombra azul, representa o progresso efetuado ao longo do semestre. Para termos comparativos, a ferramenta *Linear* inclui uma linha azul tracejada para entender se o esforço está abaixo ou acima do esperado.

Percentagem de sucesso - Percentagem de tarefas acabadas no final do *sprint*.

(a) 1º *sprint*

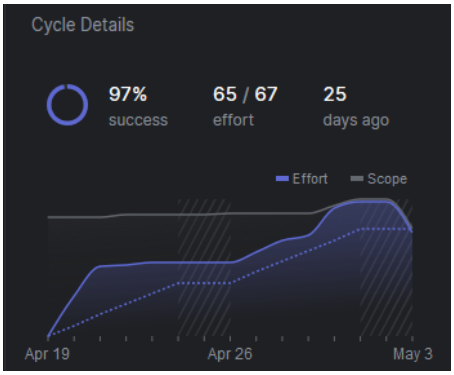
(b) 2º *sprint*

(c) 3º *sprint*

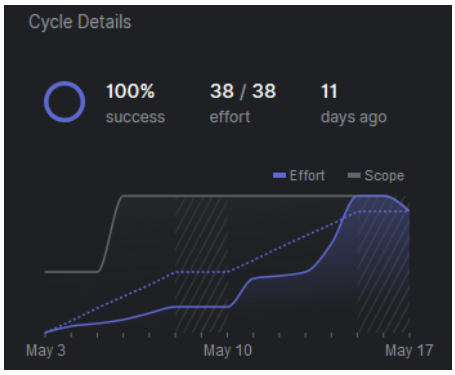


(d) 4º *sprint*

(e) 5º *sprint*



(f) 6º *sprint*



(g) 7º *sprint*

Figura 2.11: Decorrer do 2º semestre

Conclusões

Como podemos observar pelas figuras 2.11(a) a 2.11(g) as estimativas nos primeiros *sprints* foram pouco corretas, sendo que, a percentagem de sucesso apenas ultrapassou os 65% no 4º *sprint*. Este aspecto foi normal e previsível visto que, no início do período de desenvolvimento, o aluno estava numa fase de integração e aprendizagem, assim como, a equipa ainda não tinha informação suficiente para saber o ritmo de trabalho do aluno.

Assim, tendo em consideração que o 5º *sprint* foi um *sprint* extraordinário devido ao facto do *scrum master* estar de férias e haver um atraso na aprovação dos *merge request* devido a essa situação, podemos analisar que a taxa de sucesso aumentou progressivamente até ao 6º *sprint* onde a percentagem de sucesso foi perto de 100%. Este crescimento progressivo da percentagem de sucesso prova que houve uma aprendizagem progressiva dentro da equipa que, a cada *sprint*, perceberam melhor o ritmo de trabalho do aluno e conseguiram estimar de maneira mais correta a *sprint* seguinte.

Por fim, podemos reparar que alguns *sprints* tiveram mais esforço planeado que os anteriores numa situação que não faz sentido esse aumento, como, por exemplo, do *sprint* nº5 (Fig. 2.11(e)) para o *sprint* nº6 (Fig. 2.11(f)) em que houve um acréscimo de 54 pontos para 67 pontos quando a taxa de sucesso do *sprint* 5 foi de apenas 44%. Este aumento de esforço deu-se pois, uma tarefa apenas é dada como completa no terminar do *sprint* quando passa de *"In review"* para *"Ready to test"* e, como muitas tarefas do *sprint* 5 não foram revistas antes do *sprint* acabar, essas mesmas passaram para o *sprint* seguinte automaticamente o que, por vezes, resultava numa nova *sprint* já com esforço designado antes do planeamento acontecer. Este acontecimento recorrente foi tido em consideração e foi corrigido ao longo do semestre, tendo o aluno terminado as tarefas com mais antecedência para poder haver tempo suficiente para os *code reviews* antes do *sprint* acabar.

2.2.3 Planeamento dos testes

Em qualquer produto de *software*, a componente de qualidade do mesmo é altamente importante para o seu sucesso no mercado. Assim, a atividade de testar sistematicamente a plataforma em busca de falhas, representa uma papel fundamental em qualquer processo de desenvolvimento de um produto de *software* para garantir que o mesmo é, não só funcional, mas estável e de qualidade.

Assim, em adição ao planeamento temporal feito ao longo da presente secção, foi feito, também, um planeamento dos testes que terão de estar presentes no decorrer do segundo semestre.

No contexto do estágio, o aluno programou testes automatizados que, em paralelo com os testes manuais realizados pela restante equipa da Grama, foram corridos no final de cada *sprint* com o objetivo de encontrar quaisquer falhas nas funcionalidades desenvolvidas ao longo da mesma.

Testes Automatizados

De modo a testar a plataforma de forma automática, o estagiário seguirá a filosofia de testes normal da empresa. Como tal, irá seguir uma abordagem híbrida desenvolvendo testes de **integração** com dependências ***mock***.

Mocking consiste em criar uma versão falsa de um serviço interno ou externo que substituirá o serviço real de forma a ajudar o processo de testes.

Por outro lado, um teste de **integração** consiste em testar módulos individuais do *software* em grupos, de forma a validar a comunicação e transferência de informação entre estes grupos.

Testes Manuais

Para complementar os testes automatizados, serão feitos, no final de cada *sprint*, ***Smoke tests*** às funcionalidades implementadas durante a mesma.

Um *Smoke test* (também chamado de "*Build Verification Testing*"ou "*Confidence Testing*") é um teste de *software* cujo objetivo é garantir que as funcionalidades implementadas estão estáveis [9].

Em adição, no final do semestre, foram realizados **testes de aceitação** por parte da equipa da Grama. O objetivo dos mesmos será testar todas as funcionalidades do produto de modo a garantir que as mesmas estão de acordo com o planeado. Assim, será feito um plano de testes de acordo com os requisitos descritos na Secção 4.3 em que cada teste realizado irá conter a seguinte informação:

- **ID** : ID do teste (e.g.: Test-001).
- **Cenário** : Cenário da funcionalidade testada (e.g.: Login com credenciais válidas).
- **Descrição** : Descrição do teste (e.g.: Entrar na aplicação com credenciais corretas).
- **Passos** : Os passos necessários para realizar o teste.
- **Resultado espectável** : Resultado esperado (e.g.: Entrar na *homepage*).
- **Resultado real** : Resultado real (e.g.: Entrou na *homepage*).
- **Status** : Status do teste (e.g.: Passou).

Capítulo 3

Estado da arte

O presente capítulo explora todos os temas que estão associados ao âmbito do projeto Sala-Z para, assim, criar uma base de conhecimentos que possibilitaram compreender todo o contexto que rodeia o tema do projeto. O capítulo está divido em cinco secções, são estas:

A **primeira secção** explora o conceito de *venues* explicando o que são e como os tipos e características de cada uma influencia os seus eventos musicais.

Na **segunda secção** é explorada a plataforma antiga do Sala-Z, explorando quais as suas funcionalidades e como estas serão aproveitadas no desenvolvimento da nova plataforma.

A **terceira secção** apresenta os principais concorrentes ao Sala-Z. Nesta secção será analisado cada um destes concorrentes, explorando o que estes fornecem aos seus clientes com o seu produto e comparando os mesmos com o que a plataforma Sala-Z fornece aos seus clientes, tentando concluir porque é que esta plataforma de gestão de salas de espetáculo é diferente e potencialmente melhor que as suas competidoras.

Em adição, na **quarta secção**, será demonstrado o estudo feito em relação às tecnologias atuais mais usadas por equipas de desenvolvimento de *software* por todo o mundo. Estas tecnologias serão comparadas e será feita uma decisão na escolha de tecnologia de *frontend* e *backend* usada no desenvolvimento da nova versão do Sala-Z.

Por fim, na **quinta secção**, serão analisados os riscos da plataforma e as possíveis estratégias de mitigação.

3.1 *Venues*

Uma *venue* é um local que está preparado para receber um ou vários artistas para terem uma atuação para um público grande ou pequeno. Desde as *venues* mais pequenas, como bares, até grandes salas de concertos, existem vários tipos de *venues*, porém, nem todas são apropriados para qualquer tipo de música pelas limitações que este pode apresentar (a nível de sonoridade por exemplo).

Neste capítulo serão expostas as características e tipos de *venues*, apresentando os pontos fortes e fracos de cada um e quais os estilos musicais mais apropriados para as mesmas.

Características de uma *venue*

Existem características num espaço destinado a concertos que têm de ser tidas em consideração antes dos gerentes do mesmo contratarem um grupo de artistas.

Em primeiro lugar, o tamanho do espaço. Grandes artistas de *pop* ou *rock* por exemplo, atraem muitas

pessoas e cobram valores muito elevados, pelo que, normalmente, atuam em grandes palcos onde existem plateias preparadas para albergar centenas ou milhares de pessoas. Por outro lado, espaços mais pequenos, como bares, estão preparados para estilos musicais mais casuais como bandas *jazz* ou como *DJs*.

Em segundo lugar, o facto do espaço ser coberto ou não é uma limitação para alguns tipos musicais, por exemplo, uma orquestra clássica necessita de uma sala acusticamente preparada, por outro lado, uma banda de *rock* não necessita, necessariamente, de atuar num local fechado.

Por fim, o horário de atuação também poderá influenciar o estilo de música. Por vezes bares ou discotecas que estão abertos durante a noite têm maior probabilidade de contratar artistas como *DJs* pois a faixa etária é mais jovem, enquanto que, locais que estão abertos durante o dia, tendem a contratar mais músicos populares pois o público tende a ser mais repartido por todas as idades.

Tipos de *venues*

Existem vários tipos de *venues*, porém, podemos analisar as mais vulgares dentro da sociedade, são estes:

Os bares - são locais pequenos que têm pouca capacidade e podem ter uma componente mais de relaxamento em que a música é tocada baixo para os clientes poderem falar e estar, são exemplos, bandas de *jazz*, *DJs*, fado, entre outros. Por outro lado, também poderão ter uma componente mais de dança, geralmente em ambiente noturno, onde os alvo são jovens o que pode influenciar música mais eletrónica, *rock*, *hip hop*, entre outros.

Estádios e arenas – espaços aberto para uma plateia enorme, geralmente é onde grandes artistas/bandas atuam.

Salas de concertos/ópera – espaços que são direcionados para grandes orquestras ou para óperas onde a importância está na qualidade acústica da sala pois, geralmente, os instrumentos que atuam nestes locais não têm amplificação eletrónica, o que, em condições de acústica medíocre, complica tanto para o público como para o artista, ouvir com boa qualidade o som dos restantes instrumentos.

Anfiteatros – são espaços fixos e ovais, geralmente ao ar livre, onde podem atuar vários tipos de artistas que não necessitem das melhores condições acústicas.

Obviamente poderíamos pensar e encontrar mais locais onde existem espetáculos, porém, serão parecidos aos já referidos anteriormente de alguma maneira.

Concluindo podemos dizer que um espaço terá algumas componentes chave que nos ajudam a perceber qual o melhor tipo de artista para atuar no mesmo, são estes: se tem cobertura ou não, o horário das atuações normais, a preparação acústica da sala, e o tamanho do palco e da plateia.

3.1.1 Logística de um espetáculo

Para um evento ser bem sucedido é importante que o mesmo seja bem organizado, seja bem conduzido e seja bem finalizado. É fácil para um mero participante dizer que o evento começa quando o artista que querem ouvir pisa o palco ou que acaba quando a última luz do recinto desliga, porém, para um organizador/promotor de um evento, o mesmo começa muito antes. Desde os planos que lidam ao evento até ao último aperto de mão, a organização de um evento é um processo complicado.

Em primeiro lugar, promover um evento musical é um processo dispendioso e árduo. Geralmente um evento é constituído por vários intervenientes que podem ser artistas, equipas técnicas de suporte (como luzes ou som), gerentes de locais para eventos, entre muitos outros. Em adição, muitos promotores de eventos têm a responsabilidade acrescida de encontrar os melhores locais possíveis para realizar os mesmos, tendo sempre de considerar o tamanho e se se adapta ao tipo de evento que está a ser organizado, por exemplo, um concerto de *jazz* poderá ser realizado num ambiente fechado como um bar, enquanto que, um concerto *pop/rock* poderá ser realizado num palco grande ao ar livre que necessita de imenso espaço para o público poder estar a assistir. Em resumo, promover um evento é um processo complexo que depende

de muitas variáveis para ser bem sucedido, sendo que, um dos maiores trunfos de qualquer promotor de eventos é a sua base de contactos pois, sem eles, os promotores serão obrigados a usar canais como email ou números de telemóvel profissionais para chegar ao indivíduo/espaço que desejam negociar.

Em segundo lugar, gerir o evento em si é um processo árduo. É importante para um gerente saber exatamente como está a correr o evento, isto poderá passar por saber a receita do evento, saber se o seu espaço já está cheio ou ainda pode vender mais bilhetes (se este for o caso) saber quem são os elementos do *staff*, tratar de todas as licenças ou contratos necessários, entre outros.

Por fim, é sempre importante dar por fechado o evento com o pé direito. Garantir que todos os artistas e *staff* são pagos e garantir que todos os contratos ou qualquer outro tipo de documentos estão completamente tratados e guardados, por vezes, podem ser tarefas facilmente esquecidas se não houver um processo de gestão bem delineado.

Agentes

Assim, durante o processo de organização de um evento musical, são vários os agentes que participam no mesmo. Entre os quais podemos salientar os seguintes:

Gerente da *venue* - Responsável por uma *venue* (que pode ser uma sala de espetáculos, um bar com música ao vivo, entre outros...), comunica com promotores de eventos, agentes ou diretamente com artistas para organizar eventos musicais nos seus estabelecimentos. São responsáveis pela bilhética e por determinar se a sala consegue albergar mais público.

Promotor de eventos - Responsável por planear, promover e realizar um evento. Geralmente comunica com gerentes das *venues* para arranjar locais para realizar os mesmos e com agentes/artistas para atuarem.

Agentes de artistas - Responsáveis por angariar eventos para os seus artistas (recebem ofertas de promotores de eventos e de gerentes de *venues*, porém, também podem procurar eventos para os seus artistas e convencer os gestores das *venues* que o artista consegue encher a mesma). Normalmente, também são responsáveis por negociar contratos e organizar viagens, estadias, entre outros.

Artistas - O músico que pode, ou não, ter um agente que o represente e que, geralmente, procura ou é requisitado para eventos.

Staff - Toda a equipa que ajuda na organização e condução do evento.

3.2 Sala-Z 1.0

Como já referido anteriormente, a empresa Grama desenvolveu, parcialmente, a plataforma Sala-Z (Fig. 3.1), porém, esta plataforma apresentou alguns problemas no seu desenvolvimento sendo arquivada pela equipa da Grama.

Como forma de revitalizar o projeto, foi criado um estágio para recriar de raiz a ideia já pré estabelecida na primeira versão do produto. Assim, a segunda versão da plataforma Sala-Z é uma evolução da plataforma já existente, com novas funcionalidades e um foco diferente, por outras palavras, a nova versão será focada na gestão das *venues* e não apenas na listagem de *venues* existentes. Contudo, algumas funcionalidades da primeira versão estarão presentes na nova versão do produto e poderão servir de base de conhecimento para o aluno desenvolver a segunda versão da plataforma.

Assim, na presente secção, serão apresentadas as funcionalidades da plataforma já existente e será feita uma análise das diferenças que a nova versão irá conter em relação à versão anterior.

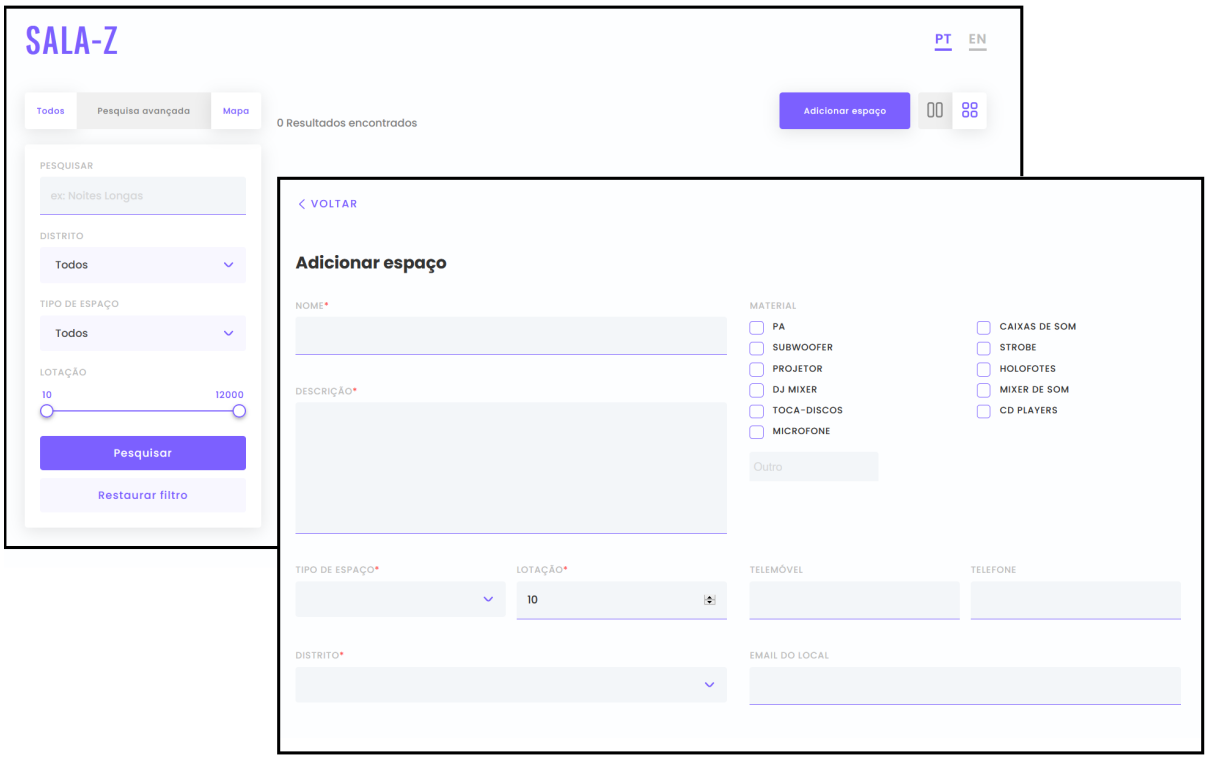


Figura 3.1: Primeira versão da plataforma Sala-Z [4]

3.2.1 Funcionalidades

A plataforma antiga do Sala-Z teve como foco principal a criação e visualização de espaços para eventos. Com um *design* e funcionalidades simples, o Sala-Z v1.0 permite:

- **Autenticação** - como visitante da plataforma posso autenticar-me na plataforma.
- **Listar *venues*** - como visitante da plataforma posso listar todas as *venues* ou listar *venues* de acordo com um filtro:
  - Listar por nome
  - Listar por distrito
  - Listar por tipo
  - Listar pela lotação
- **Ver detalhes** - Como visitante, posso ver os detalhes de uma *venue*.
- **Criar uma *venue*** - autenticado na plataforma, posso enviar um pedido de criação de uma nova *venues* na plataforma com:
  - Nome do espaço
  - Descrição
  - Tipo
  - Material que a *venue* fornece aos artistas
  - Lotação
  - Contactos (email, telemóvel, ...)
  - Local (distrito, cidade, ...)
  - Imagem
- ***Backoffice*** - Como membro da equipa de *backoffice*, posso aprovar ou negar um pedido de criação de *venue*.

3.2.2 Reutilização de componentes

Apesar das tecnologias usadas no desenvolvimento da nova versão do Sala-Z serem distintas das tecnologias usadas no desenvolvimento da primeira versão, algumas das funcionalidades que irão ser implementadas serão semelhantes às funcionalidades já desenvolvidas na versão anterior, pelo que, as mesmas, poderão servir de referência para o aluno.

Assim, no final do primeiro semestre e no decorrer do segundo semestre, o estagiário estudou o repositório antigo da plataforma com o objetivo de aprender a estruturar um projeto e as melhores práticas de desenvolvimento. Com este conhecimento, o aluno passou a desenvolver a plataforma podendo reutilizar componentes como:

- **Autenticação** - Como o processo e as ferramentas de autenticação (como a utilização do serviço *Cognito* [10] da *Amazon Web Services* (AWS) para autenticação dos utilizadores) são semelhantes, o aluno poderá utilizar o código antigo como referência para a nova versão do produto.
- **Listagem e criação de *venues*** - Como na versão antiga do Sala-Z, é possível criar e listar *venues*, pelo que, foram reutilizados muitos dos parâmetros de criação e filtros de listagem utilizados na versão anterior.
- ***Backoffice*** - Apesar do novo *backoffice* ser distinto do *backoffice* do Sala-Z v1.0, existirão muitas semelhanças entre ambos, pelo que, também será uma fonte de referência para o aluno.

3.3 Produtos concorrentes

Para realizar o estudo dos concorrentes diretos e indiretos à plataforma Sala-Z, foi realizada uma análise do mercado para perceber se existem tecnologias semelhantes à que está a ser desenvolvida e perceber as abordagens tomadas pelas mesmas.

Assim, o objetivo deste capítulo será, com o agregado de informação obtido sobre os concorrentes, perceber no que o Sala-Z se distingue das outras plataformas semelhantes de forma a criar uma proposta de valor interessante a possíveis futuros clientes.

Em primeiro lugar, temos a plataforma ***Gigwell*** [11] que oferece aos artistas três produtos distintos que possibilitam procurar novos eventos ou *tours*, gerir todo o processo de negociação com possíveis *venues* interessadas em abrigar o artista e, por fim, permite contabilizar os bilhetes comprados para o espetáculo oferecendo uma análise estatística do mesmo.

Em adição, a plataforma ***Gigplanner*** [12] fornece aos artistas inúmeras ferramentas de gestão pessoal e de eventos em que irão participar. Apesar de não apresentar muitas funcionalidades e de ser apenas direcionada aos artistas, a plataforma da *Gigplanner* fornece aos artistas uma forma de gerir o seu negócio pessoal bastante evoluída.

Por outro lado, e como maior concorrente no cenário nacional, temos o ***Coliseum*** [13] da empresa *Empis acessos* [14]. A *Empis* é uma empresa de bilhética portuguesa que fornece suporte a eventos tendo maior foco no suporte na gestão de acessos e venda de bilhetes. Visto isto, a *Empis* desenvolveu uma aplicação de gestão de salas de espetáculo que, em adição à gestão de venda e compra de bilhetes, gere eventos (data, classificações, ...), locais (definição de filas e do recinto em si) e gestão de caixa.

De seguida, temos o ***Prism.fm*** [15]. É um *software* bastante evoluído que se foca no agendamento, organização e gestão de eventos musicais oferecendo ao cliente (equipa que organiza o evento) a possibilidade de procurar artistas baseado em dados do *Spotify*, enviar pedidos de atuação ao artista, receber *feedback* dos agentes sobre o histórico dos artistas, etc.

Por fim, a plataforma ***Muzeek*** [16] foca-se no agendamento e gestão de eventos musicais. Destaca-se pelo sua forte componente de automação, tornando todos os passos de gestão de eventos mais rápidos e fáceis de realizar. Em adição, é uma plataforma com vários planos que contém, também, uma opção gratuita.

3.3.1 Comparação entre plataformas

Através da exploração e análise feita a cada uma das plataformas concorrentes ao Sala-Z, foi elaborada uma tabela (Tabela 3.1) para sumarizar e comparar as mesmas a nível de funcionalidades e, também, pontos fortes e pontos fracos de cada uma. Em adição, foi adicionada a antiga versão do Sala-Z para entender as diferenças que esta terá em comparação com a nova versão da plataforma.

	Gigwell	**Gigplanner**	**Coliseum**	**Prism.fm**	**Muzeek**	**Sala-Z 1.0**	**Sala-Z 2.0**
Direcionada a:	Artistas Promotores de eventos	Artistas	Gerentes de venues	Gerentes de venues Promotores de eventos	Promotores de eventos Gerentes de venues Artistas	Artistas	Gerentes de venues
Gestão da banda/artista	✗	✓	✗	✗	✓	✗	✗
Procura de artistas	✗	✓	✗	✓	✓	✗	✓
Gerir *venues*	✗	✗	✓	✓	✓	✗	✓
Listar *venues*	✓	✗	✗	✗	✗	✓	✗
Visualizar/gerir eventos	✓	✓	✓	✓	✓	✗	✓
Ver/gerir receita do evento	✓	✗	✓	✓	✓	✗	✓
Estatísticas monetárias	✓	✗	✗	✓	✓	✗	✓
Criação automática de documentos (e.g. contratos)	✓	✗	✗	✓	✓	✗	✗
Venda de bilhetes	✗	✗	✗	✗	✓	✗	✗
Integração com plataformas de música (e.g. Spotify)	✓	✗	✗	✓	✓	✗	✓
Fortes:	Plataforma muito completa para artistas	Detalhe na gestão do artista	Bilhética	Gerar ofertas e análise feita em tempo real Integração com o Spotify	Plataforma bastante completa com possibilidade de ter um plano grátis	Repositório de *venues* usado por artistas para encontrar locais para atuar	Gestão detalhada da *venue* e dos eventos que a mesma alberga
Fracos:	É direcionada aos artistas não se focando tanto na gestão da *venue*	Apenas destinado a artistas	Poucas *features*	Um pouco confuso Apenas destinado a promotores de eventos	Muitas das *features* são pagas Muitas funcionalidades, pouco foco	Pouco conteúdo	Em fase inicial de desenvolvimento

Tabela 3.1: Análise comparativa da concorrência

Através da tabela 3.1 podemos concluir que, de todas as plataformas analisadas, as plataformas *Prism.fm* e *Muzeek* apresentam-se como as maiores concorrentes ao Sala-Z. Apesar de muitas das funcionalidades serem semelhantes numa fase inicial do desenvolvimento do Sala-Z, o objetivo da plataforma será, no futuro, não só apoiar os gerentes das *venues* a gerirem o seu negócio mas, também, auxiliar artistas a encontrar *venues* para atuar, assim como, gerir o seu negócio e a sua banda.

Em adição, a plataforma apenas terá um concorrente direto em Portugal que não apresenta grandes semelhanças ao Sala-Z v2.0, pelo que, poderá ter mais impacto a nível nacional.

3.4 Tecnologias

Nesta secção é apresentado o estudo realizado sobre algumas das tecnologias mais relevantes no cenário de desenvolvimento de *software* analisando a sua taxa de uso, assim como, as diferenças entre ambas. Estas informações irão fundamentar a decisão tomada pelo estagiário e pela equipa da Grama na escolha de tecnologias usadas no desenvolvimento da plataforma Sala-Z v2.0.

3.4.1 Tecnologias de *Frontend*

Dentro do cenário de *frontend* existem inúmeras escolhas possíveis. Das possibilidades foram extraídas três: ***Angular***, ***React.js*** e ***Vue.js*** por apresentarem uma grande relevância a nível da comunidade de desenvolvimento de *frontend* de *web applications*, assim como, por serem ferramentas utilizadas no dia-a-dia da equipa de desenvolvimento da empresa Grama.

Angular

Angular é uma *framework* muito bem estabelecida dentro da comunidade de programadores por todo o mundo. A mesma adota a linguagem de *TypeScript* e é bastante completa, o que a faz também ser pesada e de alguma dificuldade de aprendizagem [17][18][19].

