Mestrado em Engenharia Informática

Estágio
Relatório final

Desenvolvimento de *software* em aplicações *Web* em *Java*

Case study: Time Link

Tiago Barradas Ceia

tbceia@student.dei.uc.pt

Orientadores

DEI | FCTUC

Bruno Cabral

iClio | History for the new Media

Alexandre Pinto

Joaquim Ramos de Carvalho

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia
Universidade de Coimbra
Pólo II, Pinhal de Marrocos, 3030 – 290 Coimbra
+351 239 790 000 | info@dei.uc.pt

iClio, LDA
History for the New Media
Instituto Pedro Nunes, Rua Pedro Nunes, 3030 – 199 Coimbra
+351 910 013 636 | geral@iclio.pt

Candidato
Nome: Tiago Barradas Ceia
Número de Estudante: 2006125564
Contacto: tbceia@student.dei.uc.pt

Orientador | DEI
Nome: Bruno Cabral
Contacto: bcabral@dei.uc.pt

Orientadores | iClio
Nome: Alexandre Pinto
Contacto: alexandrepinto@iclio.net

Nome: Joaquim Ramos de Carvalho
Contacto: joaquimcarvalho@iclio.net

Resumo

O *Time Link* é uma aplicação *Web* utilizada em recriação de comunidades históricas, tendo obtido vários resultados positivos no que toca a estudos nesta área por meio de investigação académica. Também conhecido pela sigla *MHK* (*Micro-History with Kleio*) este projeto resulta do trabalho desenvolvido na Faculdade de Letras da Universidade de Coimbra, ao longo de vários anos, pelo Doutor Joaquim Ramos de Carvalho.

Esta ferramenta assenta no uso de bases de dados estruturadas de forma a que a relação entre factos/entidades históricas seja analisada de forma eficiente e replicável. Para esse efeito é usada uma inserção de dados orientada às fontes, usando a notação *Kleio* desenvolvida por Manfred Thaller como ponte entre documentos antigos e a inserção de dados no sistema.

O contexto deste estágio baseia-se no desenvolvimento de *software* para esta aplicação, melhorando a sua usabilidade e adicionando novas funcionalidades, pois estes são uns dos pontos críticos que separam o *Time Link* entre projeto de investigação académica e produto. Mais especificamente a inserção de dados na aplicação e a sua instalação em máquina não são triviais para os utilizadores primários da ferramenta e requerem alguns conhecimentos na área de informática, o que se traduz numa barreira à sua utilização.

Hoje em dia as aplicações *Web* são utilizadas a nível mundial e por qualquer tipo de utilizador, disponibilizando informação e serviços através de uma simples ligação à internet. Apesar do *Time Link* ser implementado dentro deste paradigma, é ainda instalado localmente nas máquinas dos utilizadores, estando o *software* a ser implementado dentro deste contexto. Ainda assim o desenvolvimento destas aplicações engloba várias fases nomeadamente, planeamento, levantamento de requisitos, *design*, implementação e testes, tornando o trabalho a ser desenvolvido pelo estagiário relevante para a disciplina de estágio em engenharia informática.

Palavras-chave

"*Time Link*" , "*Package*" , "Requisito" , "*Kleio*" , "Projeto" , "Objetivo" , "Aplicação"

Índice

Índice de figuras

Índice de tabelas

1. Introdução

O Instituto Pedro Nunes (IPN) em Coimbra, é incubadora de várias empresas focadas em variadas áreas tecnológicas, estando entre elas a iClio, que procura a partir da inovação a introdução de novos produtos no mercado nacional e internacional. Mais especificamente esta empresa foca-se em conteúdos nas áreas da História, Património e Cultura tentando estabelecer uma ponte entre estes temas e as novas tecnologias como por exemplo dispositivos móveis e aplicações Web.

A iClio, fundada em 2010 por investigadores da Faculdade de Letras da Universidade de Coimbra, participa também em concursos tecnológicos, como por exemplo o congresso mundial de aplicações móveis onde em Fevereiro de 2011 viu o seu primeiro produto, o guia turístico "Just in Time Tourist" (JiTT), concebido para dispositivos móveis, ser considerado uma das três melhores aplicações para visitar a cidade de Barcelona. [1]

Dentro desta área temos também a aplicação Web MHK/Time Link. Este sistema é particularmente eficaz na recolha e tratamento de informação biográfica dispersa por várias fontes e suporta um processo reversível de reconstrução desta informação. Hoje em dia esta ferramenta é utilizada por vários Historiadores em investigação de micro-história, onde se delimita espacial e temporalmente a temática em questão e onde as fontes de informação são analisadas exaustivamente. O Time Link é um projeto open source, para onde a iClio contribui ativamente, sendo também o tema deste estágio. [2]

Este estágio assenta no desenvolvimento de software para a aplicação Time Link, propondo desafios ao nível de novos serviços e de agilização do processo de instalação e usabilidade. Sendo esta aplicação implementada dentro do paradigma Web, mas requerendo ainda instalação local por parte dos utilizadores e implicando a instalação de componentes como base de dados e servidor para o seu correto funcionamento, a agilização do processo de instalação será um dos pontos críticos a melhorar (packages de instalação), acompanhada pela implementação de software adicional (novas funções da aplicação).

Com estes objetivos bem definidos, espera-se deste estágio uma melhoria significativa do Time Link, contribuindo assim para um projeto há muito desenvolvido e apresentando aos seus utilizadores mais vantagens em utilizar esta ferramenta.

1.1. Estrutura do documento

Definido o âmbito do estágio e sua relevância, passamos à descrição da estrutura deste documento e dos principais tópicos a serem abordados.

- Antes de se iniciar o desenvolvimento de qualquer tipo de projeto de *software*, é determinante que a metodologia de trabalho a ser seguida e o planeamento sejam bem definidos, o que evita decisões tardias no que toca a pontos a desenvolver e retificações a efetuar. Assim sendo, no capítulo 2, vamos abordar estes dois tópicos, identificando também os riscos derivados desta metodologia, descrevendo assim o trabalho desenvolvido ao longo deste ano letivo e as estratégias de mitigação destes riscos.
- Numa fase posterior, definida no capítulo 3, vamos entrar num contexto mais específico de aplicações *Web* na área da genealogia, comparando a ferramenta para a qual foi efetuado o desenvolvimento de *software* com outras de sucesso. Além desta comparação vamos especificar a forma como funciona o *Time Link* internamente e o porquê desta aplicação ser diferente de outas ferramentas existentes nesta área.
- Após esta fase estamos preparados para apresentar os desenvolvimentos de *software* que permitem a agilização do processo de instalação e usabilidade, e novas funcionalidades como por exemplo importação de novos tipos de dados no *Time Link*, criação de famílias, entre outras. Como é normal num projeto de *software* de qualidade, estes desenvolvimentos são divididos nas fases de requisitos, arquitetura, implementação e testes (capítulos 4, 5, 6 e 7);
- Numa fase final vamos analisar o trabalho desenvolvido ao longo do ano letivo e refletir sobre a experiência de estágio em engenharia informática (capítulo 8).