Em adição, é uma das *frameworks* mais usadas dentro da empresa Grama, sendo que, o facto do alto suporte oferecido pela equipa pode afetar a decisão de escolha da mesma.

React.js

React é uma ferramenta desenvolvida pelo *Facebook* e, em paralelo com *Vue*, adota a linguagem de *JavaScript*. É uma das tecnologias mais populares dentro da comunidade pois insere o seu foco na simplicidade e na facilidade de criação dos *UIs*. Porém, *React* não é uma *framework*, o que implica que seja necessária a importação de bibliotecas para obter algumas funcionalidades o que torna esta ferramenta leve inicialmente mas, à medida que novas bibliotecas são adicionadas, pode-se tornar mais complexa, assim como, pesada [17][18][19].

Vue.js

Por fim, *Vue.js* é uma *framework* de *JavaScript* que tenta unir o melhor das duas tecnologias referenciadas anteriormente pois corre sobre o *template* de *Angular* e adota políticas do *React* o que a torna incrivelmente leve. Por essa razão, tanto a comunidade de *Angular* como de *React* têm interesse nesta ferramenta e, assim, está numa ascensão de popularidade enorme [18].

3.4.2 *Frontend* - Comparações

Após o conhecer as diferentes plataformas foi feita uma comparação entre as três para, assim, tomar a decisão de qual das tecnologias usar no desenvolvimento do *frontend* da plataforma.

	Angular	React.js	Vue.js
Ano de lançamento	2010	2013	2014
Linguagem	TypeScript	JavaScript	JavaScript/TypeScript
Vantagens	Plataforma muito completa	Leve	Leve
	Grande suporte da comunidade	Fácil aprendizagem	Fácil aprendizagem
Desvantagens	Aprendizagem difícil	Necessita de apoio de bibliotecas	Menor suporte da comunidade em comparação com as restantes tecnologias
	Ferramenta pesada	Muitas dependências podem criar complexidade	
Ideal para	Plataformas de grande escala com muitas *features*	Plataformas com muitas componentes diferentes	Plataformas com muitas componentes diferentes
Popularidade	Médio-alto	Alto	Alto
Apoio da empresa	Alto	Médio	Médio-alto

Tabela 3.2: Comparação de tecnologias de *frontend*

Relevância na comunidade

Para compreender a relevância destas três tecnologias dentro da comunidade, foi analisado o número de estrelas de repositórios do *Github* que utilizam as mesmas. Assim, com o auxilio do gráfico 3.2 disponibilizado pelo blog da *Codeinwp* [5], podemos observar muito nitidamente três pontos distintos:

- Podemos visualizar que a *framework Angular* é a que apresenta menos estrelas das três. Isto é compreensível visto que é uma *framework* difícil de aprender, ou seja, é muito dependente da comunidade fiel à mesma.
- Podemos também reparar que *React* e *Vue* têm vindo a crescer imenso desde 2016 e dominaram completamente sobre a antiga *framework Angular*.
- Por fim, podemos ver que desde 2019, *Vue* têm se apresentado como a *framework* mais apelativa à comunidade de desenvolvimento, ultrapassando *React* a meio de 2018.

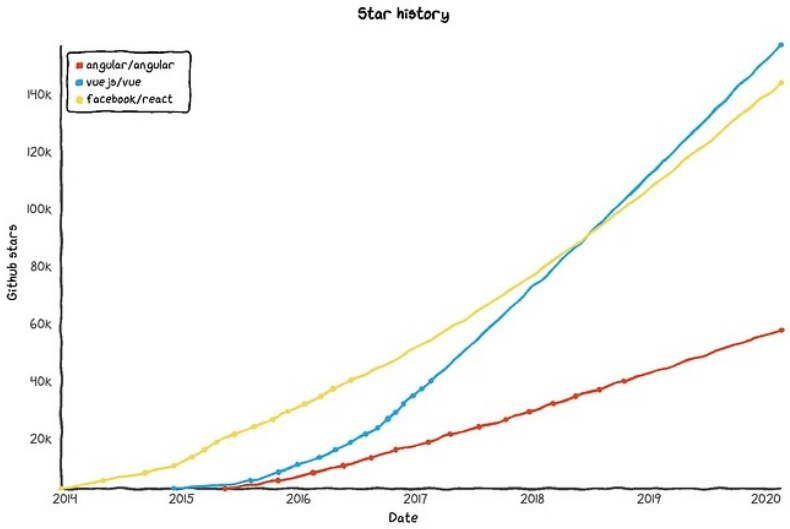


Figura 3.2: Popularidade de *frameworks* de *frontend* [5]

3.4.3 Tecnologias de *Backend*

Como no cenário de *frontend*, existem várias tecnologias de desenvolvimento de *backend* que poderão ser usadas no projeto. Na presente secção serão abordadas algumas das tecnologias mais relevantes na comunidade de desenvolvimento de *software*, assim como, dentro da empresa Grama.

Por fim, dada a oportunidade do aluno trabalhar com o serviço da AWS *Lambda* [20], será feita uma análise das diferenças entre uma arquitetura com servidor e uma arquitetura *serverless*, tomando as devidas conclusões.

Spring Boot

Spring é uma plataforma *open-source* de *Java* que se destaca por ter recursos especiais como injecção de dependências, assim como, vários módulos adicionais interessantes (Spring JDBC, MVC, Security, AOP, ORM e Test) que reduzem substancialmente o tempo de implementação [21][22].

Spring Boot é uma versão já inicializada da plataforma *Spring* que torna o começo da produção do projeto mais rápido, minimizando os passos mais confusos comuns nesta fase.

Express.js

Express.js é uma *framework open-source* que adota a linguagem de *JavaScript* e corre sobre *Node.js*. A principal diferença entre esta tecnologia e outras é que, nesta, a execução de eventos é realizada em *single-thread*, ou seja, torna a *framework* rápida e com baixo consumo de memória [21].

Quarkus

Quarkus é uma *framework full stack* nativa em *Kubernetes*[1] muito recente (2019) que, semelhantemente a *Spring*, é corrida em *java* mas oferece melhorias a nível de *start up* e *memory footprint*. Por sua vez, *Quarkus* otimiza a sua linguagem para *containers* o que torna esta plataforma mais eficaz em cenários *Kubernetes*, *cloud*, ou mesmo, *serverless* [24].

[1] "*é uma plataforma de orquestração de containers open source que automatiza muitos dos processos manuais envolvidos no deployment, gestão e escalabilidade de aplicações corridas em containers.*"[23]

3.4.4 Backend - Comparações

	Spring	Express.js	Quarkus
Ano de lançamento	2002	2010	2018
Linguagem	Java	JavaScript	Java
Vantagens [25]	Comunidade de java bem estabelecida	Rápido e leve	Pouca utilização de memória e primeira resposta muito rápida
	Suporte *multi-threading*	*Single-threading* exige menos memória	Eficiente para ambientes *cloud*, *containers* e *serverless*
Desvantagens [25]	Uso de memória pode ser grande	Não suporta *multi-threading*	Muito novo (pouco suporte da comunidade)
	Pode incluir dependências que não são usadas	Não é ideal para computação pesada	-
Desempenho [25][26]	Alto poder de computação	Muito rápido	De forma geral, melhor que Spring
	Mais lento que Express.js	Baixo poder de computação	
Popularidade[27]	Baixo	Médio	Médio
Apoio da empresa	Alto	Baixo	Médio

Tabela 3.3: Comparação de tecnologias de backend

3.4.5 Arquitetura com Servidor *vs Serverless*

Tendo em consideração que a empresa tem acesso ao serviço prestado pela *AWS*, *Lamba*, e que a primeira versão da plataforma Sala-Z foi desenvolvida com uma arquitetura *serverless*, foi dada ao estagiário a oportunidade de desenvolver o novo sistema com uma arquitetura sem servidor. Para tomar uma decisão foi necessário estudar as vantagens e desvantagens de construir um sistema com estas características.

Em primeiro lugar, é necessário entender o que é e como funciona uma computação sem servidor. Como dito anteriormente, o serviço *AWS Lambda* permite a execução de código sem ser ser necessário gerar um servidor e apenas é cobrado o tempo de computação consumido. O que ganhamos com isso?

Vantagens e desvantagens de uma solução *Serverless*

A "revolução"das arquiteturas *serverless* já é falada desde há muito tempo pelas vantagens que apresenta em relação a abordagens mais convencionais, porém, apesar de já existir esta ideia desde 2006 e com o sucesso do serviço da *Amazon, Lambda*, em 2014, ainda não é uma arquitetura bem estabelecida ou usada pelas empresas de *software* por ainda apresentar vários problemas (Referências [28], [29], [30] e [31]) .

Vantagens

Em primeiro lugar, **custo**. Obviamente que, ao pagar apenas quando é efectuado um pedido, o custo diminui quanto menor for a utilização, ou seja, compensa para sistemas em que a carga de pedidos não

é constantemente elevada.

De seguida, **segurança** pois *cloud servers* são conhecidos por serem altamente seguros por estarem a cargo de empresas com alta reputação, como a *Amazon*.

Em adição, os *developers* poupam imenso trabalho ao não terem que se preocupar com o Sistema Operativo (SO), assim como, não existe a preocupação de construir uma aplicação limitada pelo SO, apenas é necessário construir o código com a lógica de negócio e fazer *upload* para a *framework serverless*.

Por fim, **escalabilidade**. Como os recursos oferecidos pelas *frameworks serverless* podem ser atribuídos dinamicamente, estão preparados para picos de pedidos.

Problemas

Apesar das vantagens oferecidas por este tipo de arquitetura, existem ainda vários problemas com a abordagem *serverless*. Em primeiro lugar, o facto de existirem **limitações a nível da linguagem** usada para criar o código faz com que esta abordagem não seja flexível. Apesar da *AWS Lambda* disponibilizar *wrappers* que possibilitam correr funções em linguagens que não são suportadas, isto tem um custo de *performance*.

De seguida, como ainda é uma abordagem "nova", o facto de **não haver uma convenção entre *frameworks*** distintas, cria alguns problemas como o facto do *developer* ficar preso a uma empresa (***Vendor Lock***), assim como, torna-se mais difícil migrar aplicações para *serverless*.

A nível de ***performance***, é difícil avaliar e comparar este tipo de abordagem com abordagens mais convencionais (*in-house server* por exemplo) pois, por vezes, esta informação é "escondida"pelas empresas responsáveis (neste caso a AWS). Apesar destas afirmarem que a nível de performance, uma arquitetura *serverless* é melhor, existem estudos [31] que afirmam o contrário.

Por fim, é extremamente complicado correr um aplicação inteira num ambiente *serverless*. Geralmente, este serviço é usado para complementar uma abordagem com servidor em que apenas algumas funções estão alocadas no serviço *Lambda*.

3.4.6 Decisões tomadas

Após analisar cada uma das opções possíveis tanto na área de *frontend* como de *backend* e após debater as mesmas com os respectivos orientadores de estágio, foram tomadas as respectivas decisões.

Foi tomada a decisão de implementar o *frontend* com ***Vue*** por ser uma tecnologia fácil de aprender, por ser usada dentro da empresa, pelo que, existe um suporte forte oferecido ao estagiário e por ser cada vez mais popular dentro da comunidade de desenvolvimento. Por outro lado, relativamente ao *backend*, apesar da empresa ter oferecido a hipótese de implementar o mesmo num ambiente *serverless*, foi descartada esta hipótese face aos problemas que a mesma ainda possui, pelo que, foi decidido implementar o servidor com a *framework* ***Quarkus*** por ser uma evolução da ferramenta *Spring*, por existir uma vontade, tanto do lado do estagiário como da empresa, de explorar a mesma e por dar resposta a todos os requisitos da plataforma.

3.5 Riscos

Antes do desenvolvimento da plataforma foi feita uma análise dos riscos que estão envolvidos no mesmo para que, no futuro, seja possível prever e tentar evitar que os mesmos ocorram ou tenham um alto impacto no resultado final do produto.

Cada risco será composto pela sua **descrição**, **estratégia de mitigação**, **impacto** e **probabilidade** e serão repartidos em duas secções. A primeira secção será referente ao desenvolvimento do produto e problemas que o aluno poderá enfrentar no mesmo. Na segunda secção da análise de riscos serão

abordados os riscos que a plataforma em si poderá enfrentar.

3.5.1 Riscos de desenvolvimento

1. Tecnologias - O facto das tecnologias usadas no desenvolvimento do produto serem desconhecidas ao aluno será, sem dúvida, uma factor que pode influenciar o tempo que o mesmo demorará a realizar as tarefas e tornar o processo de desenvolvimento mais demorado e imprevisível numa fase inicial. Em adição, as tecnologias a serem utilizadas são relativamente recentes (ver Capítulo 3.4) o que poderá levar a que exista alguma falta de suporte por parte da comunidade.

Estratégia de mitigação: Para contornar o facto do aluno demorar algum tempo a familiarizar-se com as ferramentas, o mesmo começou este processo no 1º semestre, começando a desenvolver um protótipo do produto. Em adição, o orientador tem uma vasta experiência com as ferramentas (ou ferramentas semelhantes) que irão ser utilizadas, pelo que, poderá ajudar em qualquer ponto do desenvolvimento do produto.

Impacto: Alto

Probabilidade: 30% - 50%

2. Falta de experiência - Face à falta de experiência do aluno com as ferramentas a serem utilizadas, o mesmo poderá ter alguma dificuldade em terminar todas as componentes do produto até ao final do ano lectivo.

Estratégias de mitigação: O planeamento efectuado para o segundo semestre deverá ser cumprido o melhor possível tentando evitar atrasos. Em adição, os requisitos deverão ser classificados pelo método de *MoSCoW* (*Must, Should, Could, Would*) para garantir que todos os requisitos críticos são implementados.

Impacto: Alto

Probabilidade: 30% - 50%

3. Pandemia - Com a evolução da pandemia durante o 1º semestre, tornou-se impossível o trabalho presencial tendo a empresa optado por trabalhar remotamente. Com esta mudança, o aluno não terá o mesmo suporte que teria em condições normais.

Estratégia de mitigação: A única solução será haver um acompanhamento remoto ao progresso feito pelo aluno, assim como, o mesmo comunicar sempre os seus problemas ao orientador de estágio através dos diversos canais de comunicação.

Impacto: Muito baixo

Probabilidade: 70% - 100%

3.5.2 Riscos da plataforma

4. Agentes não aderirem - A plataforma é completamente dependente da participação de gerentes de salas e de promotores de eventos para poder florescer.

Estratégia de mitigação: Durante o processo de análise de requisitos e durante o desenvolvimento do produto, a equipa de desenvolvimento terá contacto com possíveis clientes. Após o produto estar completo, esses mesmos possíveis clientes terão acesso à plataforma de forma a ter um conjunto de clientes já a utilizar a plataforma.

Impacto: Alto

Probabilidade: 10% - 30%

5. Pandemia - Como a pandemia afeta fortemente a área cultural, poderá haver algum atrito em os agentes pagarem pela plataforma, assim como, o facto da pandemia reduzir a quantidade de concertos terá um impacto na utilização do Sala-Z.

Estratégia de mitigação Numa perspectiva positiva, quando a plataforma for lançada para o mercado, a pandemia poderá já ter acabado ou aliviado de tal forma que seja possível voltar a haver

mais concertos, porém, é bastante provável que os problemas financeiros por parte dos agentes continuarão existir, pelo que, a plataforma poderá estar disponível de forma gratuita numa fase inicial e ser paga através de pacotes extra com funcionalidades especificas.

Impacto: Moderado

Probabilidade: 70% - 100%

3.5.3 Matriz de riscos

Após serem analisados os riscos da plataforma em junção com as estratégia de mitigação, impacto e probabilidade correspondente, foi elaborada uma matriz de riscos (Fig. 3.3) de forma a facilitar a visualização dos mesmos .

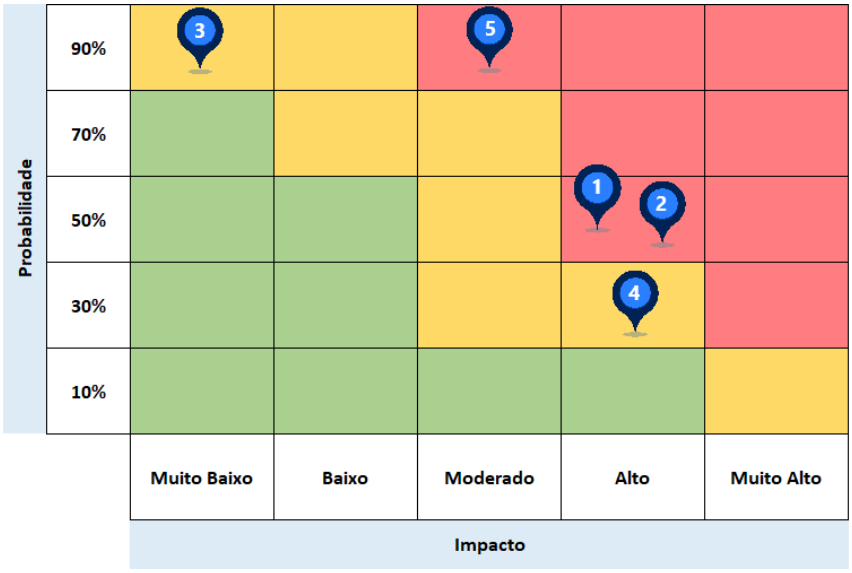


Figura 3.3: Matriz de riscos [6]

Capítulo 4

Requisitos

Para planear a arquitetura de *software* e entender detalhadamente o que será implementando no decorrer do segundo semestre, foram levantados os requisitos da plataforma e escritos em forma de ***user stories***.

Salienta-se que, devido à metodologia usada, pode haver uma motivação (através de testes de usabilidade ou outros) para haver pequenas alterações nos requisitos no decorrer do segundo semestre.

Assim, o presente capítulo apresenta todo o processo precedente ao levantamento dos requisitos, assim como, expõe todas as *user stories* escritas pela equipa.

4.1 Entrevistas

Para complementar o estudo feito no estado da arte, a equipa de desenvolvimento realizou duas entrevistas a dois possíveis clientes que possuem uma vasta experiência na sua área de forma a perceber a necessidade de cada um e, assim, complementar o Sala-Z com funcionalidades que não foram pensadas previamente.

Em primeiro lugar foi entrevistado o gerente do "Salão Brasil"em Coimbra.

Da entrevista remota de aproximadamente uma hora, a equipa conseguiu entender alguns problemas que existem no planeamento de um evento, nomeadamente, factores externos como a estadia dos artistas (número de quartos, quantas pessoas, onde vão ficar, entre outros) e as refeições (levantam-se questões como: "existem pessoas vegetarianas?", "quem vai estar na cozinha?", ou mesmo, "quantas pessoas vão comer?"). Assim, apesar de serem factores externos ao evento musical em si, poderão ser, no futuro, adições que a ferramenta poderá incluir.

Em adição aos problemas referidos anteriormente, também foi possível entender que, atualmente, estes locais não usam nenhuma ferramenta para gerir o seu negócio, pelo que, preferem meios analógicos ou apenas comunicar por *emails* para gerir a equipa ou para negociar algum concerto. Em adição, o gerente da *venue* referiu, também, que já usaram ferramentas como o *Trello* e *Slack* mas abandonaram muito rapidamente, porém, expõe a necessidade e grande interesse em ter ferramenta que possa, de alguma forma, agilizar todo o processo de gestão e ver o desempenho da casa ao longo do tempo.

Em segundo lugar entrevistámos o coleccionador de discos e lojista, Rui Ferreira, que, para além de ter uma enorme experiência na área de promoção de espetáculos e edição de discos, já foi agente de diversos artistas.

Durante a entrevista presencial, o entrevistado referiu vários problemas que sentiu e sente no seu percurso como agente e promotor de eventos, entre os quais:

- Muitos artistas tentam chegar a ele por *email*, porém, é muito complicado para ele conseguir ler e responder a todos devido ao "*spam*"constante de mensagens.

- Agendar uma data é sempre um problema por ser complicado arranjar um dia e hora que agrade tanto o artista/banda como a *venue*.
- Muitas vezes burocracias atrasam todo o processo, como por exemplo, a venda de bilhetes só poderá começar depois de uma aprovação de uma entidade superior (ex. Câmara Municipal).

Em adição foram dadas várias dicas importantes à equipa da Grama, como por exemplo, o facto de estatísticas de eventos poderem levar a conclusões erradas sobre o mesmo, por exemplo, um evento pode não ter tanta audiência pelo simples facto de estar a chover e as pessoas não quererem sair de casa, porém, o gerente pode assumir no futuro que o evento não foi bem sucedido por causa do artista.

Em conclusão, após realizadas ambas as entrevistas, a equipa ficou com uma visão mais ampla do problema em si, percebendo alguns detalhes que, anteriormente, não tinha sequer posto em causa.

Assim, estas informações irão complementar o conceito já existente para a plataforma para, deste modo, levantar os requisitos necessários para obter um produto completo e coerente.

4.2 User Stories

De modo a especificar os requisitos da plataforma, a empresa optou por descrever a aplicação de acordo com as necessidades do utilizador, ou seja, através de '*User Stories*'. Nestas, uma funcionalidade é abordada com um formato fixo em que o objetivo será definir o **tipo de utilizador** que pode cumprir um **objetivo** que tem por uma **razão**.

Assim, cada *user story* é estruturada da seguinte forma:

"Como um <tipo de utilizador> posso <objetivo> para que <razão>."

Devido à grande quantidade de *user stories*, estas foram expostas por completo na Apêndice A. Dito isto, o módulo da autenticação exibido de seguida é um fração do total das *user stories* e serve apenas de exemplo para o presente capítulo.

ES-1: **Como um** utilizador de um espaço **quero** garantir que o Sala-Z apenas é utilizado por utilizadores autenticados **para** assegurar que todas as funcionalidades da plataforma apenas são acedidas por utilizadores autorizados.

US-1: *Login*

(a) **Como um** utilizador de um espaço **quero** entrar no Sala-Z **para** usar as funcionalidades da plataforma.

US-2: Terminar Sessão

(a) **Como um** utilizador de um espaço **quero** terminar a sessão **para** outros utilizadores que têm acesso ao meu aparelho não possam utilizar a plataforma em meu nome.

US-3: Recuperação da *password*

(a) **Como um** utilizador de um espaço **quero** pedir uma recuperação de *password* **para** poder aceder à minha conta no caso de me esquecer da *password*.

US-4: Primeiro Login

(a) **Como um** utilizador de um espaço **quero** completar o meu primeiro *login* **para** poder aceitar um convite de acesso ao Sala-Z e, consequentemente, mudar a minha *password* definida pelo sistema.

4.3 Requisitos funcionais

Tendo por base as *user stories* definidas ao longo do Capítulo 4.2, será possível delinear todos os requisitos que a plataforma deverá ter para ser considerada uma *Minimum Viable Product* (MVP). Em adição, será feita uma avaliação de cada requisito consoante a sua prioridade seguindo o método de *Moscow*, ou seja, cada requisito poderá apresentar um dos seguintes quatro tipos de prioridade:

- ***Must Have*** **(M)** - Corresponde à prioridade máxima. Este requisito terá que estar presente no produto final para o mesmo ser considerado MVP.
- ***Should Have*** **(S)** - Corresponde a uma prioridade média, ou seja, o requisito é importante, porém, caso não seja implementado o produto continua válido.
- ***Could Have*** **(C)** - Corresponde a uma prioridade baixa. Este requisito é desejável mas apenas será implementado caso haja tempo extra disponível para tal.
- ***Won't Have*** **(W)** - Corresponde à prioridade mais baixa. Este requisito não será implementado para já mas poderá ser implementado no futuro.

Assim, todos os requisitos da plataforma estão representados nas tabelas 4.1, 4.2 e 4.3 que representam os requisitos referentes ao módulo da autenticação, módulo da gestão de *venues* e módulo do *backoffice*, respetivamente.

Tema	Requisito	Prioridade
Autenticação e definições pessoais	Login	(M)
	Primeiro Login	(M)
	Logout	(M)
	Recuperar password	(M)
	Mudar password	(M)
	Registar	(C)
	Mudar dados pessoais	(M)

Tabela 4.1: Requisitos comuns ao utilizadores

Tema	Requisito	Prioridade
Administração	Procurar membros da venue	(S)
	Listar membros da venue	(M)
	Filtrar membros da venue	(C)
	Ordenar membros da venue	(C)
	Remover membros da venue	(M)
	Convidar utilizador	(M)
	Reenviar convite	(S)
Definições	Listar venues	(M)

	Pedir ajuda	(C)
Definições das venues	Ver venue	(M)
	Editar venue	(M)
	Adicionar recurso à venue	(C)
	Mudar recurso da venue	(C)
	Listar recursos da venue	(C)
	Eliminar recurso da venue	(C)
Gestão de artistas	Ver lista de artistas	(S)
	Marcar artista como favorito	(C)
	Editar artista	(S)
	Filtrar lista de artistas	(S)
	Ordenar lista de artistas	(S)
	Procurar na lista de artistas	(S)
	Exportar listagem de artistas	(C)
	Adicionar artista à lista de artistas	(S)
	Ver um artista da lista de artistas	(S)
	Editar um artista da lista de artistas	(S)
	Remover um artista da lista de artistas	(S)
Gestão de eventos	Adicionar novo evento	(M)
	Editar evento	(M)
	Ver evento	(M)
	Apagar evento	(M)
	Ver eventos no calendário	(M)
	Ver lista de eventos	(M)
	Filtrar lista de eventos	(S)
	Ordenar lista de eventos	(S)
	Procurar na lista de eventos	(M)
	Exportar listagem de eventos	(S)
	Adicionar tarefa ao evento	(S)
	Ver tarefa	(S)
	Eliminar tarefa	(S)
	Editar tarefa	(S)
	Adicionar tarefa a lista de tarefas	(S)
	Listar tarefas	(S)

	Filtrar lista de tarefas	(S)
	Ordenar lista de tarefas	(S)
	Pesquisar na lista de tarefas	(S)
	Exportar lista de tarefas	(S)
	Adicionar recurso ao evento	(C)
	Eliminar recurso do evento	(C)
	Editar recurso do evento	(C)
	Sincronizar calendários	(C)
Gestão de recursos	Listar recursos da venue	(C)
	Ver utilização de recursos no calendário	(C)
	Listar recursos	(C)
	Filtrar lista de recursos	(C)
	Ordenar lista de recursos	(C)
	Pesquisar recurso na lista de recursos	(C)
	Exportar lista de recursos	(C)
Dashboard	Ver evento com mais receita (com filtros)	(M)
	Ver evento com mais entradas	(M)
	Ver artista que gerou mais receita (com filtros)	(M)
	Ver artista com maior número de eventos	(M)
	Receita por mês	(M)
	Receita anual por tipo	(M)
	Número de eventos por mês	(M)
	Ver eventos num calendário	(M)
	Ver tipo de música que gerou mais receita (com filtros)	(C)
	Ver tipo de música com mais eventos associados	(C)

Tabela 4.2: Requisitos referentes à gestão de venues

Tema	**Requisito**	**Prioridade**
Backoffice	Criar venue	(M)
	Associar utilizador a venue	(M)
	Reenviar convite	(M)
	Apagar utilizador	(M)
	Listar venues	(M)

Listar utilizadores de venues	(M)
Editar venue	(M)

Tabela 4.3: Requisitos referentes ao backoffice

4.4 Diagrama de casos de uso

Depois das *user stories* estarem completas e os requisitos funcionais descritos, podemos extrair os casos de uso da presente plataforma. Assim, o Sala-Z terá dois utilizadores principais, o ***Venue User*** que é o utilizador que irá utilizar a plataforma para gerir a(s) sua(s) sala(s) de espetáculo e o ***Sala-Z Administrator*** que, acedendo ao *backoffice*, será responsável por gerir toda a plataforma, os seus utilizadores e *venues* existentes.

4.4.1 Autenticação e informações pessoais

Na figura 4.1 podemos observar os agentes anteriormente referidos que, após efetuarem o *login* poderão usar as funcionalidades da plataforma que têm acesso.

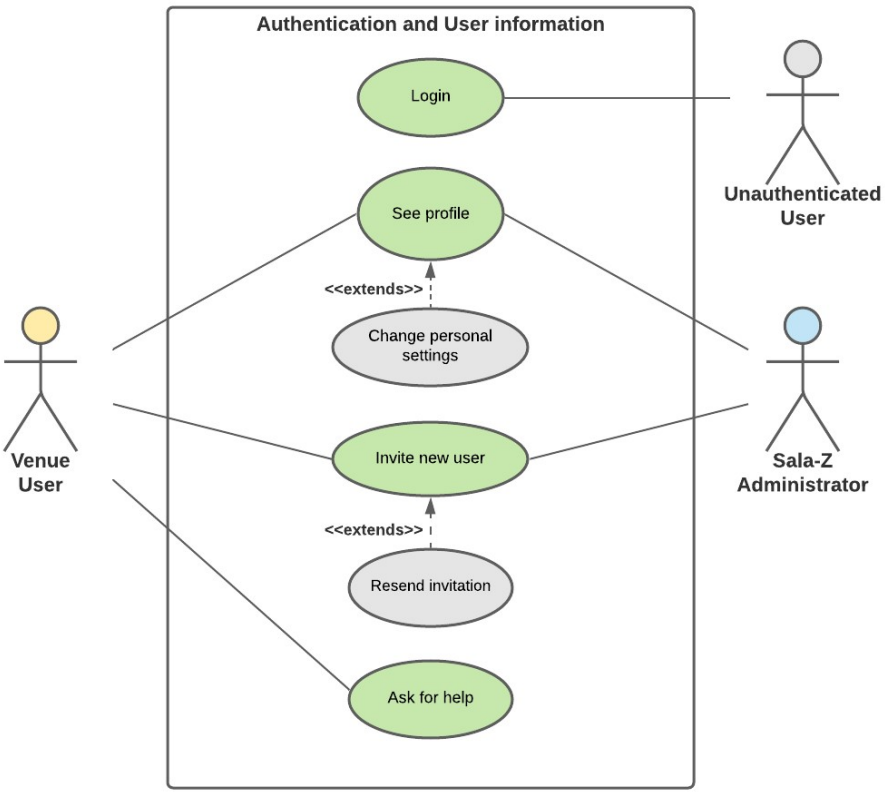


Figura 4.1: Casos de uso de autenticação e informações do utilizador

4.4.2 *Backoffice*

Já na figura 4.2 podemos ver as funcionalidades do *backoffice*, ou seja, as funcionalidades que apenas o *Sala-Z Administrator* terá acesso.

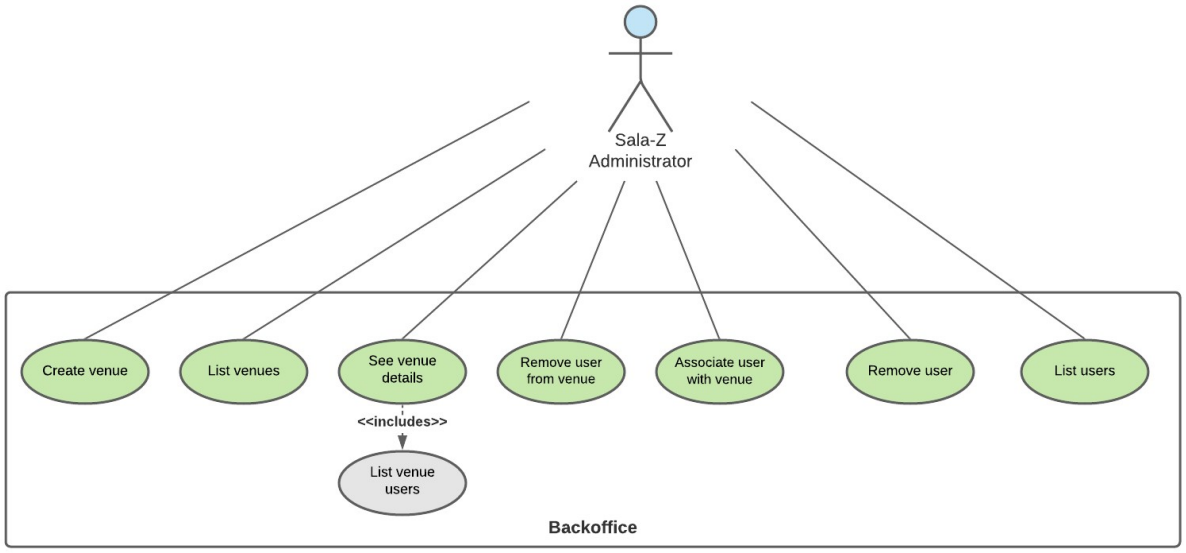


Figura 4.2: Casos de uso *backoffice*

4.4.3 *Venue user*

Por fim, podemos analisar no conjunto de imagens seguintes (Fig. 4.3 a Fig. 4.7) todas as funcionalidades que um utilizador comum (*venue user*) terá quando acede à plataforma.

Em primeiro lugar, o utilizador poderá gerir cada uma das suas *venues* associadas (Fig. 4.3).

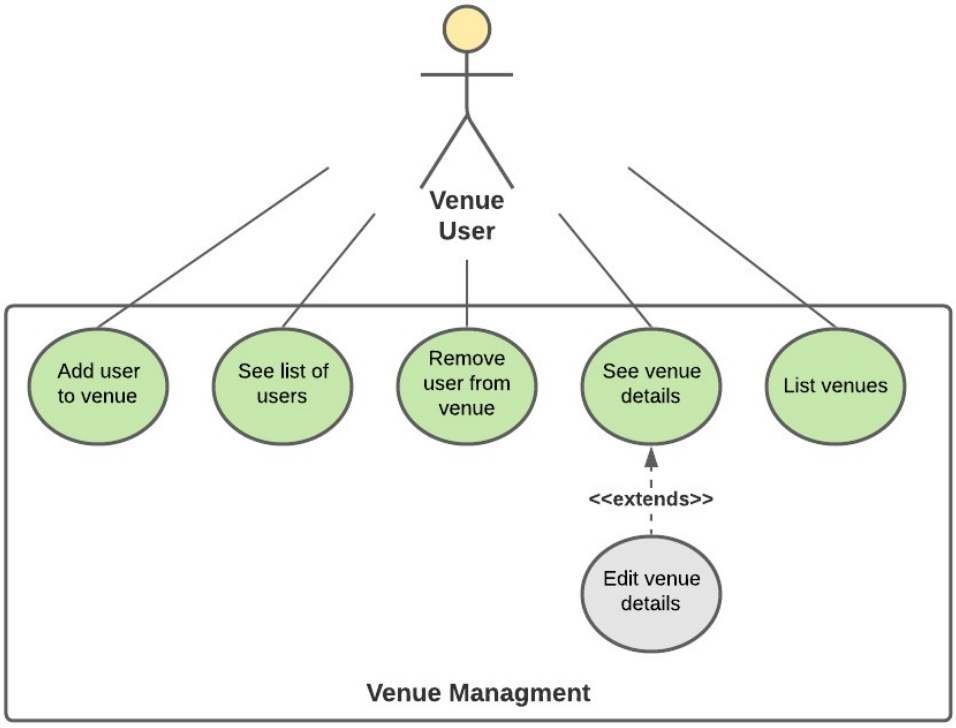


Figura 4.3: Caso de uso - *Venue managment*

Tendo pelo menos uma *venue* associada, o utilizador poderá gerir quaisquer eventos de quaisquer *venues* (Fig. 4.4).

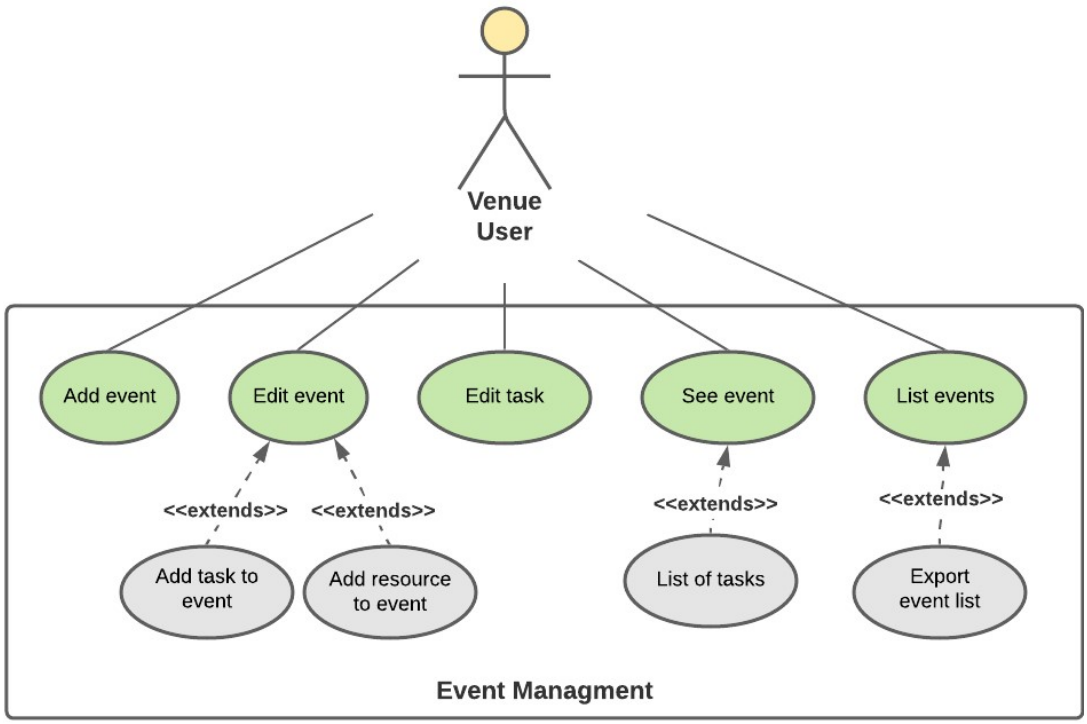


Figura 4.4: Caso de uso - *Event managment*

O utilizador poderá, também, visualizar estatísticas relacionadas com os seus estabelecimentos e/ou eventos a partir de uma *dashboard* (Fig. 4.5).

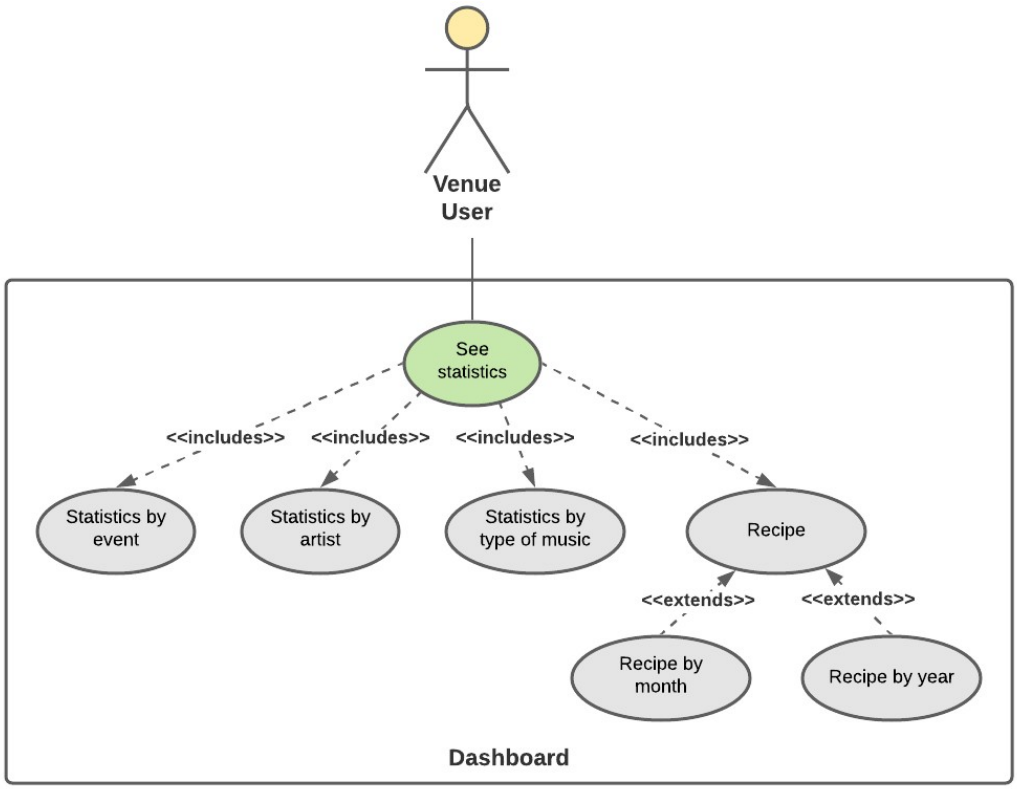


Figura 4.5: Caso de uso - *Dashboard*

Por fim, o utilizador poderá consultar ou gerir outros componentes do negócio como, recursos (Fig. 4.6) ou artistas (Fig. 4.7).

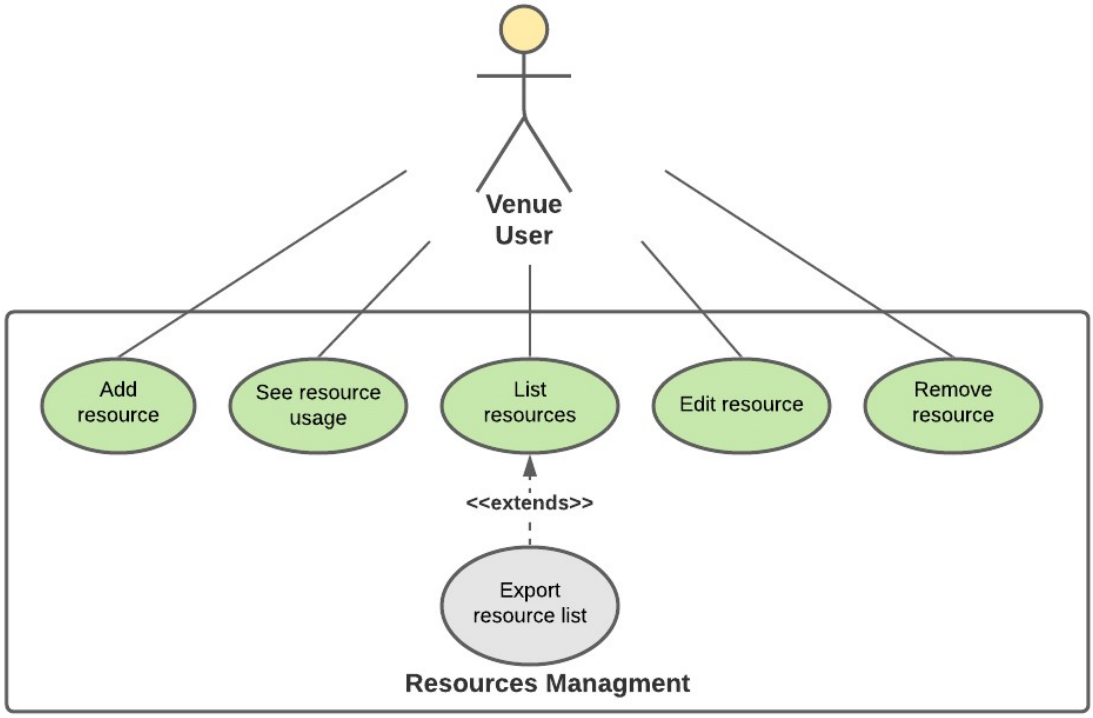


Figura 4.6: Caso de uso - *Resource managment*

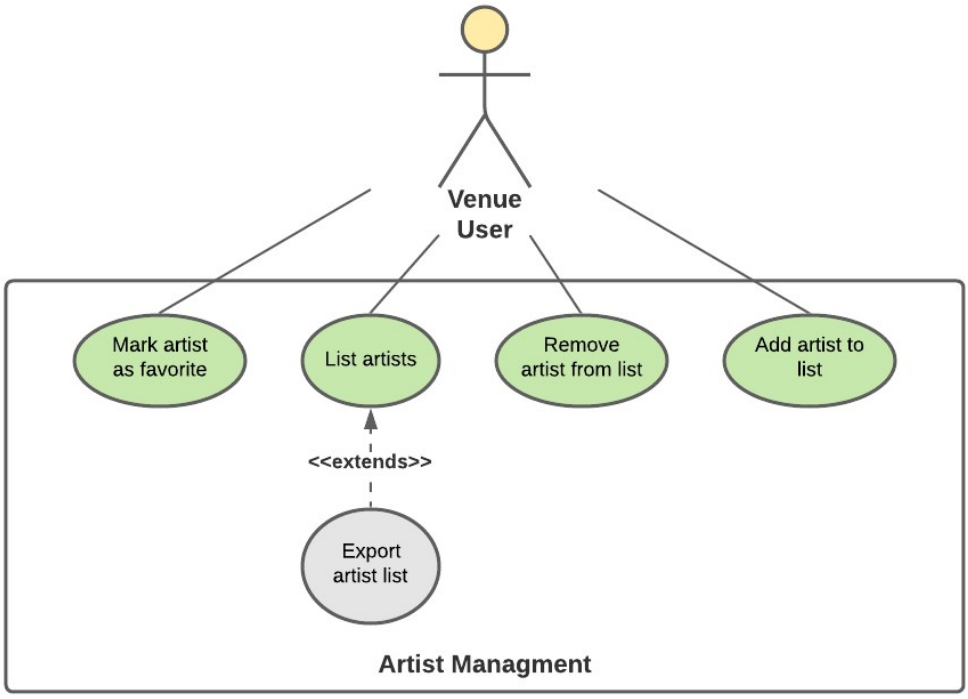


Figura 4.7: Caso de uso - *Artist managment*

4.5 Requisitos não funcionais

Para terminar o presente capítulo, foi também feita uma abordagem aos atributos de qualidade presentes na plataforma, ou seja, foram definidos os requisitos não funcionais que a plataforma terá que conter para ser considerada válida.

4.5.1 Disponibilidade

Disponibilidade é a capacidade do sistema estar pronto para executar uma tarefa num dado momento sob as condições específicas. Na presente plataforma, o produto deverá conter uma disponibilidade de **99.99%** pois é o oferecido pelo serviço *Amazon Elastic Compute Cloud* (EC2)[32] da *Amazon Web Services* (AWS).

Possível cenário: O utilizador, ao aceder ao Sala-Z, terá uma probabilidade de 99.99% de encontrar o sistema disponível.

4.5.2 Escalabilidade

Escalabilidade é a capacidade do sistema ser capaz de se adaptar e manter uma boa *performance* quando o número de pedidos feitos ao mesmo é elevado. Mais uma vez, como a plataforma vai estar alocada na AWS, o serviço EC2 oferece uma **escalabilidade horizontal**, ou seja, existe a possibilidade de adicionar poder computacional em alturas de maior *stress*.

Possível cenário: A plataforma, num cenário de grande *stress*, deverá manter-se disponível e responder a qualquer pedido num máximo de 3 segundos.

4.5.3 Segurança

Segurança é a capacidade do sistema de proteger dados e resistir a acessos não autorizados. Assim, a plataforma deverá manter a **confidencialidade** e **integridade** dos dados do utilizador (segundo o Regulamento Geral sobre a Proteção de Dados (RGPD)), **autenticar** ambas as entidades (através do serviço *Cognito* [10] da AWS[10]) e **negar a repudiação**.

Possível cenário: O utilizador autenticado na plataforma tem a garantia que os seus dados são confidenciais.

4.5.4 Usabilidade

Usabilidade é a facilidade de aprendizagem e uso por parte do utilizador quando apresentado com a plataforma Sala-Z. Com o conhecimento adquirido no Estado da Arte (Cap.3) e das entrevistas realizadas a possíveis clientes, a plataforma deverá ser desenhada de modo a cumprir este requisito, sendo que, no futuro, o mesmo será ou não aprovado pelos possíveis clientes abordados nas entrevistas.

Possível cenário: Durante o funcionamento normal do sistema, o utilizador deverá conseguir encontrar um objetivo facilmente com num máximo de quatro cliques.

Capítulo 5

Arquitetura de Software

No presente capítulo é exposta a arquitetura do *software* construída de acordo com as *user stories* levantadas no Capítulo 4.2. A arquitetura seguirá o modelo C4, por ser um modelo já usado anteriormente pelo aluno e por ser simples de compreender e construir.

Como referido anteriormente, a empresa está subscrita e usa os serviços oferecidos pela AWS, pelo que, em adição à arquitetura de *software*, o presente capitulo também fará uma breve introdução aos serviços que estarão presentes no desenho da aplicação.

5.1 Amazon Web Services (AWS)

A AWS é um conjunto de mais de 175 serviços oferecidos pela empresa da *Amazon* ligados a armazenamento, rede, análise, serviços de aplicação gestão, entre outros. De todos os serviços oferecidos pela *Amazon*, será feita uma a análise de oito serviços utilizados pela empresa Grama que estarão presentes na arquitetura do Sala-Z.

São estes:

CloudFront - "*O CloudFront agiliza a distribuição do conteúdo web estático e dinâmico, como arquivos .html, .css, .php e ficheiros estáticos. Quando os utilizadores solicitam algum conteúdo, o CloudFront fornece-o através de uma rede mundial de pontos de presença que oferecem baixa latência e alto desempenho*". [33]

S3 - "*O Amazon Simple Storage Service (S3) oferece armazenamento para a Internet. É possível utilizar o Amazon S3 para armazenar e recuperar qualquer volume de dados, a qualquer momento, de qualquer lugar na web*". [34]

API Gateway - "*O API Gateway permite criar e implementar APIs REST e WebSocket a qualquer escala. É possível criar APIs robustas, seguras e dimensionáveis que acedem à AWS ou outros serviços web, bem como dados armazenados na Cloud AWS*". [35]

EC2 - "*O EC2 é um serviço web que fornece capacidade de computação escalável – literalmente, servidores dos datacenters da Amazon – para criar e hospedar sistemas de software*". [32]

ELB - "*O Elastic Load Balancing (ELB) distribui automaticamente o tráfego de entrada do aplicativo em vários destinos, como instâncias do EC2. Monitoriza a integridade dos destinos registados e roteia o tráfego apenas para os destinos íntegros. O Elastic Load Balancing oferece suporte a três tipos de load balancers: Application Load Balancers, Network Load Balancers e Classic Load Balancers*". [36]

RDS - "*O Amazon Relational Database Service (RDS) é um serviço web que facilita a configuração, a operação e escalabilidade de uma base de dados relacional na cloud. Fornece a capacidade económica e redimensionável para uma base de dados relacional padrão do setor e gere tarefas comuns de administração de base de dados*". [37]

SES - "*O Amazon Simple Email Service (SES) é um serviço de envio e recepção de emails que fornece uma maneira fácil e económica de enviar emails*". [38]

Cognito - "*O Cognito é um serviço que cria identidades exclusivas para os utilizadores, autentica-as com provedores de identidade, além de guardar dados de dispositivos móveis na Cloud AWS*". [10]

5.2 Modelo C4

O modelo C4 é um método de criação de uma arquitetura de formar incremental começando numa visão mais ampla e acabando numa visão detalhada de cada componente do *software*.

Assim, a arquitetura é dividida em quatro níveis:

1. **Contexto** - é apresentado o *scope* do sistema e relações existente com agentes e outros sistemas externos.
2. ***Containers*** - é decomposto cada sistema em diversos *containers*.
3. **Componentes** - cada *container* será decomposto em várias componentes.
4. **Código** - cada componente poderá, por fim, ser decomposta em diagramas UML.

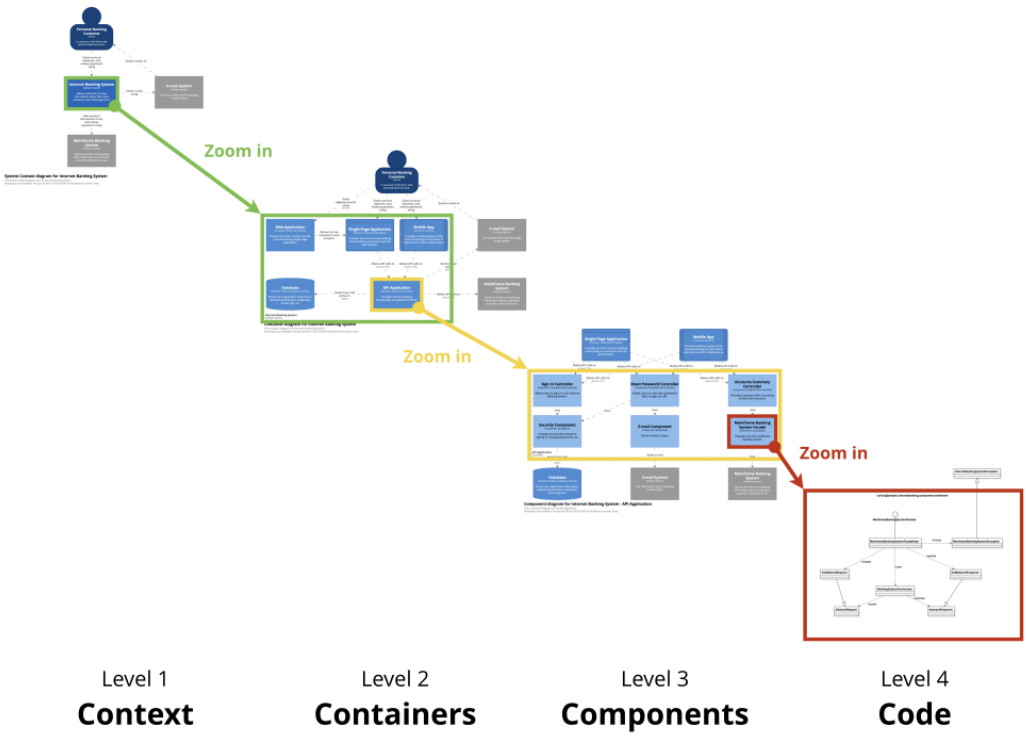


Figura 5.1: Exemplo de uma organização do modelo C4 [7]

5.3 Arquitetura

Na presente secção será apresentada a arquitetura da plataforma Sala-Z de acordo com o modelo C4.

5.3.1 Contexto

Na visão do contexto são definidos os utilizadores da plataforma, assim como, as APIs e serviços externos que a plataforma utilizará.

No presente cenário definimos dois tipos de utilizador:

Utilizador de uma *venue* - o utilizador que irá entrar na aplicação como utilizador que gere uma ou mais *venues*.

Administrador do Sala-Z - administrador da plataforma que poderá associar utilizadores comuns a *venues* e poderá gerir as *venues* e utilizadores da plataforma.