2. Metodologia de trabalho e planeamento

Neste capítulo vamos descrever a metodologia de trabalho seguida durante o estágio e o planeamento efetuado no âmbito do desenvolvimento de *software*.

2.1. Metodologia de trabalho

A metodologia de trabalho deste estágio está dividida em cinco fases: análise do problema e soluções existentes, identificação e validação de requisitos, *design* da solução e seu desenvolvimento, fase de testes e entrega final. Vamos então passar a descrever estas fases, tendo em conta os processos utilizados pela empresa e o contexto da ferramenta para a qual foi desenvolvido o *software*.

2.1.1. Análise do problema e soluções existentes

Inicialmente analisámos o trabalho desenvolvido em termos de *software* para aplicações *Web* nos últimos anos na área da genealogia, comparando com o projeto *Time Link* e as ideias da empresa. Com esta análise identificámos os serviços disponíveis nesta área e propostas de solução que permitiram oferecer aos utilizadores um serviço melhor ou diferente dos que utilizam neste momento.

2.1.2. Levantamento e validação de requisitos

Nesta fase foi efetuado o levantamento de requisitos (definição do modelo do processo de requisitos, identificação de fontes etc.) sobre as funcionalidades implementadas tentando planear da melhor forma o desenvolvimento de *software* detetado como prioritário. Este levantamento de requisitos e sua validação foi sempre acompanhado pelo ponto de vista do Historiador e do utilizador comum, pois estes são os utilizadores primários da aplicação. Este é um processo iterativo até os requisitos serem validados.

2.1.3. *Design* da solução e implementação

Nesta fase foi elaborado o *design* da solução do problema em questão, em termos de engenharia de *software* e interface gráfica (se necessária) e iniciada à sua implementação dentro de um processo desenvolvimento ágil. É importante referir que durante esta fase foi utilizado um sistema de controlo de versões SVN (*Subversion*) através do *googlecode*, já referenciado.

2.1.4. Testes da solução implementada

Nesta fase foram ser efetuados testes à solução desenvolvida, tentando abranger os utilizadores primários da aplicação, Historiador e utilizador comum. Apontando a sua opinião sobre a solução apresentada, esta é melhorada de forma ir de encontro ás suas expetativas.

2.1.5. Entrega final e reflexão

Nesta fase foi efetuada a entrega final da solução implementada, a reflexão sobre as opções tomadas, problemas identificados e trabalhos possíveis para o futuro, visto que normalmente existem várias formas de resolver os problemas propostos e novos serviços e melhoramentos a desenvolver.

2.1.6. Acompanhamento

Ao longo destas fases, foram agendadas reuniões presenciais semanais ou quinzenais (numa metodologia ágil) onde foi revisto o trabalho desenvolvido na fase anterior e estabelecido o da fase seguinte.

O acompanhamento não presencial foi efetuado através da plataforma de gestão de projetos *Basecamp*. Esta permite a calendarização de objetivos, troca de ficheiros e informação, agendamento de reuniões, entre outros, promovendo assim a comunicação constante entre colaboradores. [3]

O CEO da iClio, Alexandre Pinto, acompanhou a execução das tarefas, assegurando o cumprimento dos objetivos. Perguntas de carácter mais técnico foram dirigidas ao Gestor de *Software* da empresa, João Carvalho.

2.2. Planeamento

A iClio estrutura os seus projetos focada em objetivos, através de uma relação constante com os seus colaboradores, sendo estes objetivos bem definidos a curto prazo e com datas flexíveis de entrega e revisão, caso seja necessário. A empresa estabelece assim uma relação de confiança com os orientandos na gestão do tempo e de processos, sendo estes acompanhados de forma a que os objetivos finais e o planeamento sejam cumpridos.

O planeamento deste estágio foi dividido em duas fases distintas, o primeiro semestre, onde foram elaborados *packages* de instalação para o *Time Link,* e o segundo semestre, onde foram implementadas novas funcionalidades no *core* aplicação. O diagrama que traduz este planeamento segue no Anexo I, que após a revisão de prioridades por parte da empresa, alterado em Janeiro de 2012, deu origem a um novo planeamento que segue também em anexo (Anexo II).

É importante referir que esta alteração originou a introdução de uma nova componente (*package* de instalação *MAC*) e a troca de uma das funcionalidades a desenvolver (documentação do sistema de *logs* substituída pela importação GEDCOM).

Para melhor entendimento das fases descritas no diagrama de planeamento deste estágio, passamos agora a descrever o trabalho elaborado nas mesmas.

- Após a primeira reunião com o orientador da empresa, e numa fase inicial do estágio, foi planeada uma semana de *brainstorming* e análise das ideias da empresa, com o objetivo de tentar trazer para esta ferramenta novos serviços que se possam tornar mais-valias para o *Time Link*.

- Após essa fase inicial foi elaborado o um estudo sobre o *Time Link,* onde foram analisadas aplicações de sucesso nesta área, e as suas funcionalidades. Esta fase permitiu entrar no contexto genealógico pois não possuía *background* em aplicações deste tipo. Ainda dentro desta fase foram estudados alguns projetos de investigação que utilizaram o *Time Link* como ferramenta principal e artigos sobre recriação de comunidades históricas.

- Numa fase posterior passamos à análise das funcionalidades desenvolvidas no primeiro e segundo semestre. Como já foi referido, o *Time Link* requer a instalação de várias componentes de instalação complexa, sendo necessário conhecimentos de informática para que o utilizador comum consiga ter a aplicação funcional no seu computador. Obviamente que qualquer pessoa que use esta ferramenta não deveria necessitar destes conhecimentos, pois este requisito traduz-se numa barreira à utilização da ferramenta. Por esse motivo foi proposto pelo orientador da empresa a elaboração de um *package* de instalação da aplicação, ou seja, algo que tornasse trivial a instalação do *Time Link* por qualquer utilizador. Assim sendo ficou decidido que para o primeiro semestre seria elaborado este *package,* contendo todas as ferramentas necessárias (base de dados, servidor, tradutor, editor de texto e *Java Runtime Environment*) para o bom funcionamento da aplicação. Após este desenvolvimento decidiu-se implementar também um *package* do mesmo tipo, mas que fosse suportado pelo sistema operativo *MAC*.

- Numa fase de planeamento do *software* a ser desenvolvido no segundo semestre, foram analisados vários pontos já avaliados como prioritários para o futuro do *Time Link*. Na página *mhk* do *googlecode*, podemos encontrar uma lista de funcionalidades que podem vir a ser desenvolvidas bem como defeitos da aplicação. Após uma análise destas *issues* foram escolhidas três funcionalidades a implementar (uma delas substituída pela importação GEDCOM), mais uma que teve origem na semana de *brainstorming*. Passamos então a enunciar o desenvolvimento de *software* planeado para o presente ano letivo.[4]

Primeiro semestre:

- *Package* de instalação *Windows* do *Time Link*;
- *Package* de instalação *MAC* do *Time Link* .

Segundo semestre:

- Importação de ficheiros GEDCOM (formato eletrónico de armazenamento de dados genealógicos);
- Algoritmo de cálculo de famílias e desenho da árvore genealógica na página *Web*;
- Edição *inline* (incorporada no *Web Browser*) de ficheiros *Kleio*;
- Introdução de dados no *Web Browser*.

Durante a estes desenvolvimentos foi também elaborado o relatório de estágio, intermédio e final (primeiro e segundo semestre, respetivamente), como descrito no planeamento que segue em anexo.

2.3. Riscos e mitigação

Estando o tema deste estágio enquadrado e o seu funcionamento definido, é importante analisar os riscos presentes na metodologia de trabalho e estratégias de mitigação dos mesmos. Devido ao facto do local de trabalho deste estágio ser remoto, podemos dizer que este é o risco mais evidente, pois o processo de produção de *software* pode ser afetado por falta de acompanhamento e horário de trabalho. Este é apenas um exemplo dos riscos presentes no desenvolvimento, pelo que passamos a enumerar todos os identificados.

- Local de trabalho – Como referido anteriormente, poderia afetar o processo de produção de *software* pois poderia ser necessário acompanhamento e horário rígido de trabalho.
- Comunicação com os principais autores de aplicação – O *Time Link* é uma aplicação já existente, e por esse motivo leva ao estudo do seu estado atual em termos de implementação, permitindo-nos adicionar novas funcionalidades seguindo a mesma lógica do que tem sido implementado até aqui. Caso esta comunicação não fosse favorável poderia levar a tomadas de decisão sobre as soluções a desenvolver que não fossem as melhores, dentro do contexto do *Time Link*.

Sendo estes os principais riscos identificados, vamos avaliar os mesmos de forma perceber o impacto que poderiam ter nas fases de desenvolvimento e nos objetivos finais do estágio. Na tabela

seguinte podemos observar a classificação dos riscos, avaliados na escala de 0 a 5, em impacto e probabilidade, e a estratégia de mitigação aplicada.

Riscos	Impacto	Probabilidade	Estratégia
Local de trabalho	5	2	Planeamento bem efetuado, revisto e seguido rigorosamente.
Comunicação com autores	5	4	Agendamento de reuniões avaliação do desenvolvimento.

Tabela 1 – Identificação dos riscos do estágio.

Sendo o local de trabalho remoto, foram estabelecidas estratégias para mitigar riscos derivados deste facto seguindo o planeamento rigorosamente, tentando assim combater a ausência de acompanhamento e horário de trabalho. Com a ajuda dos orientadores foram efetuadas revisões ao planeamento, de modo a que este risco não se verificasse.

Derivado também da falta de acompanhamento estava a hipótese das decisões ao nível do desenvolvimento não serem tomadas corretamente por parte do estagiário. Estes erros verificados tardiamente poderiam atrasar o planeamento, pelo que o agendamento de reuniões e avaliação periódica do desenvolvimento mitigaram este risco.

3. Estudo - *Time Link*

Uma das primeiras fases deste estágio incidiu sobre a análise da ferramenta para a qual foi efetuado o desenvolvimento de software e sobre o estudo de aplicações de sucesso na área genealogia. Neste capitulo vamos abordar estes tópicos e concluir sobre as soluções já existentes em aplicações deste género, começando pela arquitetura do Time Link.

3.1 Arquitetura do *Time Link*

Nesta fase vamos especificar a arquitetura do *Time Link* em geral e do seu sistema, bem como o modelo em que se enquadra no âmbito de aplicações *Web* e algumas das funcionalidades já presentes.

3.1.1. O modelo MVC

O *Time Link* é uma das muitas aplicações *Web* que segue o modelo MVC (Model-View-Controller), modelo este que separa a aplicação em três camadas:

- Visualização – Páginas *Web* acessíveis pelo utilizador. Efetuam pedidos à camada de controlo e recebem as respostas do servidor.
- Controladores – Implementam toda a lógica da aplicação. Recebem pedidos da camada de apresentação/visualização e enviam as respostas respetivas, gerindo os acessos à base dados (camada de modelos).
- Modelos – Estrutura da base de dados da aplicação. Contém toda a informação armazenada e comunica com a camada de controlo devolvendo informação.

Para melhor compreensão podemos visualizar um exemplo da comunicação efetuada neste tipo de aplicações na figura seguinte. [5]

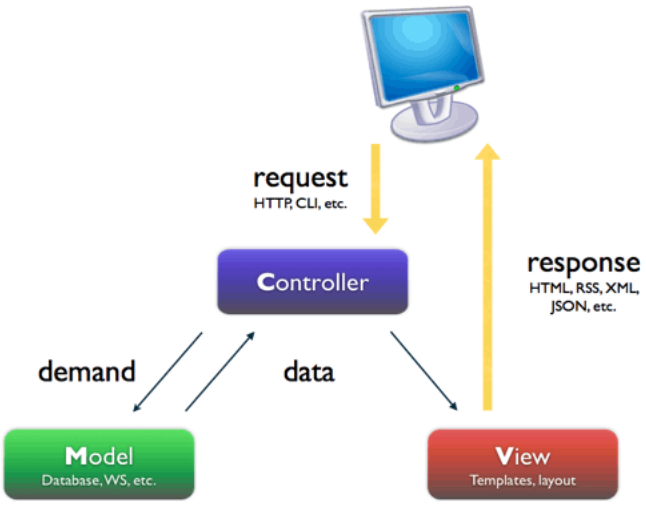

Figura 1 – Exemplo modelo MVC.