Em adição, são expostos os serviços da AWS que a plataforma irá utilizar, assim como, a APIs externas relacionadas com música que serão fontes de informação que a nossa plataforma poderá utilizar.

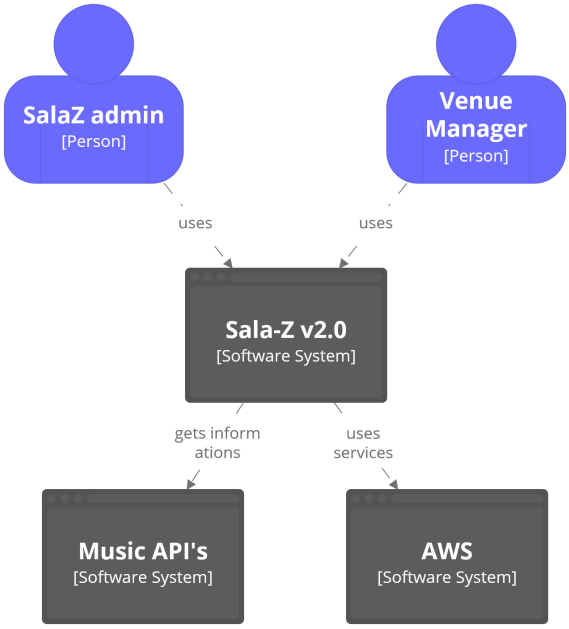


Figura 5.2: Nível 1 - Visão do contexto

5.3.2 Containers

Abrindo cada um dos sistemas de *software*, deparamo-nos com o nível dos *Containers* (Fig. 5.3) onde podemos ver a **Single Page Web Application** apresentada aos utilizadores, ou seja, o *frontend* da aplicação.

Ao ser efectuado um pedido, este será redirecionado para o *API Gateway* e de seguida para o ELB. Tendo em conta que irá haver redundância no sistema, ou seja, cada micro-serviço irá ser *deployed* em duas máquinas virtuais diferentes (EC2) para garantir que, se um EC2 cair, teremos uma máquina de *backup*, o ELB será, então, responsável por redirecionar o pedido para uma das máquinas EC2.

Assim, um EC2 alojará um dos seguintes micro-serviços da aplicação:

Authentication - serviço responsável por gerir a autenticação dos utilizadores e dos seus dados pessoais. Utilizará, principalmente, o serviço da AWS *Cognito* para gerir e validar as credenciais dos utilizadores e os *tokens* existentes.

Venue Managment - serviço responsável por gerir toda a secção das *venues* como listagem de eventos, recursos, artistas associados, entre outros.

Por fim, o serviço *Cognito* será responsável por autenticar e guardar os utilizadores que têm conta no Sala-Z.

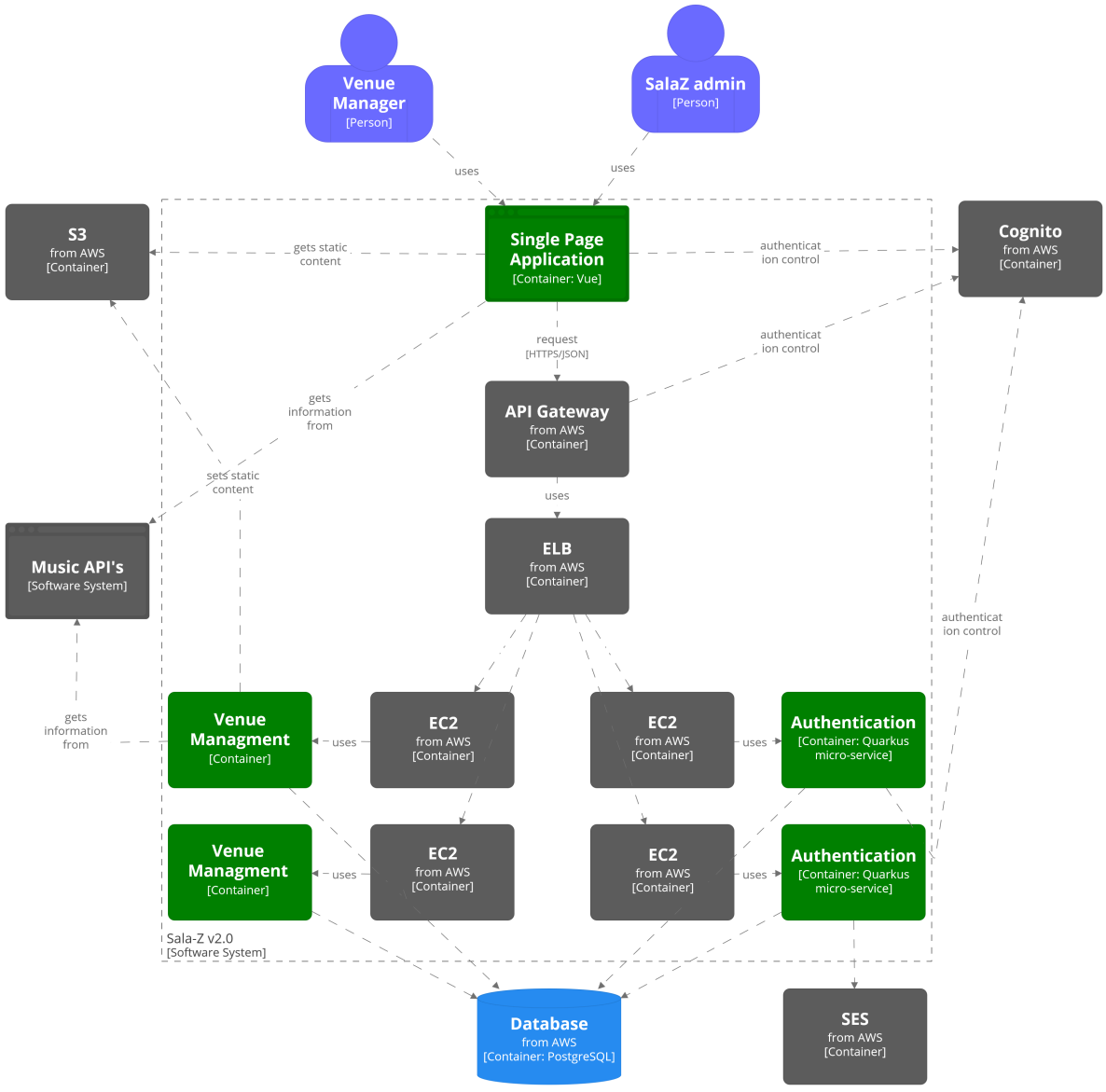


Figura 5.3: Nível 2 - Visão dos *containers*

5.3.3 Componentes

Por fim, foi detalhado cada um dos micro-serviços e a componente *web* da aplicação, visualizando a estrutura dos mesmos e as suas interações com o ambiente externo (AWS e APIs externas). Na Figura 5.4 podemos ver a componente de autenticação, sendo que o serviço da AWS, *Cognito* está no centro de ações, visto que, é o local de armazenamento seguro das informações de cada utilizador.

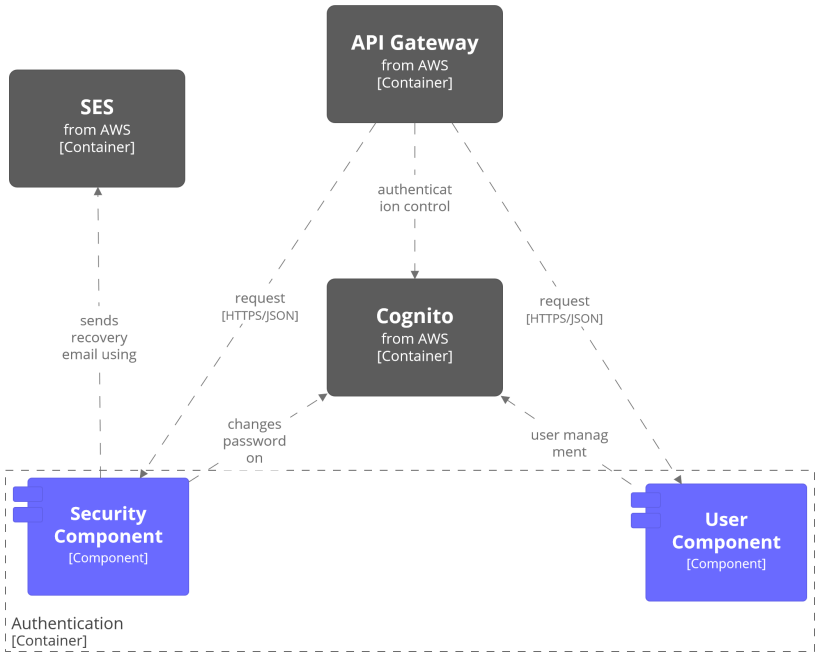


Figura 5.4: Nível 3 - Micro-serviço responsável pela autenticação no Sala-Z

De seguida, temos o micro-serviços responsável pelas ações relacionadas com a gestão de *venues* e todas as ações possíveis dentro de cada uma, com por exemplo, gerir eventos, recursos, tarefas, entre outros (Fig. 5.5).

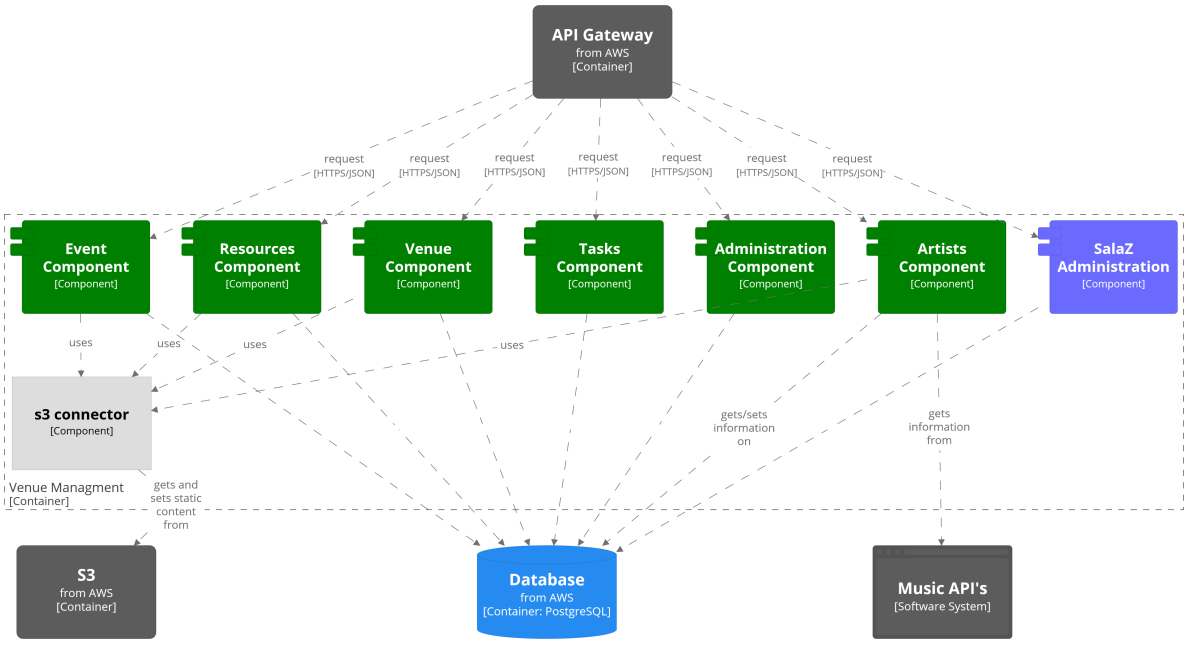


Figura 5.5: Nível 3 - Micro-serviço para gestão das *venues*

Por fim, podemos visualizar mais detalhadamente aquele que será o *frontend* do Sala-Z (Fig. 5.6). A partir de uma *single page application* o utilizador poderá visualizar cada um dos módulos da aplicação sendo que, o *Authentication guard* será responsável por garantir que o utilizador só terá acesso às páginas que têm permissões.

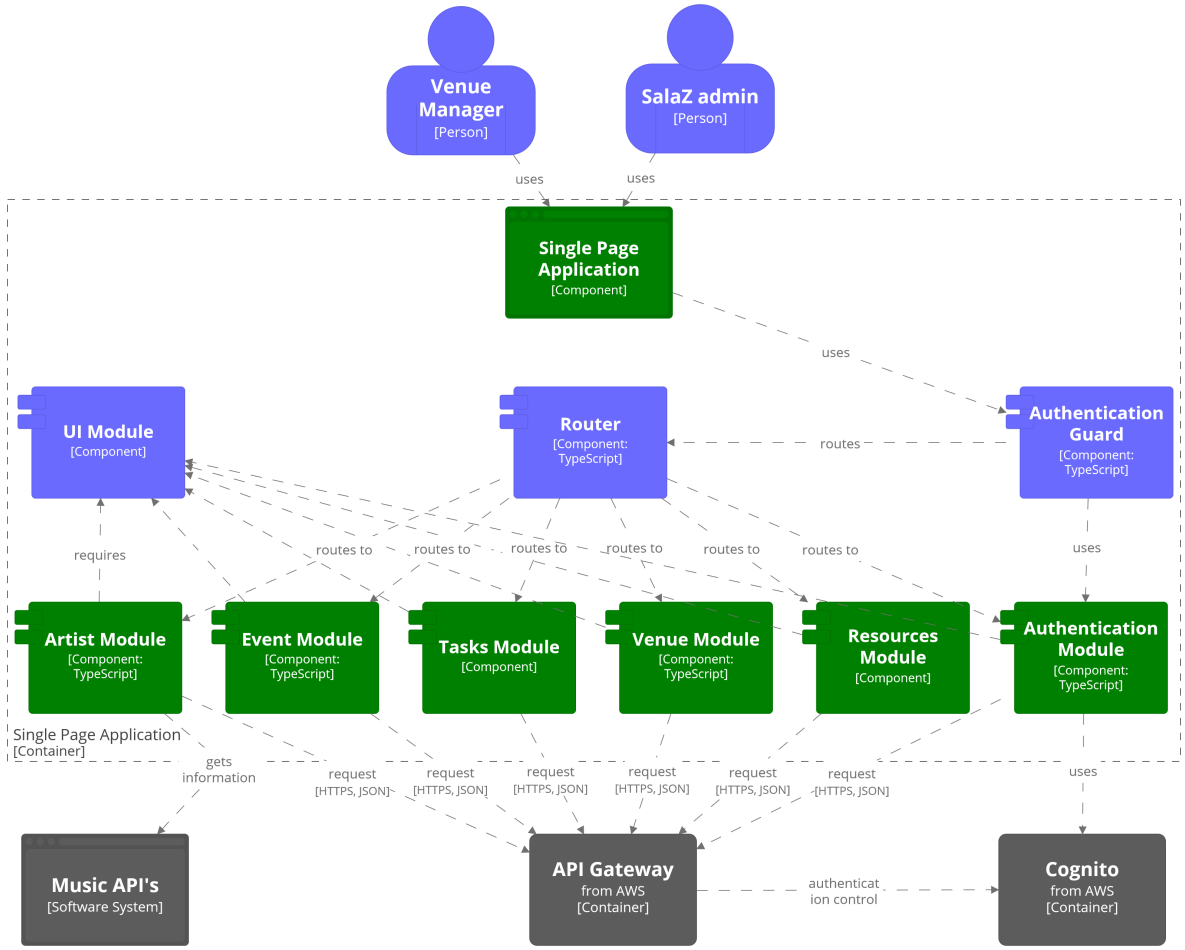


Figura 5.6: Nível 3 - Componentes do *frontend*

5.3.4 Código

Sendo que, o nível de código é meramente opcional de acordo com a documentação do modelo C4 [39] e por apresentar um nível de detalhe muito grande para uma fase inicial do projeto, este nível não será desenhado.

5.4 Diagramas Entidade-Relacionamento

De modo a visualizar as entidades presentes no projeto, foi elaborado um diagrama Entidade-Relacionamento (ER).

Um diagrama ER é uma forma de ilustrar como as entidades (ex. utilizadores, objetos, entre outros) se relacionam entre si dentro de um sistema. Neste caso (e na maior parte dos casos relacionados com Engenharia de *Software*) foi construído um diagrama para traduzir as tabelas que irão estar presentes na base de dados da plataforma.

Assim, o foco do presente ER são as três grandes entidades:

Venue: ilustra todos os dados presentes sobre uma *venue* registada no sistema (Fig. 5.7).

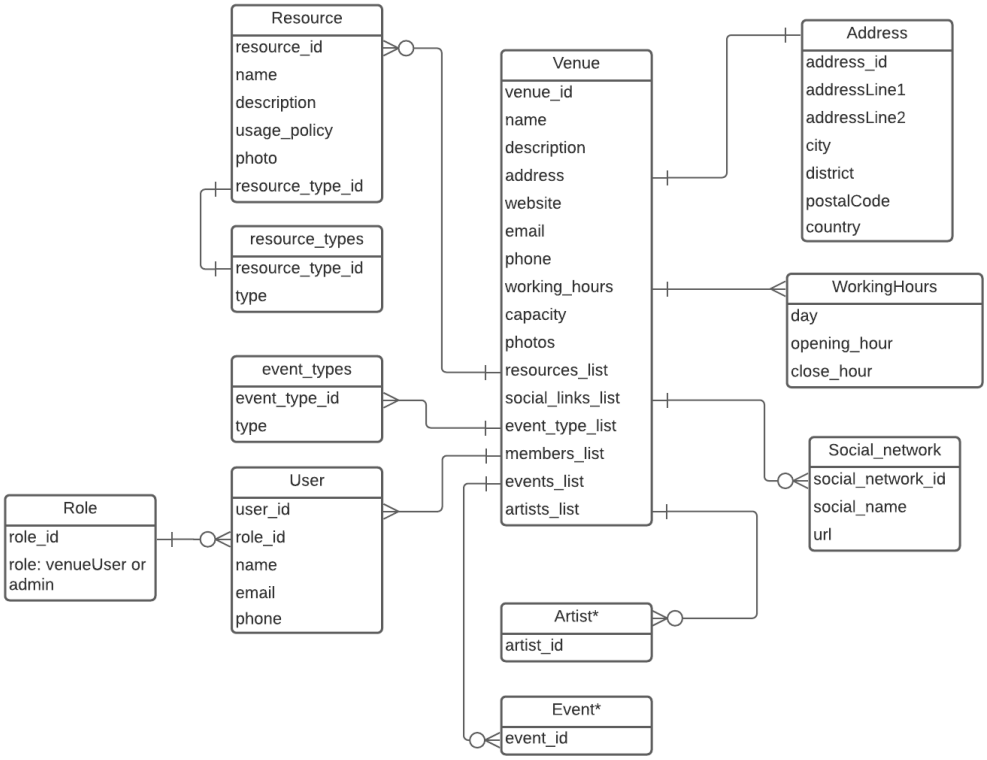


Figura 5.7: Entidade *venue* e as suas relações

Artist: ilustra todas as informações dos artistas associados a cada *venue* da plataforma (Fig. 5.8).

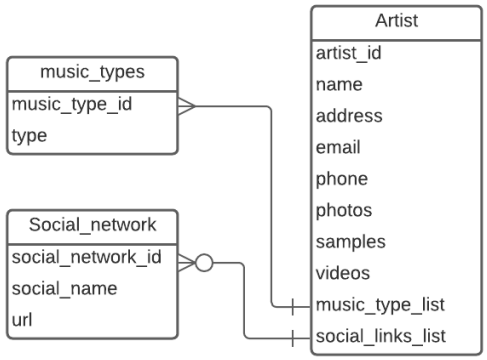


Figura 5.8: Entidade Artista e as suas relações

Event: descreve todos os eventos associados a uma *venue* (Fig. 5.9).

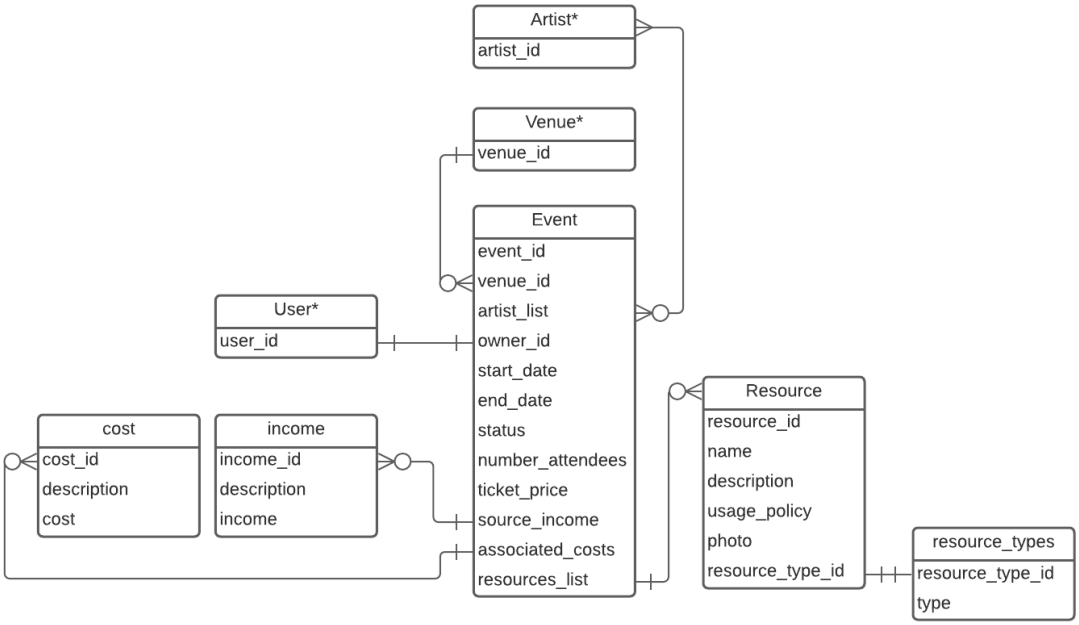


Figura 5.9: Entidade Evento e as suas relações

Capítulo 6

Desenvolvimento

O presente capítulo é dedicado à fase de desenvolvimento sendo exposto o processo adotado pelo aluno e pela restante equipa de desenvolvimento no decorrer do segundo semestre do estágio curricular, assim como, as funcionalidades desenvolvidas face aos requisitos funcionais levantados no primeiro semestre.

6.1 Processo

O processo adotado pela equipa de desenvolvimento pode ser decomposto em três partes distintas que ocorreram de forma simultânea.

Em primeiro lugar, a **construção de *mockups*** por parte da equipa de *design*. Nesta fase, todas as componentes e páginas presentes na plataforma foram desenhadas e sujeitas a aprovação por parte do *product owner* e do *scrum master*. Sendo que, para começar a desenvolver qualquer funcionalidade, os *mockups* de alta fidelidade têm de estar fechados e aprovados, este processo começou antes do aluno começar a fase de desenvolvimento, de forma a que a equipa de *design* conseguisse estar sempre à frente do processo de desenvolvimento.

Assim, o **desenvolvimento** representa a segunda parte do processo. Após cada componente estar completamente desenhada e aprovada, o aluno desenvolveu a funcionalidade alvo e todas as dependências existentes por parte dos micro-serviços. Após cada tarefa estar desenvolvida e marcada como tal, o *scrum master* e um outro *developer* da Grama fizeram *code reviews* de forma a evitar erros programáticos ou mesmo, alertar parar correcções relativas a boas práticas de escrita de código. Após esta aprovação, a tarefa era dada como *ready to test*.

Por fim, todas as tarefas marcadas como completadas foram alvo de **testes**. Nesta fase, tanto o aluno, como o **scrum master** testaram a aplicação de forma a garantir que a mesma tinha o comportamento esperado. Em adição a estes testes, foram também feitos testes de integração em cada um dos micro-serviços, garantido que todos os dados estavam a ser tratados de forma correta.

6.1.1 Tarefas

Como referido anteriormente [2.1.3], foi utilizada a ferramenta *Linear* para planear e estruturar o trabalho a ser realizado a cada *sprint*. Assim, o quadro *Kanban* foi dividido em seis colunas (Fig. 6.1): ***Backlog*** (todas as tarefas associadas ao presente projeto), ***To do*** (tarefas que estão previstas para o presente *sprint* e que foram estimadas no inicio do mesmo),***In progress*** (tarefa que está a ser desenvolvida), ***Reviewing*** (tarefas marcadas como feitas e pendentes de aprovação por parte dos *code reviews*), ***Ready to test*** (tarefas aprovadas que terão que ser testadas) e, por fim, ***Completed*** (tarefas dadas como completas).

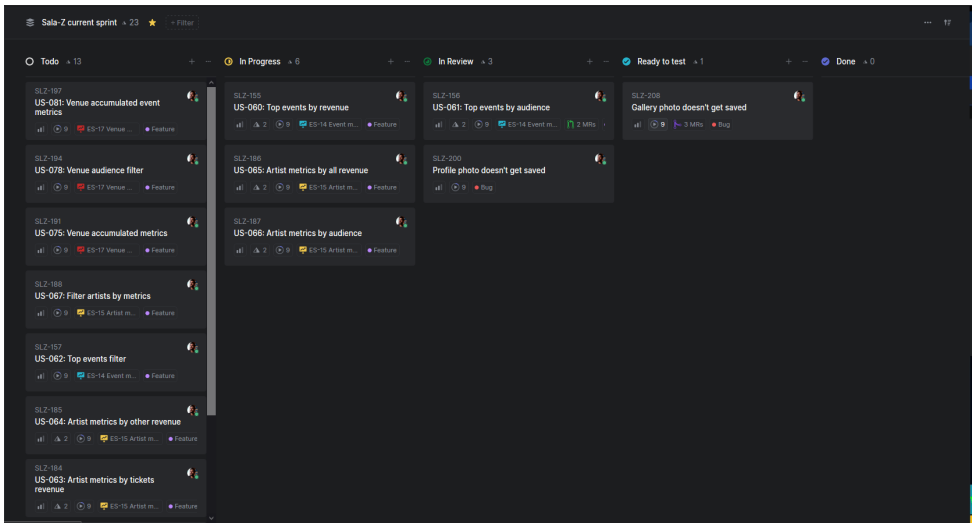


Figura 6.1: Quadro *kanban* na plataforma *Linear*

6.2 Estrutura do código

Na presente secção irá ser analisada a estrutura do código desenvolvido durante o estágio curricular. Tendo por base a estrutura do código da primeira versão do Sala-Z e com o apoio constante do coordenador da empresa, o aluno estruturou o código de forma a ser o mais semelhante possível a uma estrutura de um grande projeto da empresa.

Assim, irá ser analisado a estrutura do código de *frontend* (Subsecção 6.2.1) no seu todo e de um dos serviços de *backend*, visto que, ambos os serviços têm uma estrutura semelhante (Subsecção 6.2.2).

6.2.1 *Frontend*

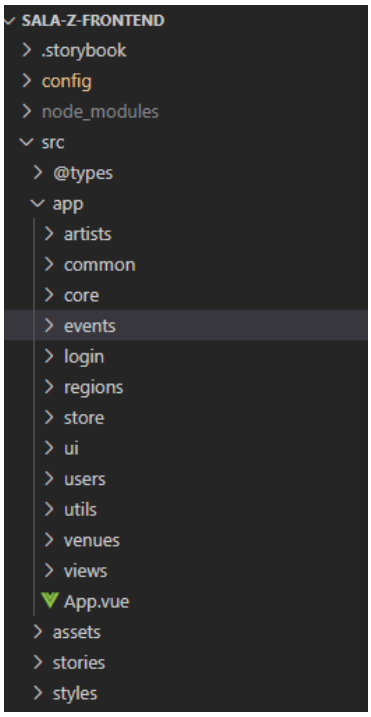


Figura 6.2: Estrutura base do *frontend* do Sala-Z

Como podemos ver na figura 6.2 o código está dividido nas seguintes pastas principais:

/**src**/**assets** - Pasta onde está todo o conteúdo estático da aplicação presente em todas as sessões dos utilizadores como logótipos, imagens fixas, entre outros.

/**src**/**styles** - Pasta onde estão armazenados todos os ficheiros de *styling scss*.

/**src**/**stories** - Pasta onde são guardadas as histórias do pacote *storybook*. O *storybook* é uma ferramenta que tem como objetivo visualizar os componentes pequenos (como botões, caixas de *input*, *etc.*) de um modo isolado, para poder testar comportamentos, diferentes temas, ações, entre outros.

/**src**/**app** - Pasta principal da aplicação onde estão guardados os diferentes módulos da mesma. Nesta, podemos ver os módulos dos artistas, eventos, *venues* e autenticação ('*login*' e '*users*') onde estão presentes os serviços, componentes e interfaces relativos aos mesmos. Em adição temos outras pastas como '*common*' e '*utils*' onde existem *scripts* de auxílio ao resto da aplicação, '*store*' onde estão guardadas as diferentes *stores* da aplicação (uma *store* é um local centralizado onde são armazenados todos os dados disponíveis em toda a aplicação [40]) e '*views*' onde estão as vistas da aplicação.

6.2.2 *Backend*

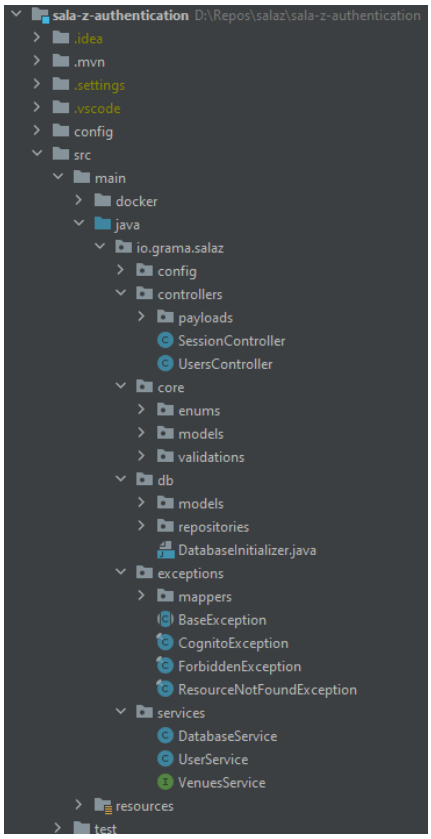


Figura 6.3: Estrutura base do *backend* do Sala-Z

Sendo que ambos os serviços de *backend* têm uma estrutura semelhante, basta analisar a estrutura do serviço de autenticação para entender como estes estão organizados.

Analisando a figura 6.3 é possível partir a estrutura em três níveis.

Controllers - São responsáveis por receber um pedido HTTP e analisar a sua estrutura. Caso o pedido esteja conforme as validações efectuadas, o pedido será processado e será enviado para um dos **serviços**.

Services - Na camada dos serviços o pedido será analisado e, consoante as permissões do utilizado, serão feitas as devidas ações sendo, ou não, necessário adquirir/mudar dados na base de dados a partir de cada um dos **repositórios**.

Repositories - Responsável por qualquer transacção à base de dados.

Em adição a esta estrutura comum em serviços de *backend*, existem outras funções/classes de auxílio como é o exemplo da pasta *"core"* onde podemos encontrar validações, modelos, entre outros, ou a pasta *"exceptions"* onde, caso o pedido não possa ser respondido corretamente, existem exceções pré-definidas.

6.3 Funcionalidades

Assim, durante o decorrer do segundo semestre foi possível desenvolver uma plataforma interessante que corresponde às expectativas levantadas no primeiro semestre e de acordo com as necessidades dos possíveis clientes contactados pela equipa da Grama.

Assim, a plataforma desenvolvida pode ser repartida em três secções distintas: **Autenticação**, ***Backoffice*** e ***Frontoffice***.

6.3.1 Autenticação

Para um utilizador aceder à aplicação terá que, previamente, ter sido convidado através de email por um administrador da plataforma. No email de convite receberá uma palavra-passe temporária que terá que alterar obrigatoriamente no seu primeiro *login*.

Assim, ao entrar na aplicação, o utilizador será abordado pelo ecrã de *login* onde poderá entrar na aplicação com o seu email e palavra-passe ou, caso se tenha esquecido da mesma, poderá a repor através de um código enviado por email (Fig. 6.4).

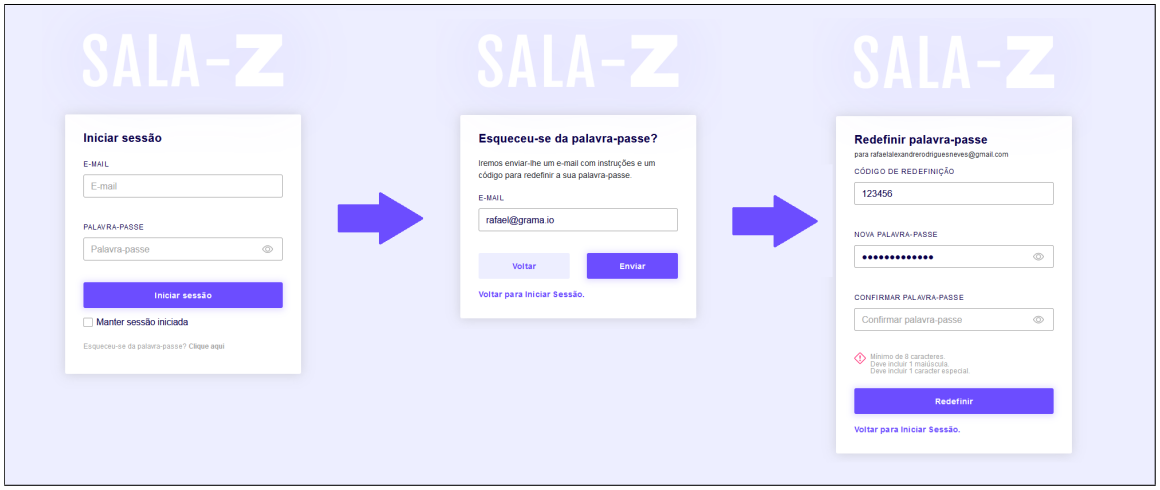


Figura 6.4: Página de *Login* e redefinição da palavra-passe

6.3.2 *Backoffice*

Para ser possível o acesso e gestão de qualquer *venue* e para ser possível um administrador da plataforma conseguir convidar outros utilizadores, o *backoffice* apresentou-se como uma tarefa de alta importância e, como tal, foi a primeira secção a ser desenvolvida após o módulo de autenticação estar completo. Assim, caso o utilizador que entrou na aplicação seja um administrador da plataforma, este terá acesso ao *backoffice* onde poderá tomar três tipos de ações.

Gestão de *venues*

Imediatamente após iniciar a sessão, é apresentado ao administrador uma listagem das *venues* existentes na plataforma onde será possível procurar por nome, região, tipo de eventos ou tipo de local (Fig. 6.5).

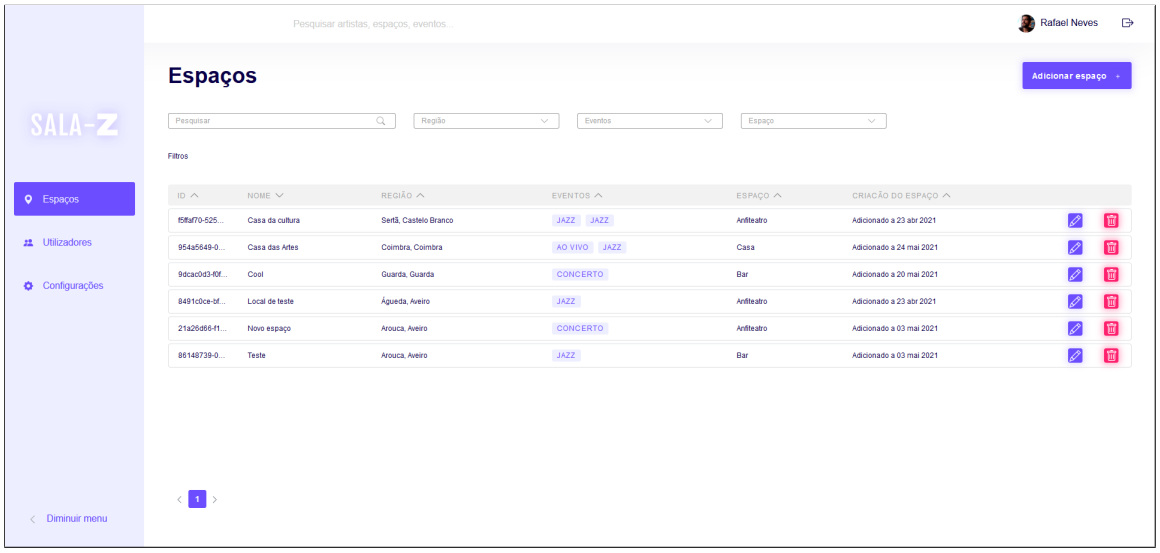


Figura 6.5: Listagem das *venues* presentes na aplicação

A partir desta página o administrador poderá tomar vários tipos de ação, como por exemplo, remover uma *venue*, editar qualquer informação de uma *venue*, ou mesmo criar um novo local (Fig. 6.6) onde terá que preencher um formulário composto por três passos.

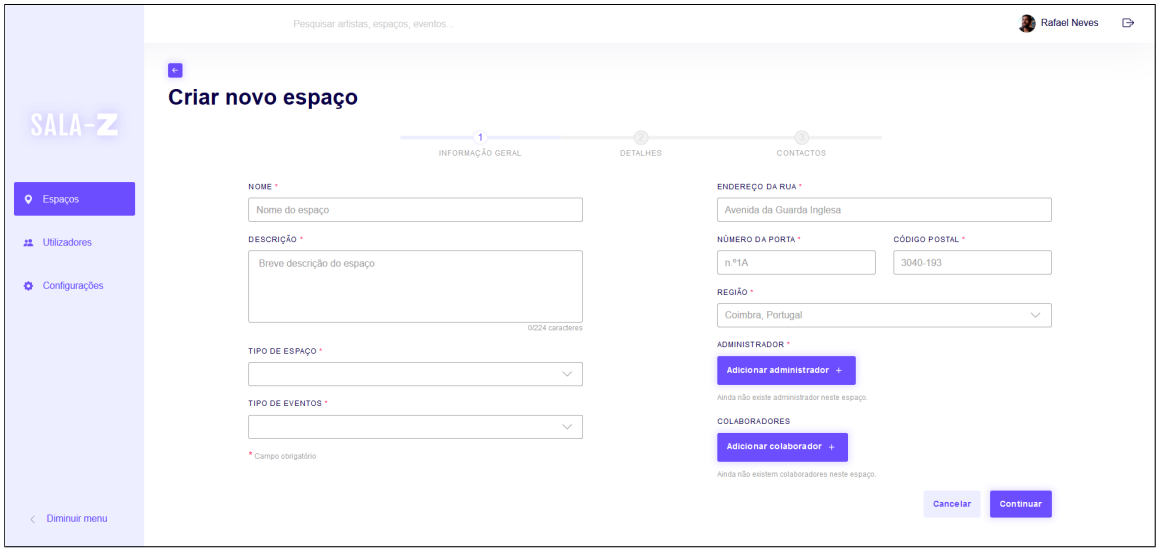


Figura 6.6: Criar uma nova *venue*

Gestão de utilizadores

Em semelhança à secção de gestão de *venues*, na secção de gestão de utilizadores, o administrador será apresentado com uma listagem dos mesmos (Fig. 6.7) onde poderá procurar por nome/email, função, espaços associados e o seu estado.

Em adição, poderá editar, eliminar ou convidar um novo utilizador para fazer parte da plataforma (com o papel de administrador ou com papel de utilizador comum da plataforma) e, caso necessário, associar o mesmo a uma *venue*.

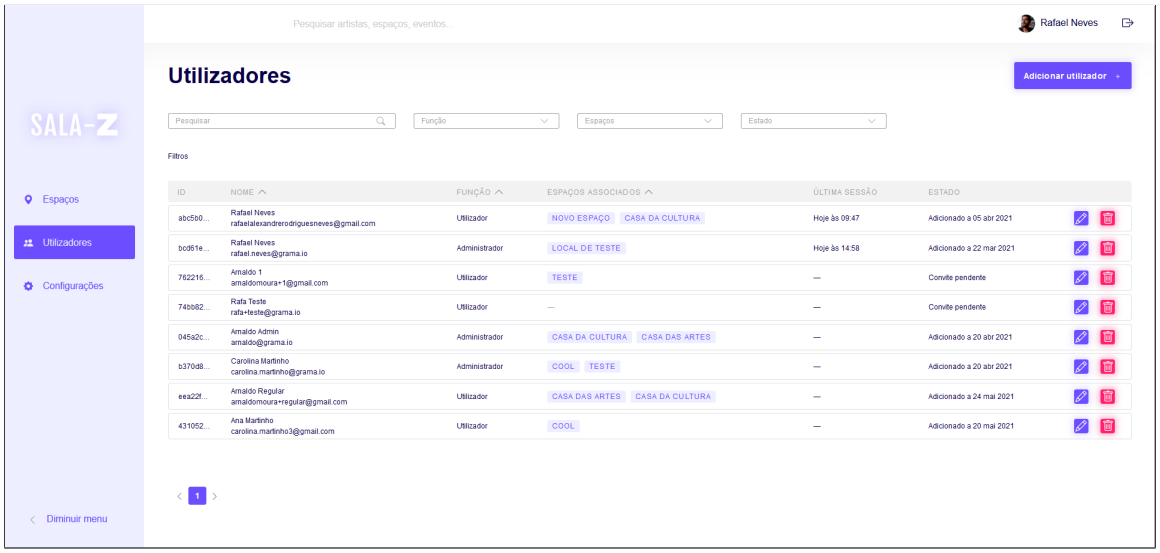


Figura 6.7: Listagem dos utilizadores presentes na aplicação

Alterar os seus dados pessoais

Por fim, o utilizador poderá alterar qualquer informação pessoal como nome, foto de perfil, email, entre outros e/ou redefinir a sua palavra-passe (Fig. 6.11).

6.3.3 *Frontoffice*

Finalmente, caso o utilizador que entrou na aplicação não seja um administrador da mesma, ou seja, qualquer utilizador convidado por um administrador do Sala-Z para fazer parte da plataforma como utilizador de uma ou várias *venues*, terá acesso a quatro secções distintas.

Dashboard

Ao entrar na aplicação, o utilizador deparar-se-á com a página da *dashboard* onde, de uma forma rápida, poderá visualizar os eventos futuros e poderá visualizar estatísticas referentes a cada um dos locais/eventos no qual está presente (Fig. 6.8).

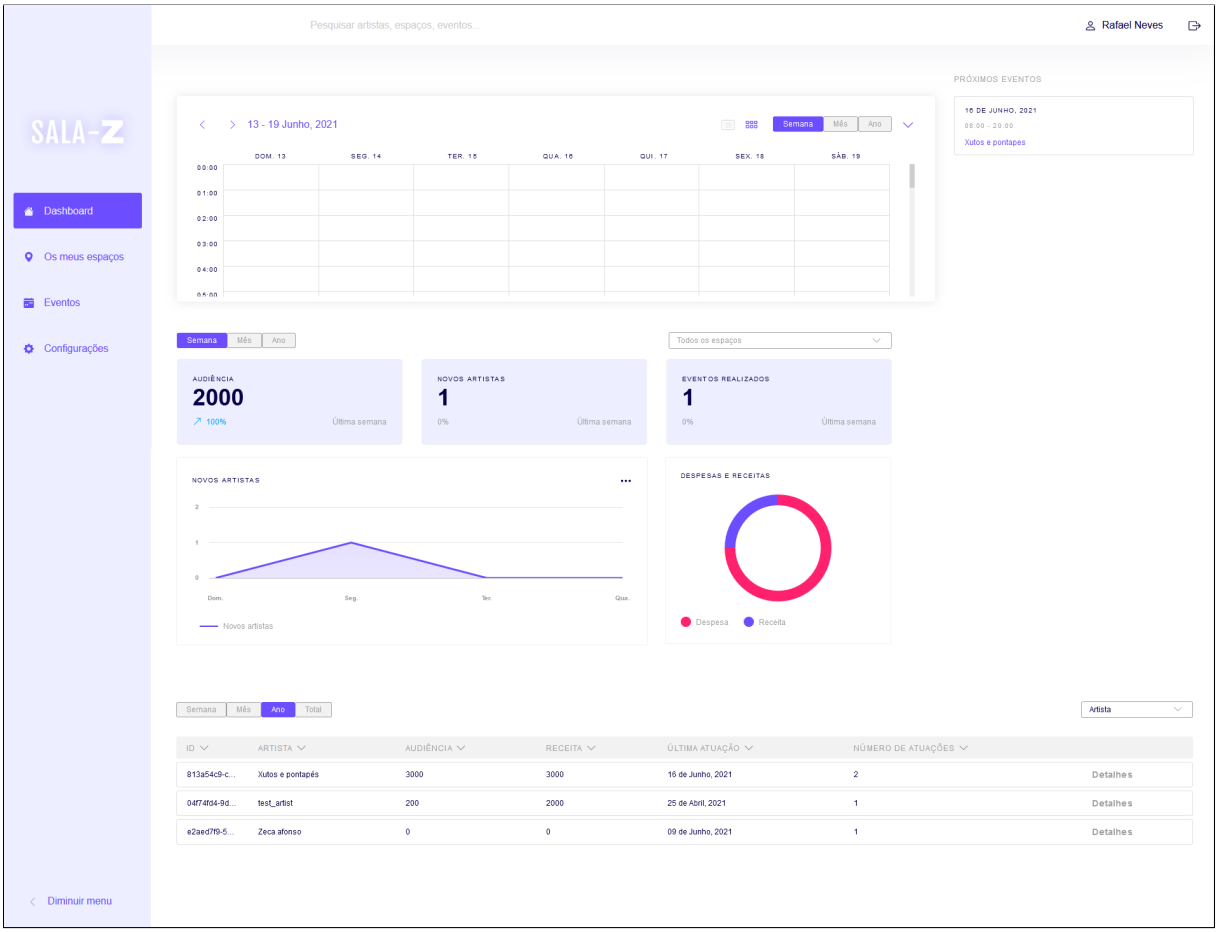


Figura 6.8: *Dashboard*

Gestão de *venues* pessoais

Na presente secção, o utilizador poderá consultar todas as *venues* na qual está presente (Fig.6.9) e visualizar ou alterar qualquer informação sobre a mesma.

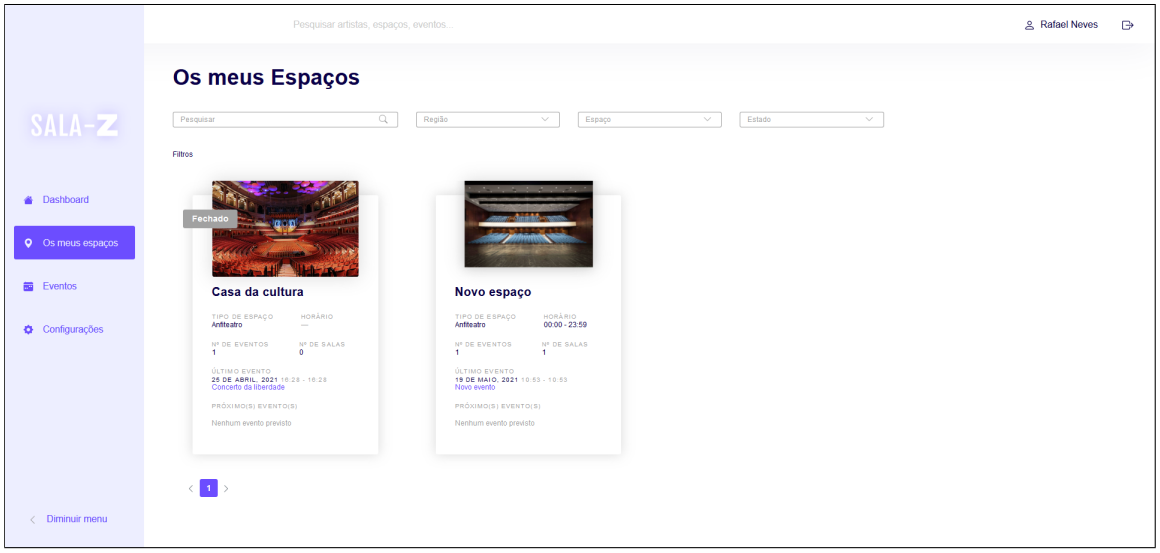


Figura 6.9: Listagem das *venues* pessoais

Eventos

Na secção dos eventos, o utilizador poderá gerir qualquer evento que esteja associado a qualquer uma das suas *venues*, podendo visualizar através de um calendário as datas dos mesmo (Fig. 6.10), visualizar os detalhes de cada um ou criar um novo evento que, à semelhança da criação de *venues* é composto por três passos.

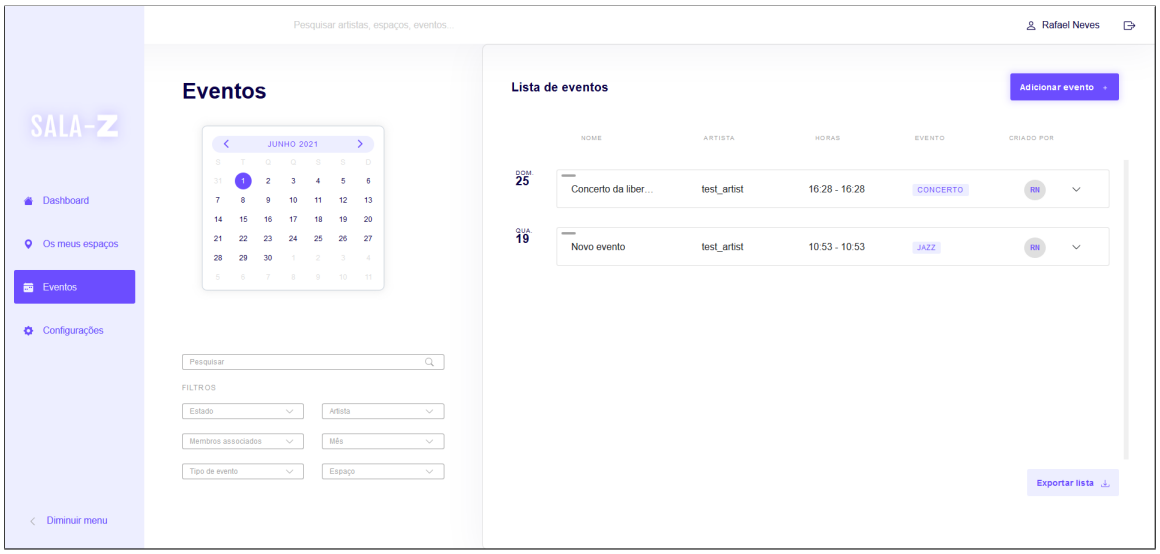


Figura 6.10: Listagem de eventos

Alterar os seus dados pessoais

De igual forma ao *backoffice*, o utilizador poderá, também, alterar qualquer informação pessoal (Fig. 6.11).

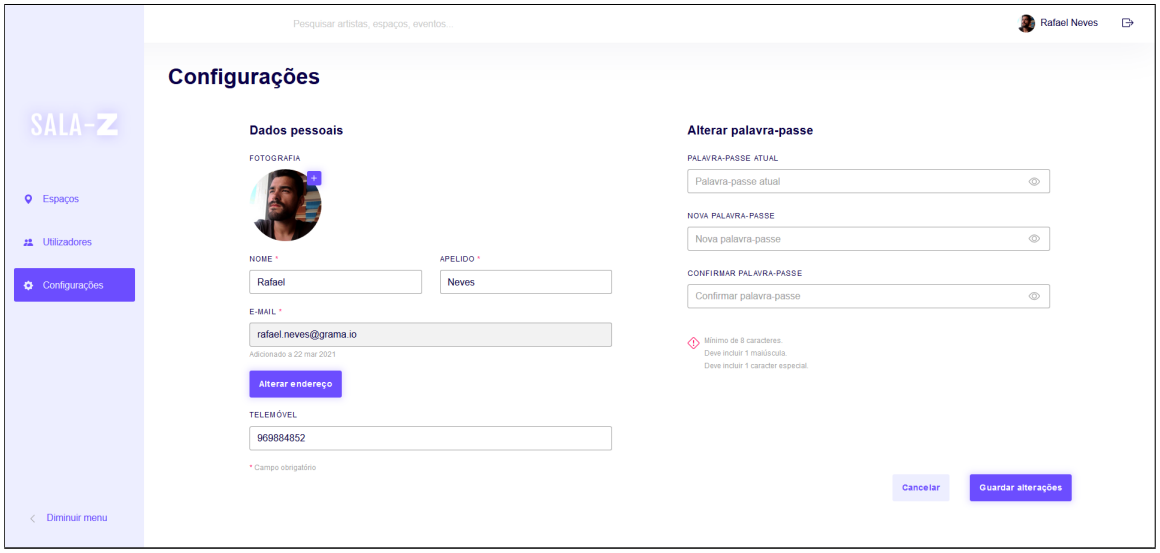


Figura 6.11: Alterar definições pessoais

6.4 Requisitos desenvolvidos

Com o terminar do segundo semestre do estágio curricular, o aluno conseguiu desenvolver grande parte dos requisitos funcionais previstos.