Após uma visão geral sobre a arquitetura do *Time Link*, vamos agora especificar melhor a camada de modelos (segundo o modelo MVC) que permite o armazenamento e gestão de dados da aplicação.

3.1.2. Camada de Modelos

O *Time Link* trabalha sobre informação relativamente complexa em termos de relacionamento. As transcrições de documentos inseridas na base de dados geram informação sobre pessoas, eventos históricos, suas relações e atributos, que necessitam de ser reconstituídos com precisão. Como é normalmente utilizado neste tipo de situações, a estrutura de base de dados do *Time Link* assenta num modelo relacional que visa o armazenamento de dados e de relações entre eles, permitindo guardar e obter informação variada.

No caso da ferramenta em análise, a estrutura de base de dados está dividida essencialmente em três núcleos: tratamento de entidades, relações e atributos.

- O núcleo de entidades engloba todo o tipo de eventos históricos e participantes (atos e pessoas, respetivamente), guardando alguns atributos fixos respetivos ao tipo de entidade em questão.
- O núcleo de relações permite relacionamentos entre as entidades (eventos e pessoas), por exemplo presenças em eventos, papel representado, relações de parentesco, etc.
- O núcleo de atributos armazena informação sobre as entidades, permitindo que qualquer tipo de atributo possa ser armazenado sem ser necessária a restruturação da base de dados.

Para representar melhor esta estrutura foi elaborado um diagrama de classes do *Time Link* contendo apenas as entidades base (pois o *Time Link* neste momento contém uma estrutura muito mais complexa que dá suporte a funcionalidades de georreferenciação, *record linking*, entre outras) na linguagem UML (*Unified Modeling Language*), muito utilizada neste tipo de representações, sendo que cada classe corresponde a uma tabela na base de dados.

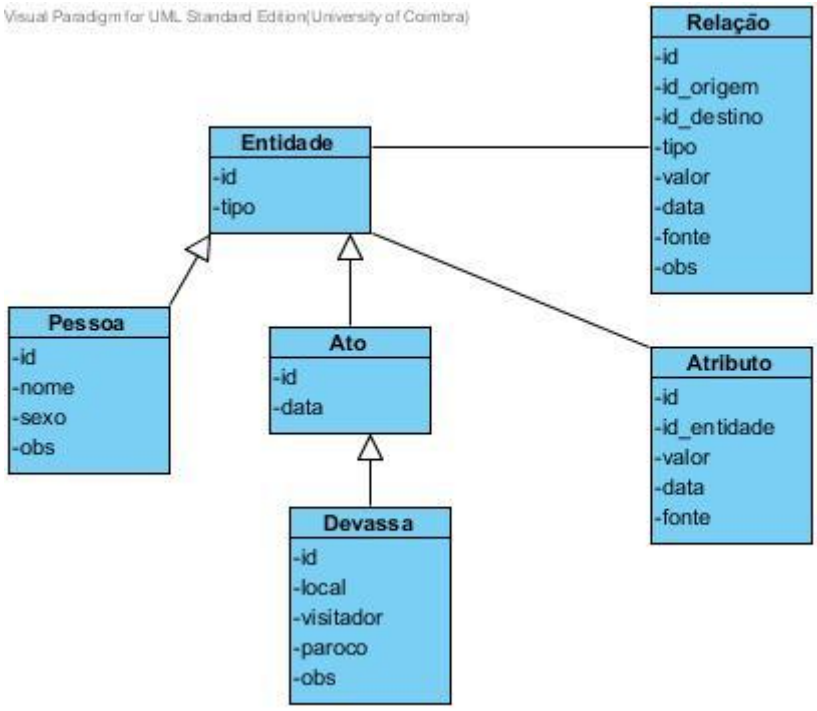

Figura 2 – Diagrama de classes do *Time Link*.

Como podemos observar, as classes Pessoa e Ato descendem da classe Entidade, ou seja, são casos específicos da mesma, o que permite distinguir facilmente entre estes dois tipos. A classe Ato pode ainda dar origem a novas entidades, mais especificamente, a classe representada como Devassa é uma classe genérica, que mediante a introdução de informação de eventos históricos pode ser criada. Este tipo de abordagem permite armazenar qualquer tipo de evento na base de dados, pois esta é adaptável.

É importante referir que a classe Pessoa armazena "ocorrências" de pessoas, e não pessoas "reais", o que leva a várias entradas na tabela sobre a mesma pessoa. Este tipo de funcionamento é essencial, pois o *Time Link* agrupa estas entradas nas tabelas em função das pessoas "reais". A este processo dá-se o nome de *record linking*.[6]

A classe Relação permite definir relações entre entidades, por exemplo relações de parentesco (pois esta classe contém o id da entidade origem, destino, e o valor da relação). A classe atributos permite adicionar qualquer tipo de atributo a uma entidade, por exemplo data de nascimento, altura, peso etc. (pois esta classe contém o id da entidade à qual pertence o atributo e o valor do mesmo).

Apesar desta ser apenas a estrutura base do *Time Link*, é sobre estes dados que trabalham as funcionalidades implementadas e descritas nos capítulos de desenvolvimento de *software*.

Uma vez vista a camada que armazena a informação do *Time Link*, vamos fazer uma breve descrição sobre as duas outras camadas do modelo MVC.

3.1.3. Camada de Controlo

Nesta camada está presente toda a lógica da aplicação que opera sobre os dados armazenados e responde aos pedidos do utilizador.

Mais especificamente o *Time Link* utiliza tipos de classes *Java* denominadas *Servlets* para receber os pedidos no protocolo *HTTP* vindos do utilizador. Essas classes dão início a toda a lógica de *back office* da aplicação, por exemplo escritas e leituras na base de dados e envio de respostas para o cliente.

Específico da aplicação *Time Link,* é a utilização de um tradutor de ficheiros *Kleio,* implementado na linguagem de programação Prolog, que traduz estes ficheiros para informação XML (*eXtensible Markup Language*) a ser inserida na base de dados. Este tradutor é inicializado com a aplicação através do código *Java* e de ficheiros de configuração (extensão .conf).