Tendo em conta o nível de prioridade de cada um dos requisitos levantados no decorrer do 1º semestre, a equipa procurou organizar o tempo disponível na segunda metade do ano curricular para desenvolver, pelo menos, todos os requisitos *must have* e o máximo de requisitos *should have* possível. Assim, o aluno desenvolveu os seguintes requisitos funcionais durante o estágio curricular (Tabela 6.1):

Tema	Requisito	Prioridade	Terminada
Autenticação e definições pessoais	Login	(M)	Sim
	Primeiro Login	(M)	Sim
	Logout	(M)	Sim
	Recuperar password	(M)	Sim
	Mudar password	(M)	Sim
	Registar	(C)	Não
	Mudar dados pessoais	(M)	Sim
Administração	Procurar membros da venue	(S)	Não
	Listar membros da venue	(M)	Sim
	Filtrar membros da venue	(C)	Não
	Ordenar membros da venue	(C)	Não
	Remover membros da venue	(M)	Sim
	Convidar utilizador	(M)	Sim
	Reenviar convite	(S)	Sim
Definições	Listar venues	(M)	Sim

	Pedir ajuda	(C)	Não
Definições das venues	Ver venue	(M)	Sim
	Editar venue	(M)	Sim
	Adicionar recurso à venue	(C)	Não
	Mudar recurso da venue	(C)	Não
	Listar recursos da venue	(C)	Não
	Eliminar recurso da venue	(C)	Não
Gestão de artistas	Ver lista de artistas	(S)	Sim
	Marcar artista como favorito	(C)	Não
	Editar artista	(S)	Não
	Filtrar lista de artistas	(S)	Não
	Ordenar lista de artistas	(S)	Não
	Procurar na lista de artistas	(S)	Não
	Exportar listagem de artistas	(C)	Não
	Adicionar artista à lista de artistas	(S)	Sim
	Ver um artista da lista de artistas	(S)	Não
	Editar um artista da lista de artistas	(S)	Não
	Remover um artista da lista de artistas	(S)	Não
Gestão de eventos	Adicionar novo evento	(M)	Sim
	Editar evento	(M)	Sim
	Ver evento	(M)	Sim
	Apagar evento	(M)	Sim
	Ver eventos no calendário	(M)	Sim
	Ver lista de eventos	(M)	Sim
	Filtrar lista de eventos	(S)	Sim
	Ordenar lista de eventos	(S)	Sim
	Procurar na lista de eventos	(M)	Sim
	Exportar listagem de eventos	(S)	Sim
	Adicionar tarefa ao evento	(S)	Não
	Ver tarefa	(S)	Não
	Eliminar tarefa	(S)	Não
	Editar tarefa	(S)	Não

	Adicionar tarefa a lista de tarefas	(S)	Não
	Listar tarefas	(S)	Não
	Filtrar lista de tarefas	(S)	Não
	Ordenar lista de tarefas	(S)	Não
	Pesquisar na lista de tarefas	(S)	Não
	Exportar lista de tarefas	(S)	Não
	Adicionar recurso ao evento	(C)	Não
	Eliminar recurso do evento	(C)	Não
	Editar recurso do evento	(C)	Não
	Sincronizar calendários	(C)	Não
Gestão de recursos	Listar recursos da venue	(C)	Não
	Ver utilização de recursos no calendário	(C)	Não
	Listar recursos	(C)	Não
	Filtrar lista de recursos	(C)	Não
	Ordenar lista de recursos	(C)	Não
	Pesquisar recurso na lista de recursos	(C)	Não
	Exportar lista de recursos	(C)	Não
Dashboard	Ver evento com mais receita (com filtros)	(M)	Sim
	Ver evento com mais entradas	(M)	Sim
	Ver artista que gerou mais receita (com filtros)	(M)	Sim
	Ver artista com maior número de eventos	(M)	Sim
	Receita por mês	(M)	Sim
	Receita anual por tipo	(M)	Sim
	Número de eventos por mês	(M)	Sim
	Ver eventos num calendário	(M)	Sim
	Ver tipo de música que gerou mais receita (com filtros)	(C)	Não
	Ver tipo de música com mais eventos associados	(C)	Não
	Criar venue	(M)	Sim
	Associar utilizador a venue	(M)	Sim

Backoffice	Reenviar convite	(M)	Sim
	Apagar utilizador	(M)	Sim
	Listar venues	(M)	Sim
	Listar utilizadores de venues	(M)	Sim
	Editar venue	(M)	Sim

Tabela 6.1: Requisitos funcionais realizados

Como podemos analisar na tabela 6.1, muitos dos requisitos funcionais previsto foram completados no decorrer do segundo semestre, sendo que, o objetivo de desenvolver pelo menos todos os requisitos *must have* foi, de facto, cumprido.

Por outro lado, alguns requisitos com menor prioridade não foram terminados durante o decorrer no estágio curricular, como por exemplo, todo o módulo de tarefas de um evento e recursos de uma *venue*. Apesar dos mesmos não terem sido desenvolvidos, a equipa concordou que este factor não apresenta um nível de importância significativo pois, o objetivo do estágio foi criar uma base significativa do Sala-Z, sendo que, esta plataforma ainda vai crescer muito mais no futuro e os requisitos em falta não apresentam uma importância significativa.

6.5 Riscos

Com o final do semestre, foi possível observar a análise de riscos e perceber se as estratégias de mitigação foram, ou não, significativas. Em adição, ocorreram desvios ao plano de trabalho que não foram pensados durante a análise de riscos realizada no decorrer do primeiro semestre que tiveram que ser mitigados ao longo do período de desenvolvimento.

6.5.1 Riscos ocorridos

Como apresentado no Capítulo 3 Secção 3.5, foram levantados três riscos de desenvolvimento aos quais foram atribuídos uma estratégia de mitigação cada. Para entender se estes riscos realmente ocorreram e se a estratégia de mitigação foi utilizada e eficaz vamos analisar cada risco descrito:

1. **Tecnologias** - O facto das tecnologias serem desconhecidas ao estagiário provocou um atraso significativo na fase inicial do período de desenvolvimento, porém, este atraso foi mitigado através do constante apoio do orientador de estágio da Grama e pelo facto do aluno ter desenvolvido algumas componentes antes do período de desenvolvimento começar oficialmente de modo a começar a aprender as tecnologias com antecedência.
2. **Falta de experiência** - Também devido à falta de experiência do aluno, houve a preocupação deste não conseguir terminar todos os requisitos no final do ano letivo, pelo que, foram atribuídas prioridades conforme o método de *MoSCoW*. Este método revelou-se eficaz pois, como observado na análise dos requisitos completados na secção 6.4, muitas funcionalidades não foram terminadas e, o facto de haver uma priorização dos requisitos fundamentais, possibilitou desenvolver um produto consistente e bem sucedido mesmo com os requisitos em falta.
3. **Pandemia** - A pandemia foi um fator presente durante grande parte do 1º semestre e a totalidade do 2º semestre, pelo que, o trabalho remoto esteve presente na maioria do estágio curricular. Este desvio daquilo que é o normal não teve um alto impacto no projeto pois, a presente área de trabalho, possibilita o trabalho remoto sem muitos problemas, pelo que, em adição à estratégia de mitigação de manter um contacto constate na equipa através de *dailies*, o risco não teve um alto impacto no desenvolvimento do projeto.

Assim, podemos concluir que todos riscos do projeto ocorreram no decorrer do ano letivo e, como tal, as estratégias de mitigação pensadas com antecedência foram fundamentais para o sucesso do estágio curricular.

6.5.2 Desvios imprevistos

Apesar de, durante o decorrer do 1º semestre, ter sido feita uma análise de riscos, houveram situações que não foram pensadas e que surgiram durante o decorrer da fase de desenvolvimento do produto. Assim, de maneira a mitigar estes problemas, a equipa criou estratégias quando apresentada com estas as situações.

Foram estas:

1. ***Deployments na AWS*** - Visto que a AWS oferece 750 horas mensais de alojamento numa máquina EC2, a equipa achou que era possível os serviços da *Amazon* alojarem o servidor do Sala-Z todo o mês, porém, como a plataforma tem dois micro-serviços foram necessárias duas máquinas EC2. Tal situação tornou impossível fazer com que a aplicação *web* tivesse disponível sempre.

   Estratégia de mitigação: Para mitigar este problema os *deployments* foram feitos no final de cada *sprint* e o servidor esteve *online* durante o período de uma semana após o *deploy*. Assim, o aluno sempre que fez um *deploy* comunicou à equipa para esta poder testar e ver as novas funcionalidades desenvolvidas no *sprint* anterior.

2. ***Code reviews*** - O facto dos *code reviews* serem feitos apenas por um elemento da equipa e como cada *sprint* esteve dependente da aprovação dos *merge requests* para o sucesso da mesma, surgiram alguns problemas no início do 2º semestre pois, por vezes, o orientador não teve disponibilidade para completar os *code reviews* antes do *sprint* acabar por motivos distintos (e.g.: As tarefas serem entregues poucos dias antes da *sprint* acabar ou o orientador estar de férias) resultando num atraso da implementação da tarefas e, por consequência, no início dos testes.

   Estratégia de mitigação: Para mitigar este problema, o estagiário fez um esforço para entregar as tarefas com tempo suficiente para o orientador conseguir fazer os *code reviews*. Em adição, como no final do semestre o aluno já tinha dominado alguns princípios de programação normais na empresa, os *code reviews* foram mais fáceis de realizar.

Concluindo, podemos dizer que a análise de risco foi um sucesso pois todas as estratégias de mitigação demonstraram-se úteis quando apresentados com cada um dos riscos possíveis, porém, houveram situações que a equipa não previu que tornaram o processo ligeiramente mais complicado.

Apesar destas situações terem sido desagradáveis em alguns aspetos, houve uma aprendizagem contínua da equipa que, com tempo e trabalho de equipa, formaram estratégias de mitigação que possibilitaram ultrapassar estes imprevistos.

6.6 Trabalho futuro

Após analisados os requisitos terminados neste estágio curricular, podemos concluir que a plataforma ainda tem muito para crescer.

Terminar o módulo dos artistas é, sem dúvida, um elemento fundamental no trabalho futuro, visto que, este módulo já foi começado neste estágio curricular e é um elemento de grande interesse do clientes da plataforma. Em adição, a possível comunicação entre gerentes das *venues* e os diversos artistas através da plataforma é uma funcionalidade que traz grande valor ao Sala-Z, pelo que, a equipa da Grama já começou a desenhar os *mockups* destas funcionalidades para serem desenvolvidas num futuro próximo (Fig. 6.12).

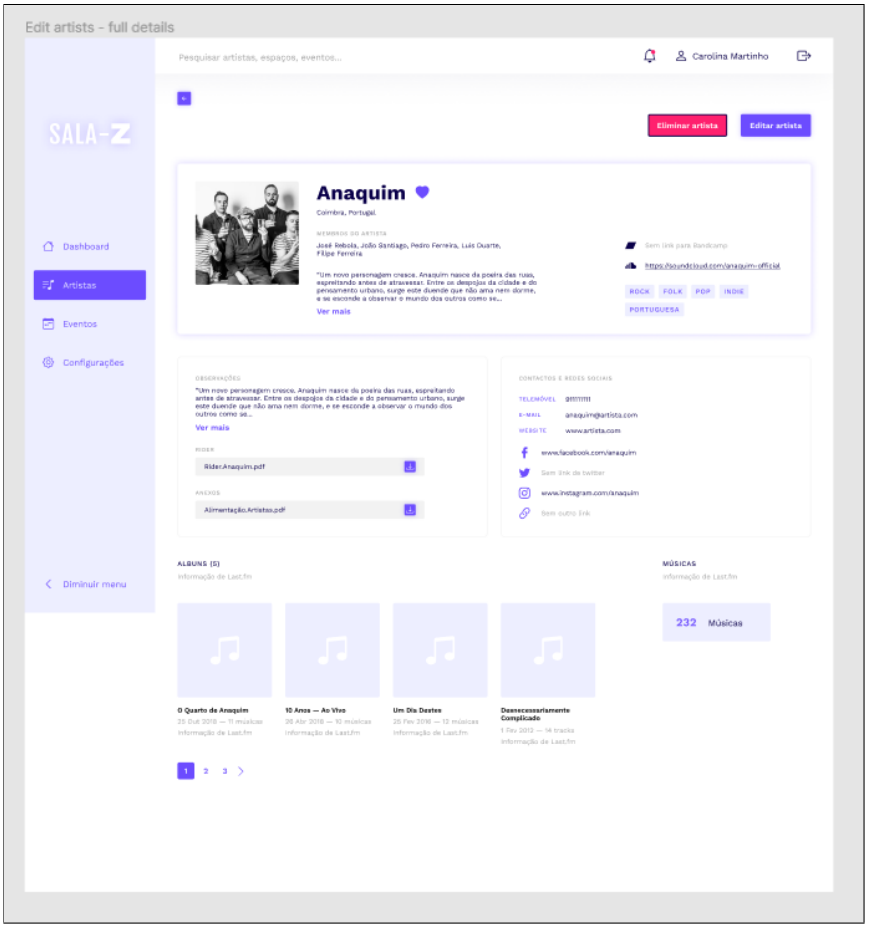


Figura 6.12: *Mockups* do módulo dos artistas

Apesar deste módulo estar a ser pensado para os gerentes das *venues* poderem ver e comunicar com os artistas, também está em mente da equipa da Grama que o Sala-Z ofereça aos próprios artistas ou agentes uma ferramenta de gestão dos seus negócios e de procura de locais para atuarem.

Em adição, existem várias funcionalidades já pensadas e discutidas, não só pela equipa de desenvolvimento da Grama, mas também com diversos clientes que mostraram interesse na plataforma, como por exemplo, introduzir tarefas associadas a eventos, assim como, introduzir bilhética na plataforma. Como tal, o Sala-Z irá ter continuidade no próximo ano letivo onde irão abrir dois novos estágios para dar continuidade ao trabalho começado durante o presente estágio curricular.

Capítulo 7

Testes

Como referido anteriormente (Secção 2.2.3), o plano de testes elaborado durante o segundo semestre seguiu a filosofia normal da empresa Grama, sendo que, foram elaborados testes automáticos sobre os dois micro-serviços da plataforma e *smoke tests* a toda a plataforma, garantido que as funcionalidades correspondentes aos requisitos fundamentais estavam, de facto, estáveis e de acordo com o esperado. Para tal, foi elaborado um plano de testes tendo por base as *user stories* e a prioridade de cada de forma a avaliar a plataforma e o sucesso da fase de desenvolvimento.

Em adição foram feitos testes aos requisitos não funcionais, garantido que a plataforma cumpre os mesmo.

7.1 Testes funcionais

Os testes funcionais sobre a aplicação foram feitos de acordo com as seguintes abordagens:

- ***white-box*** - Testes feitos pelo próprio aluno (elemento que desenvolveu a aplicação) com o objetivo de testar as funcionalidades tendo em consideração o código que escreveu e as possíveis vulnerabilidades que o mesmo possa ter.
- ***black-box*** - Testes feitos pela *designer* da equipa e por mais um elemento da mesma onde o objetivo é testar as funcionalidades e usabilidade da aplicação não tendo conhecimento prévio sobre o código desenvolvido.
- ***grey-box*** - Testes elaborados pelo *scrum master* da equipa com o objetivo de, de acordo com o conhecimento que obteve após os *merge reviews*, conseguir testar a aplicação tendo algum conhecimento sobre o código, mas também alguma abstração do mesmo.

Assim, os testes foram feitos de forma sequencial, sendo que, no final de cada *sprint*, foram testadas todas as funcionalidades desenvolvidas até ao instante presente tendo como objetivo, não só testar as funcionalidades desenvolvidas durante a *sprint*, mas também testar as funcionalidades anteriores que podem ter sido afetadas pelas mudanças recentes. Deste modo, a equipa seguiu um plano de testes constituído pelas seguintes informações:

- ***User story*** - ID da *user story* que está a ser alvo de testes.
- **ID** - ID do teste.
- **Cenário** - Cenário correspondente à ação que o teste está a avaliar.
- **Descrição** - Breve descrição sobre o objectivo do teste.
- **Passos** - Passos que o *tester* tem que seguir para realizar o teste.

- **Resultado esperado** - Resultado esperado depois da ação ser realizado.
- **Resultado obtido** - Resultado obtido pela ação realizada e se vai de acordo com o resultado esperado.
- **Estado** - Estado do teste (Passou ou Falhou).

A totalidade do plano pode ser consultado na Apêndice B ou pode ser analisado um excerto na seguinte tabela (Tabela 7.1):

User story	ID	Cenário	Descrição	Passos	Resultado esperado	Resultado obtido	Estado (P/F)
US-001	1	Login válido como utilizador de uma *venue*	Verificar se um utilizador com credenciais válidas consegue iniciar a sessão	- Abrir a aplicação - Inserir email correto - Inserir palavra-passe correta - Submeter	Entrar no *frontoffice*	Entrar no *frontoffice*	P
	2	Iniciar sessão com email inválido	Verificar se um email inválido produz uma mensagem de erro	- Abrir a aplicação - Inserir email incorreto - Inserir palavra-passe correta - Submeter	Mensagem de erro	Mensagem de erro	P
	3	Iniciair sessão com palavra-passe inválida	Verificar se uma *password* inválida produz uma mensagem de erro	- Abrir a aplicação - Inserir email correto - Inserir palavra-passe incorreta - Submeter	Mensagem de erro	Mensagem de erro	P
	4	Iniciar sessão e não manter sessão iniciada	Verificar se, após sair da aplicação, a sessão é terminada após não ter selecionado a opção "Manter sessão iniciada"	- Abrir a aplicação - Inserir credenciais válidas - Submeter - Fechar browser - Abrir aplicação	Página de *Login*	Página de *Login*	P
	5	Iniciar sessão e manter sessão iniciada	Verificar se, após sair da aplicação, a sessão é mantida após ter selecionado a opção "Manter sessão iniciada"	- Abrir a aplicação - Inserir credenciais válidas - Selecionar opção de manter sessão - Submeter - Fechar browser - Abrir aplicação	Página principal	Página principal	P
US-002	6	Terminar sessão	Verificar se sessão é terminada após clicar *logout*	- Entrar na aplicação - Clicar no icon de *logout* - Voltar uma página atrás	Sair para o *Login* e não ser possível voltar ao menu	Sair para o *Login* e não ser possível voltar ao menu	P
US-003	7	Recuperar palavra-passe com email correto e inserir código errado	Verificar se, ao inserir um código inválido, é apresentada uma mensagem de erro	- Entrar na aplicação - Recuperar palavra-passe - Inserir email correto - Inserir código inválido e palavra-passe válida	Mensagem de erro	Mensagem de erro	P
	8	Recuperar palavra-passe com email correto e inserir código correto	Verificar se, ao inserir um código válido, a palavra-passe é alterada	- Entrar na aplicação - Recuperar palavra-passe - Inserir email correto - Inserir código válido e palavra-passe válida	Palavra-passe alterada com sucesso	Palavra-passe alterada com sucesso	P
US-004	9	Não mudar palavra-passe no primeiro *Login*	Verificar se o utilizador consegue ultrapassar o passo de alteração da palavra passe sem alterar a palavra passe	- Inserir credenciais de novo utilizador - Voltar atrás no ecrã de primeiro Login - Inserir as credenciais de novo	Ecrã de primeiro *Login*	Ecrã de primeiro *Login*	P
	10	Mudar palavra-passe no primeiro *Login*	Verificar se o utilizador consegue efectuar o *Login* após mudar a palavra-passe	- Inserir credenciais de novo utilizador - Mudar a palavra-passe - Entrar na aplicação com palavra-passe nova	Página principal	Página principal	P
US-005	11	Adicionar um novo utilizador a uma *Venue*	Verificar se um utilizador com uma *Venue* pode adicionar um novo utilizador	- Navegar até edição de uma *venue* - Adicionar colaborador com email, primeiro norme e ultimo nome válido.	Convite enviado para utilizador	Convite enviado para utilizador	P
US-006	12	Reenviar um convite ao utilizador	Não desenvolvido	-	-	-	-
US-007	13	Listar todos os membros de um espaço associado	Verificar se um utilizador com uma *Venue* associada pode visualizar todos os outros membros	- Navegar até edição de uma *venue* - Adicionar um novo colaborador. - Ver detalhes da *Venue*	*Venue* com lista de utilizadores, incluindo o recentemente adicionado e o próprio	*Venue* com lista de utilizadores, incluindo o recentemente adicionado e o próprio	P

Tabela 7.1: Excerto do plano de testes levado a cabo no final do 2º semestre

7.1.1 Testes automatizados

Em adição aos *smoke tests* realizados sobre a aplicação, foram feitos testes de integração sobre cada um dos micro-serviços existentes de forma a, de uma forma rápida e eficaz, testar todos os *endpoints* existentes garantindo que os dados estão a ser tratados de forma correta.

Como é possível visualizar na imagem em baixo (Fig. 7.1), com o auxílio da ferramenta **JUnit**, cada teste é composto por um pedido feito ao micro-serviço e por uma análise à resposta obtida. Cada *script* irá enviar um certo número de pedidos, irá comparar a resposta com a esperada e, caso algum campo da resposta não corresponda ao esperado, o teste irá falhar.

```
@Test
public void changeEvent() {
    Venue venue = getVenue();
    venue = venuesDatabaseService.createVenue(venue);

    EventRequest request = getEventRequest(venue);
    Event event = eventsDatabaseService.createEvent(EventConverter.getInstance().convert(request), venue.getId(), request.getRooms());

    event.setName("Changed");
    event.setDescription("Changed");
    event.setParticipants(10000);
    event.setAudience(EventAudience.ADULT);

    ValidatableResponse response = given() RequestSpecification
            .header( s: "Content-Type", MediaType.APPLICATION_JSON)
            .body(event)
            .when().put( s: "/events/" + event.getId()) Response
            .then() ValidatableResponse
            .body( s: "payload.name", Matchers.is( value: "Changed"),
                    ...objects: "payload.description", Matchers.is(event.getDescription()),
                    "payload.participants", Matchers.is(event.getParticipants()),
                    "payload.audience", Matchers.is(event.getAudience()),
                    "category", is(Category.SUCCESSFUL.toString()))
            .statusCode(200);

    venueRepository.deleteById(venue.getId());
}
```

Figura 7.1: Teste automatizado sobre o *endpoint* de edição de eventos

Este tipo de testes foi corrido após todos os *merge requests* feitos para o *branch* principal, *develop* e imediatamente antes de todos os *deployments* feitos pelo aluno para, de uma maneira sistemática e rápida, garantir que nenhuma funcionalidade nova alterou o comportamento de uma funcionalidade já desenvolvida.

7.2 Testes não funcionais

Em adição aos testes funcionais elaborados no decorrer do segundo semestre, foram também elaborados testes sobre a escalabilidade e *performance* da plataforma quando sujeita a um alto número de pedidos atráves de *load testing*.

Load testing são testes cujo o objetivo é analisar o desempenho da aplicação quando submetido a um alto número de pedidos. Para determinar o desempenho da aplicação *web* desenvolvida durante o presente estágio, a mesma foi sujeita a um acréscimo de pedidos para, de forma comparativa, determinar o desempenho da plataforma quando o número de pedidos aumenta.

Tendo em conta que os micro-serviços estão alojados em máquinas AWS EC2 (ver capítulo 5.3) com especificações base, ou seja, 1 CPU e 2 GB de ram, os testes foram feitos de forma a testar o uso da plataforma de acordo com o número de utilizadores espectável numa fase inicial de uso da mesma. Assim, assumindo que a plataforma terá 50 utilizadores na sua fase inicial e que cada utilizador irá efectuar uma totalidade de 10 pedidos por hora.

Os resultados dos testes realizados foram os seguintes (Tabela 7.2):

# de pedidos	# de utilizadores	Tempo (ms)		
		Min.	Médio	Máx.
10	1	60	67	110
20	2	59	67	206
50	5	61	72	122
100	10	61	86	200
200	20	61	123	208
400	40	60	164	299
500	50	58	205	526

Tabela 7.2: *Load testing* realizado ao Sala-Z

Para fazer uma análise correta de *load testing* da plataforma, os testes foram corridos sobre a ação de um utilizador entrar na *dashboard* do Sala-Z, visto que, será a página inicial da plataforma e, possivelmente, a página mais visitada por parte dos utilizadores.

7.2.1 Conclusões

Analisando os resultados podemos concluir que os mesmos foram bastante positivos, sendo que, o pior resultado obtido foi de 0.5 segundos no cenário em que 50 utilizador, simultaneamente, realizam 10 pedidos ao servidor.

Sendo que, geralmente, o período de atenção normal de um utilizador ronda os 2 e os 4 segundos, ou seja, este valor representa o limite máximo desejável que uma aplicação deve responder a qualquer pedido do utilizador, podemos concluir que a plataforma, sobre circunstancias normais, cumpre o presente requisito não funcional.

Capítulo 8

Conclusão

Com o terminar do estágio curricular, o estagiário, para além de uma experiência fundamental dos conceitos básicos da estrutura de uma empresa e das metodologias de trabalho da consequente equipa de desenvolvimento, trabalhou e aprendeu com vários profissionais da área de engenharia de *software*.

Com o objetivo de criar uma plataforma organizada, estruturada e de qualidade, o aluno começou por planear o ano curricular. Este factor teve uma grande importância, não só para organizar a ordem do trabalho, mas também para entender que tarefas poderiam ser concluídas no espaço de tempo disponível.

Com o ano curricular planeado, o aluno começou por fazer uma análise do estado da arte que foi, provavelmente, um dos estudos mais importantes no decorrer do primeiro semestre. Este permitiu ao aluno e à equipa entender as necessidades das *venues*, assim como, entender aquilo que o mercado já oferece e como o Sala-Z se poderá destingir dos produtos já existentes. Em adição o estudo dos riscos da plataforma teve um papel fundamental no sucesso do segundo semestre e do desenvolvimento da plataforma através das estratégias de mitigação criadas com antecedência.

Com o estudo do estado da arte e com um conjunto alargado de *user stories* e consecutivos requisitos funcionais e não funcionais baseados no mesmo, o aluno acabou o primeiro semestre com o desenho da arquitetura do produto seguindo o modelo C4. Este modelo apresentou inúmeras vantagens pois, por começar com um nível de visão geral e depois detalhar cada uma das componentes da plataforma, o aluno conseguiu mais facilmente partir o produto complexo em vários módulos mais pequenos e simples de entender.

Assim, o aluno começou o segundo semestre com o desenvolvimento da plataforma utilizando *Vue* com *Typescript* para o *frontend* e micro-serviços na linguagem *java* utilizando a ferramenta *Quarkus*. Durante este período, o aluno aprendeu imenso com a equipa de desenvolvimento da Grama não só pelo desafio que foi criar a plataforma de gestão de salas de eventos musicais mas, também, pelos diversos *code reviews* que elementos com alta experiência fizeram aos diversos *merge requests* do aluno.

Finalmente, e de forma a garantir que o produto está de acordo com as expectativas da equipa e dos clientes, a plataforma foi testada a nível funcional e não funcional para garantir que todas as funcionalidades desenvolvidas apresentam o comportamento esperado.

Após o terminar do estágio curricular, as expectativas do aluno foram alcançadas, sendo que, desenvolveu um enorme conjunto de aptidões e capacidades ao trabalhar no ambiente profissional e acolhedor que foi o da empresa Grama.

Apesar das dificuldades encontradas durante o semestre, como por exemplo, o facto da pandemia atual não possibilitar o trabalho presencial durante o segundo semestre e o facto do aluno nunca ter tido contacto com as ferramentas usadas durante o estágio curricular (em particular com as ferramentas de desenvolvimento *Vue* e *Quarkus*), o aluno e a equipa da Grama sentiram que o estágio foi um sucesso, sendo que, apesar de alguns dos requisitos expostos durante o primeiro semestre não terem sido desenvolvidos por falta de tempo para tal, a plataforma apresentou todas as funcionalidades base e fundamentais.

Referências

[1] Tuleap. Understanding agile scrum in 10 minutes. https://www.tuleap.org/agile/agile-scrum-in-10-minutes/. (Acedido a: 10-01-2021).

[2] Project Builder. É possível usar um kanban para gerenciar projetos? https://www.projectbuilder.com.br/blog/e-possivel-usar-um-kanban-para-gerenciar-projetos/. (Acedido a: 27-09-2020).

[3] Chuka Ofili. Gitflow workflow, automated builds, integration and deployment. https://iamchuka.com/gitflow-workflow-continuous-integration-continu/. (Acedido a: 03-12-2020).

[4] Grama. Sala-z versão 1.0. https://sala-z.grama.io/portal. (Acedido a: 28-09-2020).

[5] Shaumik Daityari. Which framework to choose in 2021. https://www.codeinwp.com/blog/angular-vs-vue-vs-react/, . (Acedido a: 23-10-2020).

[6] Bianca Napoleão. Matriz de riscos. https://ferramentasdaqualidade.org/matriz-de-riscos-matriz-de-probabilidade-e-impacto/. (Acedido a: 17-01-2021).

[7] Jean-Baptiste Sarrodie. C4 model, architecture viewpoint and archi 4.7. https://www.archimatetool.com/blog/2020/04/18/c4-model-architecture-viewpoint-and-archi-4-7/. (Acedido a: 16-11-2020).

[8] Kent Beck, Mike Beedle, Arie Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto para o desenvolvimento Ágil de software. https://agilemanifesto.org/iso/ptpt/manifesto.html. (Acedido a: 10-01-2021).

[9] Guru99. What is smoke testing? https://www.guru99.com/smoke-testing.html. (Acedido a: 15-01-2021).

[10] Amazon. Cognito. https://aws.amazon.com/pt/cognito/, . (Acedido a: 15-10-2020).

[11] Gigwell. https://www.gigwell.com/. (Acedido a: 13-10-2020).

[12] Gigplanner. https://gigplanner.com/. (Acedido a: 13-10-2020).

[13] Empis. Coliseum. https://empisacessos.pt/solucoes/gestao-de-salas-de-espectaculos/, . (Acedido a: 02-10-2020).

[14] Empis. https://empisacessos.pt/, . (Acedido a: 02-10-2020).

[15] Prism.fm. https://prism.fm/. (Acedido a: 12-10-2020).

[16] Muzeek. https://www.muzeek.com/. (Acedido a: 12-10-2020).

[17] Shaumik Daityari. Angular vs react vs vue: Which framework to choose in 2021. https://www.codeinwp.com/blog/angular-vs-vue-vs-react/, . (Acedido a: 21-10-2020).

[18] Vue.js. Comparison with other frameworks. https://vuejs.org/v2/guide/comparison.html. (Acedido a: 21-10-2020).

[19] inVerita. Vue vs react vs angular: What framework would you choose? https://medium.com/swlh/vue-vs-react-vs-angular-what-framework-would-you-choose-5d77a3680b0d. (Acedido a: 21-10-2020).

[20] Amazon. Lambda. https://aws.amazon.com/pt/lambda/, . (Acedido a: 15-10-2020).

[21] TipsCode. Express vs spring boot – qual é o melhor? https://www.tipscode.com.br/express-vs-spring-boot-qual-e-o-melhor/. (Acedido a: 30-10-2020).

[22] Spring. Boot. https://spring.io/projects/spring-boot. (Acedido a: 30-10-2020).

[23] RedHat. What is kubernetes? https://www.redhat.com/en/topics/containers/what-is-kubernetes, . (Acedido a: 02-11-2020).

[24] RedHat. What is quarkus? https://www.redhat.com/pt-br/topics/cloud-native-apps/what-is-quarkus, . (Acedido a: 02-11-2020).

[25] Ryan Gleason. Node.js vs. spring boot — which should you choose? https://medium.com/better-programming/node-js-vs-spring-boot-which-should-you-choose-2366c2f76587. (Acedido a: 21-10-2020).

[26] Soufiane Sakhi. Quarkus vs spring boot: A real-world performance comparison. https://simply-how.com/quarkus-vs-spring-boot-production-performance. (Acedido a: 02-11-2020).

[27] Intagleo. The most popular backend frameworks for web development in 2020. https://www.intagleo.com/blog/most-popular-backend-frameworks-for-web-development-in-2019/. (Acedido a: 21-10-2020).

[28] Shareem Thahir. 7 reasons why serverless computing will create a revolution in cloud technology. https://datafloq.com/read/7-reasons-serverless-computing-revolution-cloud/2871. (Acedido a: 01-11-2020).

[29] Radon. 5 reasons why serverless computing is a revolution in cloud technology. https://radon-h2020.eu/2019/10/11/serverless-revolution-radon/. (Acedido a: 01-11-2020).

[30] Bernard Brode. Why the serverless revolution has stalled. https://www.infoq.com/articles/serverless-stalled/, . (Acedido a: 02-11-2020).

[31] Bernard Brode. Why the serverless revolution has stalled. https://www.infoq.com/articles/serverless-performance-cost/, . (Acedido a: 02-11-2020).

[32] Amazon. Ec2. https://aws.amazon.com/pt/ec2/, . (Acedido a: 15-10-2020).

[33] Amazon. Cloudfront. https://aws.amazon.com/pt/cloudfront/, . (Acedido a: 15-10-2020).

[34] Amazon. S3. https://aws.amazon.com/pt/s3/, . (Acedido a: 15-10-2020).

[35] Amazon. Api gateway. https://aws.amazon.com/pt/api-gateway/, . (Acedido a: 15-10-2020).

[36] Amazon. Elb. https://aws.amazon.com/pt/elasticloadbalancing/, . (Acedido a: 15-10-2020).

[37] Amazon. Rds. https://aws.amazon.com/pt/rds/, . (Acedido a: 15-10-2020).

[38] Amazon. Ses. https://aws.amazon.com/pt/ses/, . (Acedido a: 15-10-2020).

[39] C4 model. https://c4model.com/. (Acedido a: 16-11-2020).

[40] Vue school. What is a store in vue.js? https://vueschool.io/articles/vuejs-tutorials/what-is-a-store-in-vue-js/. (Acedido a: 21-06-2021).

[41] Runrun. Metodologia ágil: um presente da indústria de software para todo o universo da gestão. https://blog.runrun.it/metodologia-agil/. (Acedido a: 10-01-2021).

[42] Cprime. What is agile? what is scrum? https://www.cprime.com/resources/what-is-agile-what-is-scrum/. (Acedido a: 12-01-2021).

[43] Smartbear. What is code review? https://smartbear.com/learn/code-review/what-is-code-review/. (Acedido a: 03-12-2020).

[44] circleci. How to test software, part i: mocking, stubbing, and contract testing. https://circleci.com/blog/how-to-test-software-part-i-mocking-stubbing-and-contract-testing/. (Acedido a: 15-01-2021).

[45] Sushil Tripathi. 7 most popular backend web development frameworks in 2020. https://www.kelltontech.com/kellton-tech-blog/7-most-popular-backend-web-development-frameworks-2020. (Acedido a: 21-10-2020).

[46] Amazon. Aws. https://aws.amazon.com/pt/, . (Acedido a: 15-10-2020).

Apêndices

A Apêndice A

A.1 User stories

Módulo das *Venues*

1. Gestão da autenticação

ES-1: **Como um** utilizador de um espaço autenticado **quero** garantir que o Sala-Z apenas é utilizador por utilizadores autenticados **para** assegurar que todas as funcionalidades da plataforma apenas são acedidas por utilizadores autorizados.

US-1: *Login*

i. **Como um** utilizador de um espaço **quero** entrar no Sala-Z **para** usar as funcionalidades da plataforma.

US-2: Terminar Sessão

i. **Como um** utilizador de um espaço **quero** terminar a sessão **para** outros utilizadores que têm acesso ao meu aparelho não possam utilizar a plataforma em meu nome.

US-3: Recuperação da *password*

i. **Como um** utilizador de um espaço **quero** pedir uma recuperação de *password* **para** poder aceder à minha conta no caso de me esquecer da *password*.

US-4: Primeiro *Login*

i. **Como um** utilizador de um espaço **quero** completar o meu primeiro *login* **para** poder aceitar um convite de acesso ao Sala-Z e, consequentemente, mudar a minha *password* definida pelo sistema.

2. Painel de administração

ES-2: **Como um** utilizador de um espaço autenticado **quero** gerir os utilizadores com acesso à minha *venue* **para** poder gerir cada um .

US-5: Convidar novo utilizador para a minha *venue*

i. **Como um** utilizador de um espaço autenticado **quero** convidar um utilizador para a plataforma **para** poderem aceder e gerir a informação da minha *venue*. Quero inserir a seguinte informação:
   - Primeiro nome
   - Ultimo nome
   - *Email*

US-6: Reenviar convite para novo utilizador da minha *venue*

i. **Como um** utilizador de um espaço autenticado **quero** reenviar um convite a um novo utilizador **para** poder renovar o convite.

US-7: Ver lista de utilizadores com acesso à *venue*

i. **Como um** utilizador de um espaço autenticado **quero** ver a lista de utilizadores com acesso à minha *venue* **para** poder facilmente perceber que utilizadores têm acesso e que permissões têm. Quero ver a seguinte informação:
   - Nome
   - *Email*
   - Data do ultimo *login*

US-8: Ordenar lista de utilizadores com acesso à *venue*

i. **Como um** utilizador de um espaço autenticado **quero** ordenar a lista de utilizadores com acesso à minha *venue* **para** poder extrair facilmente qualquer informação necessária.

US-9: Procurar utilizadores com acesso à *venue*

i. **Como um** utilizador de um espaço autenticado **quero** procurar utilizadores com acesso à minha *venue* **para** poder extrair facilmente qualquer informação necessária. Que procurar por:
   - Nome
   - *Email*

US-10: Remover utilizador da *venue*

i. **Como um** utilizador de um espaço autenticado **quero** remover um utilizador com acesso à minha *venue* **para** apagar um utilizador que não deveria ter acesso à mesma.

3. **Dados do utilizador**

ES-3: **Como um** utilizador de um espaço autenticado **quero** gerir os meus dados pessoais **para** gerir e mudar os mesmos.

US-11: Mudar a *password*

i. **Como um** utilizador de um espaço autenticado **quero** mudar a minha *password* **para** a configurar de forma a ser segura e fácil de decorar.

US-12: Mudar os dados pessoais

i. **Como um** utilizador de um espaço autenticado **quero** mudar os meus dados pessoais **para** serem correctos. Quero editar as seguintes informações:
   - Primeiro nome
   - Ultimo nome
   - *Email*
   - Imagem de perfil

US-13: Ver a minha lista de *venues*

i. **Como um** utilizador de um espaço autenticado **quero** listar as minhas *venues* **para** poder facilmente aceder às suas informações e detalhes. Quero ter acesso a cada módulo de gestão da *venue* interagindo com o nome da mesma na listagem.

US-14: Contactar o *helpdesk*

i. **Como um** utilizador de um espaço autenticado **quero** pedir ajuda **para** ser assistido em qualquer problema relativo à plataforma.

4. **Definições da *venue***

ES-4: **Como um** utilizador de um espaço autenticado **quero** ver e editar a minha *venue* no Sala-Z **para** a plataforma poder mostrar informação atualizada.

US-15: Ver e mudar os dados de uma *venue*

i. **Como um** utilizador de um espaço autenticado **quero** ver e editar a(s) minha(s) *venue*(s) **para** a plataforma mostrar a informação atualizada sobre a(s) mesma(s). Quer poder mudar os seguintes pontos:
   - Nome*
   - Descrição*
   - Tipo de espaço*
   - Tipo de eventos*
   - Morada completa*
   - Região*
   - Contactos
   - *Website*
   - Redes sociais
   - Horário
   - Fotografias

   (*campo obrigatório)

ES-5 **Como um** utilizador de um espaço autenticado **quero** gerir os meus recursos **para** a plataforma mostrar informação atualizada.

US-16: Adicionar recursos a uma *venue*

i. **Como um** utilizador de um espaço autenticado **quero** adicionar todos os recursos disponíveis na minha *venue* **para** que qualquer pessoa interessada em atuar possa ver a listagem de recursos disponíveis. Quero adicionar pelo menos:
   - Nome*
   - Tipo*
   - Descrição
   - Políticas de uso
   - Fotografia

   (*campo obrigatório)

US-17: Alterar recursos de uma *venue*

i. **Como um** utilizador de um espaço autenticado **quero** editar todos os recursos disponíveis na minha *venue* **para** que qualquer pessoa interessada em atuar possa ver a listagem de recursos disponíveis atualizada. Quero poder mudar pelo menos os seguintes pontos:
   - Nome*
   - Tipo*
   - Descrição
   - Políticas de uso
   - Fotografia

   (*campo obrigatório)

US-18: Listar recursos de uma *venue*

i. **Como um** utilizador de um espaço autenticado **quero** ver um listagem de todos os recursos disponíveis na minha *venue* **para** poder facilmente entender todos os recursos disponíveis. Quero poder ver os seguintes pontos:
   - Nome*
   - Tipo*

   (*campo obrigatório)

US-19: Eliminar recurso de uma *venue*

i. **Como um** utilizador de um espaço autenticado **quero** poder eliminar qualquer recursos listado na minha *venue* **para** remover qualquer recurso obsoleto. Devo também receber um aviso antes de eliminar o recurso.

5. **Gestão do artista**

ES-6: **Como um** utilizador de um espaço autenticado **quero** aceder à listagem de artistas seguidos pela minha *venue*

US-20: Ver lista de artistas

i. **Como um** utilizador de um espaço autenticado **quero** ver a minha lista de artistas **para** poder ver a lista de artistas que contactei, artistas que contactaram a *venue* ou artistas que sigo para pontecial futuro contacto. Quero ver pelo menos:
   - Artista*
   - Tipo de música*
   - Região*
   - Dono da conta*
   - Status* (iniciado, confirmado, realizado, fechado, abandonado)
   - Número de vezes que atuou na minha *venue**
   - Data da ultima atuação na minha *venue*

   (*campo obrigatório)

US-21: Marcar artista com favorito na minha lista de artistas

i. **Como um** utilizador de um espaço autenticado **quero** poder marcar como favorito um artista na minha lista de artistas **para** poder filtrar a minha lista de artistas e alcançar os artistas favoritos mais rapidamente

US-22: Filtrar a lista de artistas

i. **Como um** utilizador de um espaço autenticado **quero** poder filtrar a minha lista de artistas **para** facilmente extrair qualquer informação que necessite. Quero filtrar por:
   - Nome
   - Região
   - Tipo de música
   - Status
   - Data do ultimo evento
   - Numero de atuações
   - Qualquer combinação dos anteriores

US-23: Ordenar a minha lista de artistas.

i. **Como um** utilizador de um espaço autenticado **quero** poder ordenar a minha lista de artistas **para** poder mais facilmente extrair qualquer informação que necessite. Quero poder ordenar por:
   - Nome
   - Região
   - Tipo de música
   - Status
   - Data do ultimo evento
   - Numero de atuações

US-24: Procurar na minha lista de artistas.

i. **Como um** utilizador de um espaço autenticado **quero** poder procurar na minha lista de artistas **para** poder facilmente extrair qualquer informação necessária. Quero poder filtrar por:
   - Nome
   - Região
   - Tipo de música
   - Status

US-25: Exportar a minha lista de artistas

i. **Como um** utilizador de um espaço autenticado **quero** poder exporta a minha lista de artistas em .csv **para** poder trabalhar sobre os mesmos *offline*.

ES-7 **Como um** utilizador de um espaço autenticado **quero** ter acesso à informação de um artista que a minha *venue* segue **para** poder facilmente extrair qualquer informação necessária.

US-26: Adicionar novo artista à minha lista de artistas.

i. **Como um** utilizador de um espaço autenticado **quero** adicionar um novo artista à minha lista de artistas **para** ter uma lista de todos os artistas que contactei, artistas que contactaram a minha *venue* ou artista que segui para possíveis futuras atuações. Quero poder criar um artista com os seguintes pontos:
   - Artista*
   - Tipo de música*
   - Região*
   - Dono da conta*
   - Status* (iniciado, confirmado, realizado, fechado, abandonado)
   - Número de vezes que atuou na minha *venue**
   - Lista de eventos passados na minha *venue*
   - Contactos
   - *Website*
   - Redes sociais
   - *Links* para *website*s com informações adicionais (e.g: *Discogs*, *Bandcamp)*
   - Fotografias
   - *Samples*

- Vídeos

  (*campo obrigatório)

US-27: Ver um artista listado na minha lista de artistas

i. **Como um** utilizador de um espaço autenticado **quero** poder visualizar um artista da minha lista de artistas **para** poder aceder aos detalhes do mesmo

US-28: Editar os detalhes de um artista listado na minha lista de artistas

i. **Como um** utilizador de um espaço autenticado **quero** poder mudar os detalhes de uma artistas **para** ter uma compreensão completa dos artistas que contactei, contactaram a minha *venue* ou artistas que segui.

US-29: Remover um artista

i. **Como um** utilizador de um espaço autenticado **quero** poder remover um artista da minha lista de artistas **para** remover qualquer artista obsoleto da mesma

6. **Gestão de eventos**

ES-8: **Como um** utilizador de um espaço autenticado **quero** ter acesso à informação dos eventos da minha *venue* **para** poder facilmente aceder a qualquer informação sobre os mesmos.

US-30: Criar evento

i. **Como um** utilizador de um espaço autenticado **quero** quero começar um novo evento **para** poder ter uma listagem de todos os eventos que estão a ser negociados na minha *venue*. Quer adicionar a seguinte informação:
   - Artista(s)*
   - Data(s) do evento*
   - Dono da conta*
   - Status (iniciado, confirmado, realizado, fechado, abandonado)
   - Recursos necessários
   - Lista de tarefas
   - Receita dos bilhetes
   - Outra receita
   - audiência

     (*campo obrigatório)

US-31: Editar um evento

i. **Como um** utilizador de um espaço autenticado **quero** editar qualquer evento **para** ter informação sobre o mesmo atualizada.

US-32: Ver evento

i. **Como um** utilizador de um espaço autenticado **quero** poder ver toda a informação sobre um evento **para** ter facilmente acesso à mesma.

US-33: Apagar evento

i. **Como um** utilizador de um espaço autenticado **quero** poder apagar um evento **para** remover qualquer evento obsoleto.

ES-9 **Como um** utilizador de um espaço autenticado **quero** ter uma lista de eventos **para** ter acesso fácil a qualquer evento negociado ou em negociação.

US-34: Ver lista de evento no calendário

i. **Como um** utilizador de um espaço autenticado **quero** ver um evento no calendário dos meus eventos **para** poder ver todos os eventos negociados.

US-35: Gerir lista de eventos

i. **Como um** utilizador de um espaço autenticado **quero** poder ver a minha lista de eventos **para** poder gerir a mesma

US-36: Filtrar lista de eventos

i. **Como um** utilizador de um espaço autenticado **quero** filtrar a minha lista de eventos **para** poder mais facilmente extrair qualquer informação. Quero poder filtrar por:

- Nome
- Data
- Status
- Tipo de música
- Dono da conta
- Qualquer combinação dos a cima listados

US-37: Ordenar lista de eventos

i. **Como um** utilizador de um espaço autenticado **quero** poder ordenar a minha lista de eventos **para** poder mais facilmente extrair informação necessária. Quero poder ordenar por:
   - Nome
   - Data
   - Status
   - Tipo de música
   - Dono da conta

US-38: Procurar na lista de eventos

i. **Como um** utilizador de um espaço autenticado **quero** poder procurar por um evento **para** ter acesso ao mesmo mais rapidamente. Quero poder procurar por:
   - Nome
   - Data
   - Status
   - Tipo de música
   - Dono da conta
   - Artistas

US-39: Exportar lista de eventos

i. **Como um** utilizador de um espaço autenticado **quero** poder extrair a minha lista de eventos em .csv **para** poder trabalhar sobre a mesma num ambiente *offline* ou poder partilhar mais facilmente.

ES-10: **Como um** utilizador de um espaço autenticado **quero** gerir todas as tarefas associadas a cada evento **para** poder mais facilmente perceber o estado de cada uma.

US-40: Criar tarefa

i. **Como um** utilizador de um espaço autenticado **quero** adicionar uma nova tarefa **para** poder mais facilmente gerir o meu evento. Quero adicionar os seguintes pontos:
   - Titulo*
   - Descrição
   - Dono da tarefa
   - Status
   - Data
   - Recursos necessários
   - Checklist
   - Artista associado

US-41: Editar tarefa

i. **Como um** utilizador de um espaço autenticado **quero** poder editar uma tarefa **para** ter a informação atualizada. Quero poder alterar os seguintes pontos:
   - Titulo*
   - Descrição
   - Dono da tarefa
   - Status
   - Data
   - Recursos necessários

- Checklist
- Artista associado
- Etiquetas

US-42: Ver tarefa

i. **Como um** utilizador de um espaço autenticado **quero** ver toda a informação sobre uma tarefa **para** obter qualquer informação necessária. Quero ver os seguintes pontos:
   - Titulo
   - Descrição
   - Dono da tarefa
   - Status
   - Data
   - Recursos necessários
   - Checklist
   - Artista associado

US-43: Apagar tarefa

i. **Como um** utilizador de um espaço autenticado **quero** poder apagar uma tarefa **para** eliminar qualquer tarefa obsoleta.

ES-11: **Como um** utilizador de um espaço autenticado **quero** ter acesso a uma listagem de tarefas **para** poder gerir as mesmas.

US-44: Ver lista de tarefas do evento

i. **Como um** utilizador de um espaço autenticado **quero** ver a lista de tarefas **para** poder visualizar o estado de cada uma.

US-45: Filtrar lista de tarefas do evento

i. **Como um** utilizador de um espaço autenticado **quero** filtrar a minha lista de tarefas **para** aceder mais rapidamente a uma tarefa. Quero filtrar pelos seguintes pontos:
   - Data
   - Status
   - Dono da tarefa
   - Etiqueta
   - Artista associado

US-46: Ordenar lista de tarefas do evento

i. **Como um** utilizador de um espaço autenticado **quero** poder ordenar a minha lista de tarefas **para** poder aceder mais facilmente a qualquer tarefa. Quero poder ordenar por:
   - Data
   - Status
   - Dono da tarefa
   - Etiqueta
   - Artista associado

US-47: Procurar na lista de tarefas do evento

i. **Como um** utilizador de um espaço autenticado **quero** procurar por uma tarefa na minha lista de tarefas **para** poder mais facilmente encontra uma tarefa específica. Quero procurar por:
   - Data
   - Dono da tarefa
   - Etiqueta
   - Artista associado

US-48: Exportar a minha lista de tarefas

i. **Como um** utilizador de um espaço autenticado **quero** exportar a minha lista de tarefas em .csv **para** poder trabalhar num ambiente *offline* ou poder partilhar com outro utilizador.

ES-12: **Como um** utilizador de um espaço autenticado **quero** gerir todos os recursos associados a um evento **para** poder mais facilmente perceber o estado de cada evento

US-49: Adicionar um recurso a um evento

i. **Como um** utilizador de um espaço autenticado **quero** adicionar um recurso disponível na minha *venue* a um evento **para** poder gerir os recursos requisitados.

US-50: Editar um recurso de um evento

i. **Como um** utilizador de um espaço autenticado **quero** editar um recurso associado a um evento **para** mudar o seu tempo de requisito.

US-51: Remover um recurso de um evento

i. **Como um** utilizador de um espaço autenticado **quero** desassociar um recurso a um evento **para** ficar disponível.

7. **Listagem de recursos**

ES-13: **Como um** utilizador de um espaço autenticado **quero** ter acesso à listagem de uso dos recursos da *venue* **para** poder ter informação sobre cada recurso

US-52: Ver no calendário recursos reservados

i. **Como um** utilizador de um espaço autenticado **quero** ver num calendário a minha lista de recursos reservados **para** gerir os mesmos

US-53: Ver lista de recursos reservados

i. **Como um** utilizador de um espaço autenticado **quero** listar os meus recursos reservados **para** saber que recursos tão reservados e para que propósito.

US-54: Filtrar lista de recursos reservados

i. **Como um** utilizador de um espaço autenticado **quero** filtrar a minha lista de recursos reservados **para** extrair qualquer informação necessária. Quero filtrar pelos seguintes pontos:
   - Nome
   - Data(s)
   - Evento

US-55: Ordenar lista de recursos reservados

i. **Como um** utilizador de um espaço autenticado **quero** ordenar a minha lista de recursos reservados **para** extrair qualquer informação necessária. Quero ordenar pelos seguintes pontos:
   - Nome
   - Data(s)
   - Evento

US-56: Procurar na lista de recursos reservados

i. **Como um** utilizador de um espaço autenticado **quero** procurar na minha lista de recursos reservados **para** extrair qualquer informação necessária. Quero ordenar por:
   - Nome
   - Evento

US-57: Exportar lista de recursos reservados

i. **Como um** utilizador de um espaço autenticado **quero** exportar a minha lista de recursos reservados em .csv **para** poder trabalhar num ambiente *offline* ou para poder partilhar com outra pessoa.

8. *Dashboard*

ES-14: **Como um** utilizador de um espaço autenticado **quero** ver os melhores eventos **para** perceber quais os eventos que correram melhor para cada uma das métricas.

US-58: Ver melhores eventos por receita dos bilhetes

i. **Como um** utilizador de um espaço autenticado **quero** ver os melhores eventos por receita dos bilhetes **para** saber quais foram os eventos mais lucrativos.

US-59: Ver melhores eventos por outra receita

i. **Como um** utilizador de um espaço autenticado **quero** ver os melhores eventos por outras receitas **para** saber quais foram os eventos mais lucrativos.

US-60: Ver melhores eventos por receita total

i. **Como um** utilizador de um espaço autenticado **quero** ver os melhores eventos por receita total**para** saber quais foram os eventos mais lucrativos.

US-61: Ver melhores eventos por audiência

i. **Como um** utilizador de um espaço autenticado **quero** ver os eventos com mais audiência **para** saber quais foram os eventos mais populares.

US-62: Filtrar melhores eventos

i. **Como um** utilizador de um espaço autenticado **quero** filtrar os melhores eventos **para** saber quais foram os melhores eventos baseados em:
   - Período de tempo
   - Artista
   - Receita dos bilhetes em média
   - Outra receita em média
   - Receita total em média
   - Popularidade do evento
   - Audiência média

ES-15: **Como um** utilizador de um espaço autenticado **quero** ver os melhores artistas **para** perceber quais os melhores artistas para cada uma das métricas.

US-63: Ver melhores artistas por receita dos bilhetes

i. **Como um** utilizador de um espaço autenticado **quero** ver o artistas que gerou mais receita por venda de bilhetes **para** saber quais foram os artistas mais lucrativos.

US-64: Ver melhores artistas por outra receita

i. **Como um** utilizador de um espaço autenticado **quero** ver o artistas que gerou mais receita por outro motivo **para** saber quais foram os artistas mais lucrativos.

US-65: Ver melhores artistas por receita total

i. **Como um** utilizador de um espaço autenticado **quero** ver o artistas que gerou mais receita **para** saber quais foram os eventos mais lucrativos.

US-66: Ver melhores artistas por audiência

i. **Como um** utilizador de um espaço autenticado **quero** ver os artistas que tiveram mais audiência **para** saber quais foram os artistas mais populares.

US-67: Filtrar melhores artistas

i. **Como um** utilizador de um espaço autenticado **quero** filtrar os melhores artistas **para** saber quais foram os melhores artistas baseados em:
   - Período de tempo
   - Tipo de música
   - Receita dos bilhetes em média
   - Outra receita em média
   - Receita total em média
   - Popularidade do artista
   - Número de eventos

ES-16: **Como um** utilizador de um espaço autenticado **quero** ver os melhores tipo musicais **para** perceber quais os melhores para cada uma das métricas.

US-68: Ver melhores tipos musicais por receita dos bilhetes

i. **Como um** utilizador de um espaço autenticado **quero** ver o tipo musical que gerou mais receita por venda de bilhetes **para** saber quais foram os mais lucrativos.