3.1.4. Camada de Apresentação

Nesta camada encontra-se toda a apresentação ao utilizador. É implementada na linguagem de *templates Apache Velocity* e na linguagem de *markup* HTML *(HyperText Markup Language*), utiliza também a linguagem de estilo CSS (*Cascading Style Sheets*) para formatação da apresentação da página, e por fim, a linguagem de *scripting* JS (*JavaScript*) que ajuda também na apresentação/interação e executa pedidos assíncronos ao servidor.

3.1.5. Estado inicial do *Time Link*

Visto que o *Time Link* é uma aplicação já existente, seria interessante apresentar algumas das funcionalidades já presentes, tentando perceber também a relação entre o trabalho desenvolvido e o estado inicial da aplicação.

O *Time Link* permite a tradução de ficheiros *Kleio* (transcrições de documentos antigos) para XML, podendo este *output* da tradução ser importado na base de dados. Esta funcionalidade já presente assemelha-se ao desenvolvimento de importação GEDCOM na medida em que foi também produzido um ficheiro XML a partir deste tipo de ficheiros, e posterior importação. Na figura seguinte podemos observar o estado anterior do *Time Link*, quando apenas permitia tradução e importação *Kleio*.

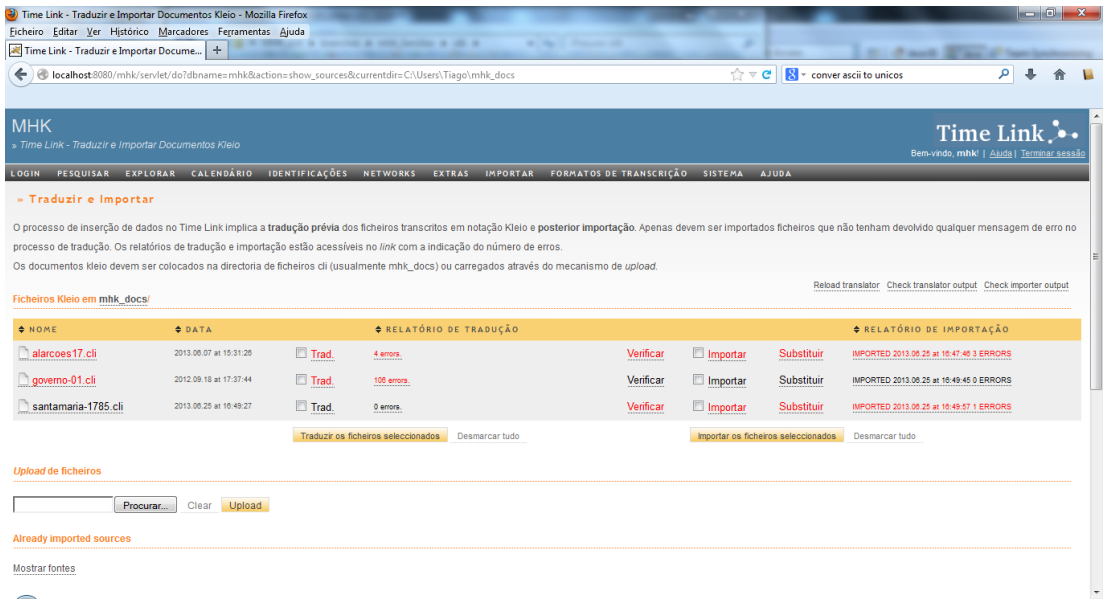

Figura 3 – Estado inicial da importação e tradução

Podemos observar que apenas existe a tabela de importação de ficheiros *Kleio* e que ainda não temos a possibilidade de criar fontes de informação no através do *Web Browser* (esta opção estaria presente no menu contido no *header* da página onde se encontra por exemplo a importação).

Outra das características do *Time Link*, é apresentação de relações de qualquer tipo entre entidades, sendo estas apresentadas na forma de texto e na página *Web* respetiva à mesma. Na figura seguinte podemos observar a apresentação desta informação (apenas de fontes *Kleio*) no estado anterior da aplicação, pois o desenvolvimento da funcionalidade de famílias permite agora a construção das mesmas (de fontes *Kleio* e GEDCOM) na base de dados e graficamente na página *Web*.

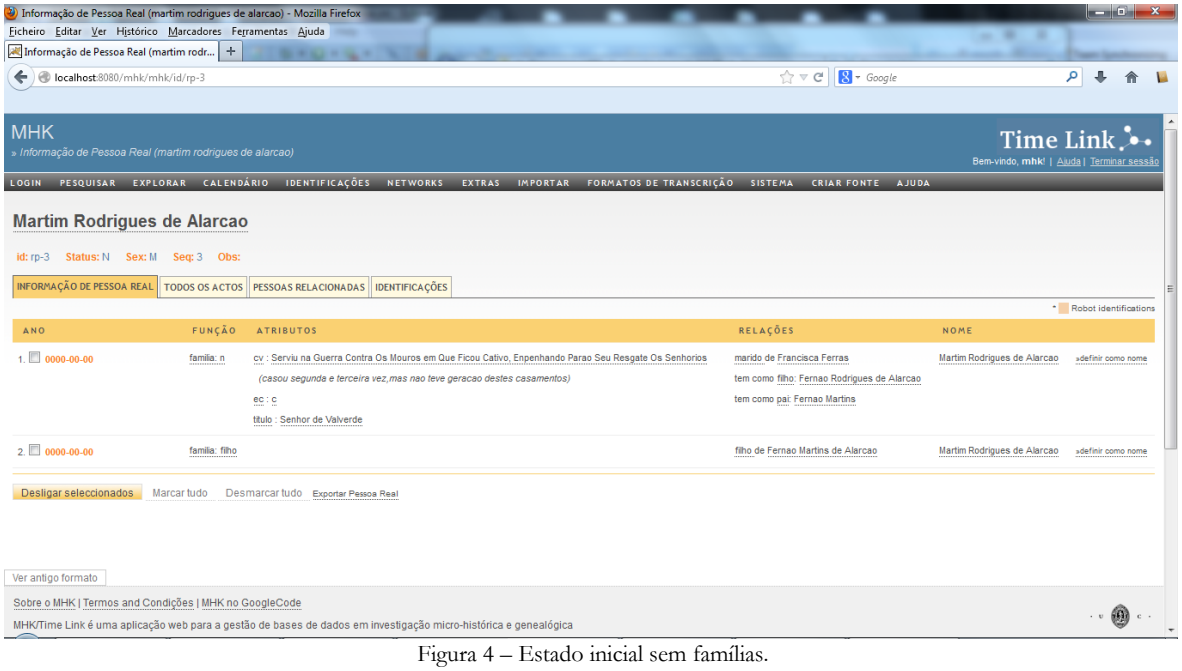

Figura 4 – Estado inicial sem famílias.

3.1.6. Arquitetura de sistema

Após uma descrição mais geral, vamos então definir como funciona o processo na camada de controlo. Como vimos anteriormente, o cliente efetua pedidos através do protocolo HTTP ao servidor, sendo estes tratados por uma classe *Java* do tipo *Servlet* que identifica o tipo de pedido e dá continuação à sua execução.

Como é normal em qualquer tipo de aplicações, alguns destes pedidos dão origem a tarefas simples como *login* ou *logout* do *Website*, mas existem também tarefas complexas e custosas em termos temporais (tradução *Kleio*, importação, etc.). Por este motivo, o *Time Link* mantém uma *thread* a executar que permite o registo deste tipo de pedidos, gerindo-os através de uma fila de tarefas FIFO (*first in*, *first out*). Tendo em conta que no caso de tarefas complexas estas executam através de classes *Java* que implementam a interface *Worker* (tarefa genérica) executada a partir da *thread*, esta abordagem permite que caso se adicionem novas tarefas a serem executadas desta forma, seja apenas necessário criar um novo *Worker* para cada tarefa especifica.

Devido a esta complexidade, na fase desenvolvimento de *software* vamos explicar como se integram as novas funcionalidades neste sistema. Por agora podemos observar na figura seguinte o diagrama da arquitetura de sistema do *Time Link*.

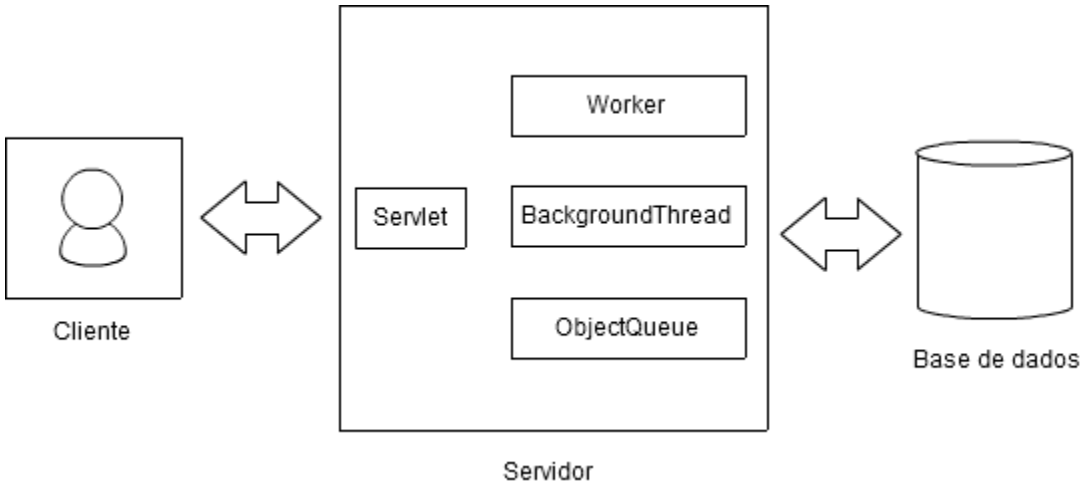

Figura 5 – Arquitetura de sistema do *Time Link*.

3.2. Utilização do *Time Link*

Após a apresentação da arquitetura do *Time Link*, e especialmente no caso desta aplicação, é de extrema importância perceber como é efetuada a sua utilização hoje em dia, pois apesar de estar implementado segundo o paradigma *Web*, o *Time Link* é instalado localmente nas máquinas dos utilizadores, sendo este o motivo principal para a implementação de *packages* de instalação. Esta execução local deve-se ao facto do *Time Link* não estar disponibilizado/*deployed* para livre utilização na

internet, pois este não é ainda um produto, sendo apenas utilizado em investigação académica e por utilizadores interessados. Apesar de existirem bases de dados que permitem testar algumas das suas funcionalidades, não permitem ao utilizador trabalhar sobre a sua própria informação.

Definida a forma de utilização do *Time Link*, vamos efetuar uma breve análise de aplicações semelhantes (dentro do paradigma *Web*) em termos genealógicos, pois os serviços disponibilizados por estas permitem perceber as necessidades dos utilizadores deste tipo de ferramentas.

3.3. Análise de aplicações semelhantes

Muito antes da globalização da internet já se desenvolvia *software* na área da genealogia, seja este com objetivo de utilização profissional ou amadora. Após esta globalização as aplicações *stand alone* perdem ênfase e dão lugar a aplicações *Web,* pois estas promovem a partilha de informação e permitem acesso a nível mundial.

Atualmente existem várias aplicações e projetos na área da genealogia e neste ponto vamos analisar algumas das mais inovadoras tentado separa-las pela sua abordagem ao negócio e ideia, baseando-nos em estudos sobre as melhores aplicações *Web* de genealogia no ano de 2012.[7]

Esta análise vai incidir também sobre as funcionalidades atuais destas aplicações, trazendo uma visão geral sobre o que os utilizadores procuram hoje em dia em ferramentas nesta área, conseguindo assim trazer novas ideias para o *Time Link*. A análise foi efetuada pelo estagiário entre os meses de Setembro e Outubro do ano 2012 e foi útil na medida em que conseguimos perceber que as funcionalidades a desenvolver (importação GEDCOM e desenho de árvores genealógicas) estão presentes na grande maioria destas aplicações, confirmando assim a importância dos desenvolvimentos deste estágio.

3.3.1. Aplicações

Ancestry.com é um dos *WebSites* mais conceituados na área da genealogia. Requer assinatura mensal mas oferece funcionalidades como criação de árvores genealógicas, *upload* de ficheiros no formato GEDCOM (formato eletrónico de armazenamento de dados genealógicos), ligação com redes sociais, procura por registos de *Census*, listas de votações, nascimentos, óbitos, entre outros. Recentemente lançou o projeto *AncestryDNA* que permite, a partir de informação ADN, voltar muitas gerações atrás e descobrir quais as suas verdadeiras origens. [8]

Archives.com é uma aplicação que segue a mesma ideia do *WebSite* anterior, com assinatura mensal, procura por registos militares, nascimentos, óbitos, casamentos, entre outros. Permite também a importação de informação em ficheiros no formato GEDCOM e está a ser pensado nesta altura que pode vir a fazer parte de ancestry.com devido ao aumento de subscrições.[9][10]