US-69: Ver melhore tipos musicais por outra receita

i. **Como um** utilizador de um espaço autenticado **quero** ver o tipo musical que gerou mais receita por outro motivo **para** saber quais foram os mais lucrativos.

US-70: Ver melhores tipos musicais por receita total

i. **Como um** utilizador de um espaço autenticado **quero** ver o tipo musical que gerou mais receita **para** saber quais foram os mais lucrativos.

US-71: Ver melhores tipos musicais por audiência

i. **Como um** utilizador de um espaço autenticado **quero** ver os tipos musicais que tiveram mais audiência **para** saber quais foram os mais populares.

US-72: Filtrar melhores tipos musicais

i. **Como um** utilizador de um espaço autenticado **quero** filtrar os melhores tipo musicais **para** saber quais foram os melhores baseados em:
   - Período de tempo
   - Artista
   - Receita dos bilhetes em média
   - Outra receita em média
   - Receita total em média
   - Popularidade do artista
   - Número de eventos

ES-17: **Como um** utilizador de um espaço autenticado **quero** ver as métricas chave da minha *venue* **para** poder gerir melhor a mesma

US-73: Receita por mês

i. **Como um** utilizador de um espaço autenticado **quero** ver a minha receita por mês **para** saber por cada tipo de receita o meu agregado mensal

US-74: Receita acumulada anual

(a) **Como um** utilizador de um espaço autenticado **quero** ver a minha receita acumulada por ano **para** saber por cada tipo de receita o meu agregado anual

US-75: Filtrar receita acumulada mensal ou anual

i. **Como um** utilizador de um espaço autenticado **quero** filtrar a minha receita **para** ver o meu agregado e/ou receita acumulada depois de filtrar os seguintes campos:
   - Período de tempo
   - Tipo de música
   - Receita dos bilhetes
   - Outra receita
   - Receita total
   - Audiência média

US-76: Ver audiência

i. **Como um** utilizador de um espaço autenticado **quero** ver a audiência por mês **para** saber o meu agregado mensal.

US-77: Ver audiência anual

i. **Como um** utilizador de um espaço autenticado **quero** ver a audiência por ano **para** saber o meu agregado anual.

US-78: Filtrar audiência

i. **Como um** utilizador de um espaço autenticado **quero** filtrar a audiência **para** ver o meu agregado e/ou audiência acumulada depois de filtrar os seguintes campos:
   - Período de tempo
   - Tipo de música
   - Receita dos bilhetes
   - Outra receita
   - Receita total
   - Audiência média

US-79: Número de eventos por mês

i. **Como um** utilizador de um espaço autenticado **quero** ver o número de eventos por mês **para** saber o agregado mensal

US-80: Número de eventos acumulados por ano

i. **Como um** utilizador de um espaço autenticado **quero** ver o número de eventos por ano **para** saber o meu agregado anual.

US-81: Filtrar número de eventos

i. **Como um** utilizador de um espaço autenticado **quero** filtrar o número de eventos **para** ver o meu agregado e/ou número de eventos acumulado depois de filtrar os seguintes campos:

- Período de tempo
- Tipo de música
- Receita dos bilhetes
- Outra receita
- Receita total
- Audiência média

ES-18: **Como um** utilizador de um espaço autenticado **quero** ver num *dashboard* o calendário **para** ver facilmente os eventos da minha *venue*

Módulo do *Backoffice*

1. **Gestão da autenticação - Administrador do Sala-Z**

ES-19: **Como um** administrador do Sala-Z **quero** garantir que o Sala-Z e o seu *backoffice* apenas é utilizador por utilizadores autenticados **para** assegurar que todas as funcionalidades da plataforma apenas são acedidas por utilizadores autorizados.

US-82: *Login*

i. **Como um** administrador do Sala-Z **quero** entra no Sala-Z **para** usar as funcionalidades da plataforma.

US-83: Terminar Sessão

i. **Como um** administrador do Sala-Z **quero** terminar a sessão **para** que outros utilizadores que têm acesso ao meu aparelho não possam utilizar a plataforma em meu nome.

US-84: Recuperação da *password*

i. **Como um** administrador do Sala-Z **quero** pedir uma recuperação de *password* **para** poder aceder à minha conta no caso de me esquecer da *password*.

US-85: Primeiro *Login*

i. **Como um** administrador do Sala-Z **quero** completar o meu primeiro *Login* **para** poder aceitar um convite de acesso ao Sala-Z e, consequentemente, mudar a minha *password* definida pelo sistema

2. Gestão de *venues* - Administrador do Sala-Z

ES-20: **Como um** administrador do Sala-Z autenticado **quero** gerir as contas de *venues* do Sala-Z **para** gerir cada uma e utilizadores associados

US-86: Criar nova *venue*

i. **Como um** administrador do Sala-Z autenticado **quero** criar uma nova *venue* **para** adicionar uma nova *venue* e associar um administrador. A *venue* terá os seguintes campos:

- Nome*
- Curta descrição
- Tipo de espaço
- Tipo de eventos

- Morada completa
- Região
- Contacto
- *Website*
- Redes sociais
- Horário
- Fotografias

US-87: Associar utilizador a uma *venue*

i. **Como um** administrador do Sala-Z autenticado **quero** associar um utilizador a uma *venue* **para** poder convidar um utilizador como administrador da *venue*. Quero inserir os seguintes campos:
   - Primeiro nome*
   - Ultimo nome*
   - *Email**

US-88: Reenviar um convite a um novo utilizador de um espaço

i. **Como um** administrador do Sala-Z autenticado **quero** reenviar um convite a um utilizador **para** renovar o mesmo.

US-89: Eliminar um utilizador

i. **Como um** administrador do Sala-Z autenticado **quero** eliminar um utilizador **para** retirar o acesso do mesmo ao Sala-Z

US-90: Listar *venues*

i. **Como um** administrador do Sala-Z autenticado **quero** listar todas as *venues* **para** poder visualizar cada uma e gerir as mesmas. Quero ver a seguinte informação:
   - Nome
   - Tipo de espaço
   - Tipo de evento
   - Região

US-91: Listar utilizadores

i. **Como um** administrador do Sala-Z autenticado **quero** listar todos os utilizadores **para** poder visualizar cada uma e gerir os mesmas. Quero ver a seguinte informação:
   - Nome
   - *venues* associadas
   - *Email*
   - Ultimo *login*

US-92: Editar *venue*

i. **Como um** administrador do Sala-Z autenticado **quero** editar uma *venue* **para** poder atualizar as seguintes informações:
   - Nome*
   - Curta descrição
   - Tipo de espaço
   - Tipo de eventos
   - Morada completa
   - Região
   - Contacto
   - *Website*
   - Redes sociais
   - Horário
   - Fotografias

B Apêndice B

B.1 Plano de testes

User story	ID	Cenário	Descrição	Passos	Resultado esperado	Resultado obtido	Estado (P/F)
US-001	1	Login válido como utilizador de uma *venue*	Verificar se um utilizador com credenciais válidas consegue iniciar a sessão	- Abrir a aplicação - Inserir email correto - Inserir palavra-passe correta - Submeter	Entrar no *frontoffice*	Entrar no *frontoffice*	P
	2	Iniciar sessão com email inválido	Verificar se um email inválido produz uma mensagem de erro	- Abrir a aplicação - Inserir email incorreto - Inserir palavra-passe correta - Submeter	Mensagem de erro	Mensagem de erro	P
	3	Iniciair sessão com palavra-passe inválida	Verificar se uma *password* inválida produz uma mensagem de erro	- Abrir a aplicação - Inserir email correto - Inserir palavra-passe incorreta - Submeter	Mensagem de erro	Mensagem de erro	P
	4	Iniciar sessão e não manter sessão iniciada	Verificar se, após sair da aplicação, a sessão é terminada após não ter selecionado a opção "Manter sessão iniciada"	- Abrir a aplicação - Inserir credenciais válidas - Submeter - Fechar browser - Abrir aplicação	Página de *Login*	Página de *Login*	P
	5	Iniciar sessão e manter sessão iniciada	Verificar se, após sair da aplicação, a sessão é mantida após ter selecionado a opção "Manter sessão iniciada"	- Abrir a aplicação - Inserir credenciais válidas - Selecionar opção de manter sessão - Submeter - Fechar browser - Abrir aplicação	Página principal	Página principal	P
US-002	6	Terminar sessão	Verificar se sessão é terminada após clicar *logout*	- Entrar na aplicação - Clicar no icon de *logout* - Voltar uma página atrás	Sair para o *Login* e não ser possível voltar ao menu	Sair para o *Login* e não ser possível voltar ao menu	P
US-003	7	Recuperar palavra-passe com email correto e inserir código errado	Verificar se, ao inserir um código inválido, é apresentada uma mensagem de erro	- Entrar na aplicação - Recuperar palavra-passe - Inserir email correto - Inserir código inválido e palavra-passe válida	Mensagem de erro	Mensagem de erro	P
	8	Recuperar palavra-passe com email correto e inserir código correto	Verificar se, ao inserir um código válido, a palavra-passe é alterada	- Entrar na aplicação - Recuperar palavra-passe - Inserir email correto - Inserir código válido e palavra-passe válida	Palavra-passe alterada com sucesso	Palavra-passe alterada com sucesso	P
US-004	9	Não mudar palavra-passe no primeiro *Login*	Verificar se o utilizador consegue ultrapassar o passo de alteração da palavra passe sem alterar a palavra passe	- Inserir credenciais de novo utilizador - Voltar atrás no ecrã de primeiro Login - Inserir as credenciais de novo	Ecrã de primeiro *Login*	Ecrã de primeiro *Login*	P
	10	Mudar palavra-passe no primeiro *Login*	Verificar se o utilizador consegue efectuar o *Login* após mudar a palavra-passe	- Inserir credenciais de novo utilizador - Mudar a palavra-passe - Entrar na aplicação com palavra-passe nova	Página principal	Página principal	P
US-005	11	Adicionar um novo utilizador a uma *Venue*	Verificar se um utilizador com uma *Venue* pode adicionar um novo utilizador	- Navegar até edição de uma *venue* - Adicionar colaborador com email, primeiro norme e ultimo nome válido.	Convite enviado para utilizador	Convite enviado para utilizador	P
US-006	12	Reenviar um convite ao utilizador	Não desenvolvido	-	-	-	-
US-007	13	Listar todos os membros de um espaço associado	Verificar se um utilizador com uma *Venue* associada pode visualizar todos os outros membros	- Navegar até edição de uma *venue* - Adicionar um novo colaborador. - Ver detalhes da *Venue*	*Venue* com lista de utilizadores, incluindo o recentemente adicionado e o próprio	*Venue* com lista de utilizadores, incluindo o recentemente adicionado e o próprio	P
US-008	14	Ordenar lista de utilizadores	Não desenvolvido	-	-	-	-
US-009	15	Procurar na lista de utilizadores	Não desenvolvido	-	-	-	-
US-010	16	Remover um membro de um espaço associado	Verificar se um utilizador com uma *Venue* associada pode remover um outro membro	- Navegar até edição de uma *venue* - Remover um novo colaborador. - Ver detalhes da *Venue*	*Venue* com lista de utilizadores, excluindo o utilizador removido	*Venue* com lista de utilizadores, excluindo o utilizador removido	P
US-011	17	Alterar a palavra-passe pessoal para uma palavra-passe que não cumpra os requisitos	Verificar se um utilizador poderá mudar a sua palavra-passe para uma que não cumpra os requisitos	- Navegar até aos dados pessoais - Inserir palavra-passe atual - Inserir palavra-passe que não cumpra os requisitos	Mensagem de erro	Mensagem de erro	P
	18	Alterar a palavra-passe pessoal	Verificar se um utilizador poderá mudar a sua palavra-passe e se pode efetuar *Login* com a mesma	- Navegar até aos dados pessoais - Inserir palavra-passe atual - Inserir nova palavra-passe - Terminar sessão - Efetuar o *Login* com a nova palavra-passe	Página principal	Página principal	P
US-012	19	Alterar os dados pessoais	Verificar se, ao serem alterados os dados pessoais, estes são guardados corretamente	- Navegar até aos dados pessoais - Alterar dados pessoais - Sair - Navegar até aos dados pessoais	Página com os dados pessoais alterados	Página com os dados pessoais alterados	P

US-013	20	Visualizar todas as *venues* pessoais associadas ao presente utilizador	Verificar se todas as *venues* associadas ao presente utilizador estão a ser mostradas corretamente	- Navegar até à página das *venues* pessoais.	Página com todas as *venues* pessoais	Página com todas as *venues* pessoais	P
US-014	21	Contactar o *helpdesk*	Não desenvolvido	-	-	-	-
US-015 Editar uma venue	22	Alterar qualquer detalhe de uma *venue* pessoal	Verificar se todas as alterações feitas a uma *venue* estão a ser corretamente alteradas	- Navegar até à página das *venues* pessoais. - Clicar numa *venue* - Editar todos os campos - Sair - Ver detalhes da mesma	Detalhes com a informação alterada corretamente	Detalhes com a informação alterada corretamente	P
US-016	23	Adicionar recurso a uma *venue*	Não desenvolvido	-	-	-	-
US-017	24	Mudar recursos de uma *venue*	Não desenvolvido	-	-	-	-
US-018	25	Listar recursos de uma *venue*	Não desenvolvido	-	-	-	-
US-019	26	Eliminar recurso de uma *venue*	Não desenvolvido	-	-	-	-
US-020	27	Listar artistas	Verificar se é possível listar todos os artistas da plataforma (desenvolvido parcialmente)	- Navegar até à página dos eventos. - Criar evento - Abrir *dropdown* dos artistas - Verificar listagem de artistas	*Dropdown* com todos os artistas da plataforma	*Dropdown* com todos os artistas da plataforma	P
US-021	28	Marcar artista como favorito	Não desenvolvido	-	-	-	-
US-022	29	Filtrar artistas	Não desenvolvido	-	-	-	-
US-023	30	Ordenar artistas	Não desenvolvido	-	-	-	-
US-024	31	Procurar artista	Não desenvolvido	-	-	-	-
US-025	32	Exportar lista de artistas	Não desenvolvido	-	-	-	-
US-026	33	Adicionar artista	Verificar se é possível adicionar novo artista (desenvolvido parcialmente)	- Navegar até à página dos eventos. - Criar evento - Adicionar um artista não existente na *dropdown* dos artistas - Concluir criação do evento - Ver detalhes do evento criado	Evento criado com artista criado	Evento criado com artista criado	P
US-027	34	Ver artista	Não desenvolvido	-	-	-	-
US-028	35	Editar artista	Não desenvolvido	-	-	-	-
US-029	36	Remover artista	Não desenvolvido	-	-	-	-
US-030	37	Criar um novo evento através do método não detalhado	Verificar se é possível criar o evento e se a informação é guardada corretamente	- Navegar até à página dos eventos. - Criar um evento - Inserir informações - Salvar	Evento criado com sucesso	Evento criado com sucesso	P
	38	Criar um novo evento através do método detalhado	Verificar se é possível criar o evento de forma detalhada e se a informação é guardada corretamente	- Navegar até à página dos eventos. - Criar um evento - Seguir para todas as informações - Criar evento - Salvar	Evento criado com sucesso	Evento criado com sucesso	P
US-031	39	Editar um evento	Verificar se é possível editar um evento e se toda a informação é guardada com sucesso	- Navegar até à página dos eventos. - Editar um evento - Inserir novas informações - Salvar	Evento alterado com sucesso	Evento alterado com sucesso	P
US-032	40	Ver informações sobre um evento	Verificar se é possível ver um evento da listagem de eventos	- Navegar até à página dos eventos. - Selecionar um evento e expandir	Evento detalhado	Evento detalhado	P
US-033	41	Eliminar um evento	Verificar se é possível eliminar um evento e se o mesmo é removido da listagem de eventos	- Navegar até à página dos eventos. - Selecionar um evento - Eliminar	Evento eliminado com sucesso	Evento eliminado com sucesso	P
US-034	42	Ver um evento no calendário	Verificar se é possível criar o evento e se a informação é guardada corretamente	- Navegar até à página dos eventos. - Criar um evento - Inserir informações - Salvar	Evento criado com sucesso	Evento criado com sucesso	P
US-035	43	Listar eventos	Verificar se é possível listar todos os eventos pertencentes a uma *venue* associada ao utilizador	- Navegar até à página dos eventos. - Ver lista de eventos	Eventos apresentados com sucesso	Eventos apresentados com sucesso	P
US-036	44	Filtrar eventos	Verificar se é possível filtrar a lista de eventos	- Navegar até à página dos eventos. - Filtrar lista de eventos	Eventos listados de acordo com a filtragem	Eventos listados de acordo com a filtragem	P
US-037	45	Ordenar lista de eventos	Não desenvolvido	-	-	-	-
US-038	46	Procurar evento	Verificar se é possível procurar evento por nome	- Navegar até à página dos eventos. - Pesquisar na barra de pesquisa	Eventos listados de acordo com a pesquisa	Eventos listados de acordo com a pesquisa	P
US-039	47	Exportar lista de eventos	Verificar se é possível exportar a lista de eventos como um .csv	- Navegar até à página dos eventos. - Clicar em "Exportar eventos"	Começar um download de um ficheiro .csv válido	Começar um download de um ficheiro .csv válido	P
US-040	47	Criar tarefa	Não desenvolvido	-	-	-	-

US-041	48	Editar tarefa	Não desenvolvido	-	-	-	-
US-042	49	Ver tarefa	Não desenvolvido	-	-	-	-
US-043	50	Apagar tarefa	Não desenvolvido	-	-	-	-
US-044	51	Ver lista de tarefas do evento	Não desenvolvido	-	-	-	-
US-045	52	Filtrar lista de tarefas do evento	Não desenvolvido	-	-	-	-
US-046	53	Ordenar lista de tarefas do evento	Não desenvolvido	-	-	-	-
US-047	54	Procurar na lista de tarefas do evento	Não desenvolvido	-	-	-	-
US-048	55	Exportar lista de tarefas pessoais	Não desenvolvido	-	-	-	-
US-049	56	Adicionar um recurso a um evento	Não desenvolvido	-	-	-	-
US-050	57	Editar um recurso de um evento	Não desenvolvido	-	-	-	-
US-051	58	Remover um recurso de um evento	Não desenvolvido	-	-	-	-
US-052	59	Ver no calendário recursos reservados	Não desenvolvido	-	-	-	-
US-053	60	Ver lista de recursos reservados	Não desenvolvido	-	-	-	-
US-054	61	Filtrar lista de recursos reservados	Não desenvolvido	-	-	-	-
US-055	62	Ordenar lista de recursos reservados	Não desenvolvido	-	-	-	-
US-056	63	Procurar na lista de recursos reservados	Não desenvolvido	-	-	-	-
US-057	64	Exportar lista de recursos reservados	Não desenvolvido	-	-	-	-
US-058	63	Ver melhores eventos por receita dos bilhetes	Não desenvolvido	-	-	-	-
US-059	64	Ver melhores eventos por outra receita	Verificar se é possível obter os melhores eventos por outra receita	- Navegar até à página da *dashboard*. - Selecionar eventos na *dropdown*. - Ordenar tabela das métricas por receita	1ª Linha pertence ao evento com mais receita acumulada	1ª Linha pertence ao evento com mais receita acumulada	P
US-060	65	Ver melhores eventos por receita total	Verificar se é possível obter os melhores eventos por receita total	- Navegar até à página da *dashboard*. - Selecionar eventos na dropdown. - Ordenar tabela das métricas por receita	1ª Linha pertence ao evento com mais receita acumulada	1ª Linha pertence ao evento com mais receita acumulada	P
US-061	66	Ver melhores eventos por audiência	Verificar se é possível obter os melhores eventos por audiência	- Navegar até à página da *dashboard*. - Selecionar eventos na dropdown. - Ordenar tabela das métricas por audiência	1ª Linha pertence ao evento com mais audiência	1ª Linha pertence ao evento com mais audiência	P
US-062	67	Filtrar melhores eventos	Verificar se é possível obter os melhores eventos	- Navegar até à página da *dashboard*. - Selecionar eventos na dropdown. - Ordenar tabela das métricas	1ª Linha pertence ao melhor evento de acordo com a ordenação realizada	1ª Linha pertence ao melhor evento de acordo com a ordenação realizada	P
US-063	68	Ver melhores artistas por receita dos bilhetes	Não desenvolvido	-	-	-	-
US-064	69	Ver melhores artistas por outra receita	Verificar se é possível obter os melhores artistas por outra receita	- Navegar até à página da *dashboard*. - Ordenar tabela das métricas por receita	1ª Linha pertence ao artista com mais receita acumulada	1ª Linha pertence ao artista com mais receita acumulada	P
US-065	70	Ver melhores artistas por receita total	Verificar se é possível obter os melhores artistas por receita total	- Navegar até à página da *dashboard*. - Ordenar tabela das métricas por receita	1ª Linha pertence ao artista com mais receita acumulada	1ª Linha pertence ao artista com mais receita acumulada	P
US-066	71	Ver melhores artistas por audiência	Verificar se é possível obter os melhores artistas por audiência	- Navegar até à página da *dashboard*. - Ordenar tabela das métricas por audiência	1ª Linha pertence ao artista com mais audiência	1ª Linha pertence ao artista com mais audiência	P
US-067	72	Filtrar melhores artistas	Verificar se é possível obter os melhores eventos	- Navegar até à página da *dashboard*. - Ordenar tabela das métricas	1ª Linha pertence ao melhor artista de acordo com a ordenação realizada	1ª Linha pertence ao melhor artista de acordo com a ordenação realizada	P
US-068	73	Ver melhores tipos musicais por receita dos bilhetes	Não desenvolvido	-	-	-	-
US-069	74	Ver melhores tipos musicais por receita dos bilhetes	Não desenvolvido	-	-	-	-
US-070	75	Ver melhores tipos musicais por receita total	Não desenvolvido	-	-	-	-
US-071	76	Ver melhores tipos musicais por audiência	Não desenvolvido	-	-	-	-

US-072	77	Filtrar melhores tipos musicais	Não desenvolvido	-	-	-	-
US-073	78	Receita por mês	Verificar se é possível visualizar a receita mensal	- Navegar até à página da *dashboard*. - Ver gráfico de receita mensal	É possível visualizar a receita do mês atual	É possível visualizar a receita do mês atual	P
US-074	79	Receita acumulada anual	Verificar se é possível visualizar a receita anual	- Navegar até à página da *dashboard*. - Ver gráfico de receita anual	É possível visualizar a receita do ano atual	É possível visualizar a receita do ano atual	P
US-075	80	Filtrar receita acumulada	Não desenvolvido	-	-	-	-
US-076	81	Ver audiência mensal	Verificar se é possível visualizar a audiência mensal	- Navegar até à página da *dashboard*. - Ver gráfico de audiência	Número de audiência mensal	Número de audiência mensal	P
US-077	82	Ver audiência anual	Verificar se é possível visualizar a audiência anual	- Navegar até à página da *dashboard*. - Ordenar tabela das métricas	Número de audiência anual	Número de audiência anual	P
US-078	83	Filtrar audiência	Não desenvolvido	-	-	-	-
US-079	84	Número de eventos por mês	Verificar se é possível verificar o número de eventos mensal	- Navegar até '*Dashboard*' - Visualizar número de eventos por mês	Número de eventos mensal	Número de eventos mensal	P
US-080	85	Número de eventos acumulados por ano	Verificar se é possível verificar o número de eventos anual	- Navegar até '*Dashboard*' - Visualizar número de eventos por ano	Número de eventos anual	Número de eventos anual	P
US-081	86	Filtrar número de eventos	Não desenvolvido	-	-	-	-
US-082	84	Login válido como administrador	Verificar se um administrador com credenciais válidas consegue iniciar a sessão	- Abrir a aplicação - Inserir email correto - Inserir palavra-passe correta - Submeter	Entrar no *frontoffice*	Entrar no *frontoffice*	P
US-083	85	Terminar sessão como administrador	Verificar se sessão é terminada após clicar *logout*	- Entrar na aplicação - Clicar no icon de *logout* - Voltar uma página atrás	Sair para o *Login* e não ser possível voltar ao menu	Sair para o *Login* e não ser possível voltar ao menu	P
US-084	86	Recuperar palavra-passe como administrador	Verificar se, ao inserir um código válido, a palavra-passe é alterada	- Entrar na aplicação - Recuperar palavra-passe - Inserir email correto - Inserir código válido e palavra-passe válida	Palavra-passe alterada com sucesso	Palavra-passe alterada com sucesso	P
US-085	87	Primeiro *Login* como administrador	Verificar se o utilizador consegue efectuar o *Login* após mudar a palavra-passe	- Inserir credenciais de novo utilizador - Mudar a palavra-passe - Entrar na aplicação com palavra-passe nova	Página principal	Página principal	P
US-086	88	Criar uma *Venue*	Verificar se um administrador consegue criar uma *venue*	- Navegar até listagem de *venues* - Cliclar em 'Adicionar *venue*' - Preencher formulário	Página com listagem de *venues* com a *venue* criada	Página com listagem de *venues* com a *venue* criada	P
US-087	89	Editar um *Venue*	Verificar se um administrador consegue editar uma *venue*	- Navegar até listagem de *venues* - Editar um *venue* - Mudar qualquer campo	Detalhes da *venue* alterados	Detalhes da *venue* alterados	P
US-088	90	Eliminar uma *Venue*	Verificar se um administrador consegue eliminar uma *venue*	- Navegar até listagem de *venues* - Eliminar uma *venue*	Página com listagem de *venues* sem a *venue* eliminada	Página com listagem de *venues* sem a *venue* eliminada	P
US-089	91	Associar novo utilizador a uma *venue*	Verificar se um administrador consegue associar um novo utilizador a uma *venue*	- Navegar até listagem de utilizadores - Clicar em 'Adicionar utilizador' - Preencher formulário - Adicionar pelo menos uma *venue*	Página com listagem de utilizadores com utilizador criado e com a(s) *venue(s) associada(s)*	Página com listagem de utilizadores com utilizador criado e com a(s) *venue(s) associada(s)*	P
	92	Associar utilizador existente a uma *venue*	Verificar se um administrador consegue associar um utilizador existente a uma *venue*	- Navegar até listagem de utilizadores - Editar utilizador - Adicionar pelo menos uma *venue*	Página com listagem de utilizadores com utilizador editado e com a (s) *venue(s) associada(s)*	Página com listagem de utilizadores com utilizador editado e com a(s) *venue(s) associada(s)*	P
US-090	93	Reenviar convite a utilizador	Verificar se um administrador consegue reenviar um convite a um utilizador	- Navegar até listagem de utilizadores - Reenviar convite a um utilizador	Email enviado ao utilizador		
US-091	94	Eliminiar utilizador	Verificar se um administrador consegue eliminar um utilizador	- Navegar até listagem de utilizadores - Eliminar um utilizador	Listagem de utilizadores sem o utilizador eliminado	Listagem de utilizadores sem o utilizador eliminado	P
US-092	95	Listar *Venues*	Verificar se um administrador consegue listar todas as *venues*	- Navegar até listagem de *venues*	Listagem de venues apresentada corretamente	Listagem de venues apresentada corretamente	P
US-093	96	Listar utilizadores	Verificar se um administrador consegue listar todos os utilizadores	- Navegar até listagem de utilizadores	Listagem de utilizadores apresentada corretamente	Listagem de utilizadores apresentada corretamente	P