GenealogyBank.com assenta no mesmo modelo de negócio que os anteriores. Introduz a possibilidade de consultar jornais de pequenas e grandes cidades dos Estados Unidos da América. Contém também registos militares, obituários, documentos históricos, entre outros.[11]

FamilySearch.org segue um modelo de negócio diferente pois não requer qualquer encargo para o uso das funcionalidades da aplicação. Permite também a procura de informação a partir de registos de nascimento, casamento, residência, entre outros, possuindo também livros com história de famílias para livre consulta. [12]

Dyncoopnet-pt.org é um projeto que surge num contexto um pouco diferente, pois ainda não sendo um produto final, é um projeto de investigação que procura tirar vantagem da grande quantidade de documentos de duas naturezas: letras de câmbio e correspondência de mercadores. Usa como ferramenta principal de recolha de dados o *Time Link*.[13]

3.3.2. Comparação entre aplicações analisadas

Segundo comparações entre os melhores serviços de procura genealógica do ano de 2012 podemos confirmar que analisámos as melhores aplicações desta área nos últimos anos, observando a figura em anexo (Anexo III). [14]

Podemos também confirmar que a informação retirada previamente pelo estagiário vai de encontro à apresentada na medida em que são utilizados vários tipos de fontes de informação nestas aplicações, assim como vários serviços disponibilizados, no entanto, a sua grande maioria não dispensa mensalidade.

Alguns destes *WebSites* apostam em serviços inovadores, nomeadamente testes de ADN (introduzidos por ancestry.com e archives.com). É importante também referir que o formato GEDCOM é muito utilizado por este tipo de aplicações, pois promove a partilha de dados entre as mesmas.

Visto que objetivo deste estágio é fornecer ao utilizador novos serviços e melhorar a usabilidade do *Time Link*, a análise de todas estas ferramentas e suas funcionalidades é fundamental, conseguindo perceber melhor o que o utilizador procura neste tipo de aplicações.

3.3.3. Desenvolvimentos possíveis para o *Time Link*

Depois de efetuada uma primeira análise de algumas aplicações *Web* na área da genealogia pelo estagiário, consultando estes *WebSites* e testando as suas funcionalidades, podemos observar vários pontos que tivemos em conta para melhorar o *Time Link*, tendo como critério de comparação os serviços disponibilizados, usabilidade destas aplicações, o estado atual do *Time Link* e as ideias da empresa.

Inicialmente apenas existia a possibilidade de importar dados no formato *Kleio*, e, pelo que se observou o formato GEDCOM é bastante utilizado neste tipo de aplicações, tanto para importação como para exportação de dados, podendo facilitar a integração com outros tipos de sistemas.

No processo de tradução de documentos antigos, o utilizador tem de produzir um ficheiro na notação *Kleio* para este ser depois traduzido em dados a serem inseridos no sistema. Esta criação do ficheiro é manual e local, pelo que poderia ser efetuada a inserção de dados diretamente na aplicação, por exemplo num formulário ou ambiente gráfico, gerando o ficheiro *Kleio*, se necessário, e posterior tradução.

Para além destes aspetos, o utilizador necessitava de conhecimentos de informática para a instalação e utilização do *Time Link*, pois era obrigado a instalar todas as ferramentas necessárias para executar a aplicação (servidor, base de dados e tradutores) e seguir com rigor um certo número de regras para que conseguisse executar a mesma com sucesso. Como seria de esperar, um utilizador comum não deveria necessitar de ter estes conhecimentos para contribuir para o projeto, logo, agilizando esta fase do processo, (por exemplo inserção e tradução *online* através do *Web browser*, e *packages* de instalação da aplicação) seja uma grande mais-valia para o *Time Link*, permitindo o contributo de informação a nível mundial.

3.3.4. Contribuições deste estudo para o estágio

Este estudo contribuiu para o estágio na medida em que se observou que a maioria das aplicações têm processos agilizados para execução das funcionalidades presentes, o que veio confirmar a ideia de que existe ainda uma grande barreira na utilização do *Time Link* por utilizadores comuns e Historiadores, tornando prioritária a elaboração dos *packages* de instalação da aplicação.

Outro aspeto relevante para este estágio foi o facto de estas aplicações suportarem a edição e geração de ficheiros *online* em vários formatos, o que trouxe uma nova ideia o *Time Link*. Como foi dito anteriormente a criação de ficheiros *Kleio* é local e tem que seguir várias regras que por vezes podem não ser triviais para um utilizador não familiarizado com as convenções desta linguagem. Se este ficheiros forem ser criados na aplicação exibindo regras claras aquando a inserção de dados, por exemplo num formulário, criamos também uma mais-valia para esta ferramenta, eliminando mais uma barreira à utilização completa do *Time Link*.

Como referido anteriormente, a maioria das aplicações analisadas tem suporte à exportação e importação de ficheiros no formato GEDCOM. O *Time Link*, neste momento apenas fornece a opção de exportar dados neste formato, o que se traduz também numa barreira à sua utilização. Podemos tomar por exemplo um utilizador que tem os seus dados genealógicos inseridos noutra ferramenta e pretende migrar os mesmos para o *Time Link*. Esta migração tem de ser efetuada manualmente, com o

utilizador a inserir de novo os seus dados na ferramenta, o que pode ser otimizado permitindo a importação de ficheiros no formato GEDCOM.

3.4. Projetos de investigação e teses

Como referido anteriormente o *Time Link* tem sido usado como ferramenta principal em vários projetos de investigação e teses. Assim sendo achamos importante o estudo de alguns destes trabalhos para ajudar a perceber o que realmente faz desta aplicação um projeto com futuro promissor.

Entre os vários projetos e teses analisados estão a tese de Mestrado de João Carvalho (João Carvalho, 2009) [15], a tese de Doutoramento de Joaquim Ramos de Carvalho (Ramos de Carvalho, 2007) [16], e alguns artigos sobre investigação nesta área. [17][18]

Esta fase contribuiu para uma melhor perceção do que se pretende em aplicações deste género, no contexto da genealogia, podendo até trazer novas ideias de desenvolvimento. Os resultados dos projetos de investigação elaborados com o *Time Link* como ferramenta principal mostram o seu verdadeiro potencial permitindo várias conclusões interessantes sobre hábitos de comunidades históricas e recriação das mesmas.

Um exemplo destes projetos é o artigo do Doutor Joaquim de Carvalho, referenciado anteriormente, onde foi efetuado um estudo sobre as visitas episcopais em Soure, no séc. XVI, que permitiu identificar padrões recorrentes em relações ilícitas de casais. Mais especificamente concluiu-se, utilizando o *Time Link* para construção da "rede" de informação, que algum parente do homem interveniente na relação ilícita apadrinhava alguém na família da mulher também envolvida na relação.

3.5. Notação *Kleio overview*

Tem sido referido até aqui que o *Time Link* permite importação de ficheiros na notação *Kleio*. Esta notação é utilizada em transcrições de documentos antigos e a sua linguagem tem origem no *Max-Plank Institute für Geschichte*.

Os ficheiros produzidos (extensão .cli) são também denominados por ficheiro de transcrição de uma fonte e utilizam conceitos de grupo e elemento, sendo que estes podem ser correspondidos a uma entidade e seus atributos (segundo um modelo de bases de dados), respetivamente.

Um exemplo de uma transcrição de um documento é o registo de um batizado, onde este tem o grupo batizado e sendo os seus elementos o padrinho, o pai, a mãe etc. Todas as diferentes fontes (eventos históricos registados) estão documentadas num ficheiro onde as regras e convenções são bem explícitas.

Este tipo de ficheiros (convertidos para XML pelo *Time Link*) eram na verdade, a sua única fonte de informação. Tendo em conta que a importação de dados no formato GEDCOM foi incluída no planeamento, e introdução de dados no *Web Browser* já estava presente no mesmo, é de extrema importância que a forma como é efetuada a conversão e importação dos ficheiros *Kleio* assim como as suas normas sejam analisadas.

3.6. O *Time Link* como projeto *open source*

Como já foi referido anteriormente o *Time Link* é uma aplicação *open source* para onde a iClio contribui ativamente, sendo que o código da aplicação está disponível no *googlecode*, referenciado anteriormente, assim como instruções de instalação e utilização.

No âmbito deste tipo de projetos, várias entidades podem contribuir com código e ideias para a aplicação, sendo que o *software* desenvolvido neste estágio poderá ou não, dependendo das opções da empresa, ser disponibilizado para posterior edição. O estagiário irá desenvolver *software* para um dos "ramos" do *Time Link*, suportado pela iClio.

É importante também referir que no contexto de projetos deste tipo, todas as ferramentas e bibliotecas utilizadas serão de livre utilização e devidamente referenciadas.

4. Requisitos

Nesta fase vamos apresentar o levantamento de requisitos do *software* desenvolvido durante o estágio, tendo em conta todas as limitações do *Time Link* já apresentadas. Definindo o modelo do processo de requisitos e identificando casos de uso e outras possíveis fontes, foi elaborada a especificação de requisitos sobre as componentes de *software* desenvolvidas.

4.1. Modelo e atores do processo de requisitos

Como é normal num projeto de engenharia de *software*, inicialmente deve ser definido e contextualizado o modelo do processo de requisitos a ser seguido durante o desenvolvimento das componentes de *software*. No caso do *Time Link*, visto que é uma aplicação *open source*, permite-nos inserir os desenvolvimentos no contexto deste tipo de aplicações, e restringir o *software* de instalação e bibliotecas a utilizar a esta gama de livre utilização.

Definido este contexto, faz sentido tentar perceber quem vai ter um papel ativo no processo de requisitos (*stakeholders*), pois estes indivíduos ou atores participam ativamente nas fases de análise, especificação e validação de requisitos. Assim sendo estas fases vão ser acompanhadas pelo CEO da empresa, Alexandre Pinto, o gestor de *software* da mesma, João Carvalho e pelos Historiadores João Pinho e Sónia Nobre.

Ainda dentro deste modelo, e especifico do tipo da implementação dos instaladores, é a análise de componentes a incorporar nos mesmos, como servidor e base de dados necessárias na execução do *Time Link*.

4.2. Fontes e técnicas de identificação de requisitos

É comum também num projeto de desenvolvimento de *software*, identificar-se as potenciais fontes de requisitos e técnicas de identificação dos mesmos, por forma a perceber e definir com detalhe os objetivos e o comportamento do *software* a produzir. O facto do *Time Link* ser uma aplicação *open source*, e a implementação dos *packages* e funcionalidades ser efetuada neste contexto, é exemplo de uma destas fontes, pois implicitamente faz com que a utilização de um *software* de instalação livre de encargos ou até mesmo bibliotecas *open source* seja traduzida em requisitos. Entre as potenciais fontes identifica-mos:

- Contexto *open source* – Traduz-se na utilização de *software* de instalação e bibliotecas livres encargos;
- *Stakeholders* – Assumindo o seu papel e o de utilizadores vão ajudar a identificar a grande parte dos requisitos das componentes de *software*;

- Componentes a utilizar – Obriga à integração destas componentes de modo a permitir ao utilizador instalar e executar o *Time Link* de forma intuitiva.

Através de testes com vários *softwares* de instalação e aplicação de técnicas de identificação de requisitos, nomeadamente a identificação de casos uso com utilizadores e autores da aplicação e a integração das componentes a desenvolver no *Time Link*, resultam os requisitos do *software*.

4.3. Requisitos – *Packages* de instalação

Nesta fase, através de testes efetuados pelo estagiário, contato com Historiadores que utilizam a aplicação e ideias da empresa, foram identificados alguns casos em que se verificam dificuldades na utilização do *Time Link*. Estes casos são derivados também da necessidade de instalação e execução dos vários pré-requisitos da aplicação, que passamos a descrever.

4.3.1. Pré-requisitos do *Time Link*

Antes de definição dos casos de uso faz sentido evidenciar que o *Time Link* necessita de alguns pré-requisitos instalados para funcionar corretamente na máquina do utilizador. Neste ponto vamos enumerar as componentes necessárias para o seu bom funcionamento e explicar os seus propósitos. É importante também referir que as versões das componentes utilizadas nos instaladores diferem consoante o sistema operativo (*Windows* e *MAC*), sendo estas referidas posteriormente.

- *Java Runtime Environment* – Muitas máquinas já têm este ambiente instalado, mas devido ás diferentes arquiteturas de *software* (por exemplo sistema operativo) assumimos necessário ter a certeza de que este está presente na máquina (caso *Windows*) e da sua localização, pois, alguns dos pré-requisitos que se seguem necessitam de configuração e executam sobre o ambiente *Java*. O sistema operativo *MAC* já contém este ambiente instalado pelo que não foi introduzido no *package* de instalação.
- MySQL – É dos sistemas de base de dados relacional mais utilizados (vasta comunidade e de livre utilização) e é também utilizado no *Time Link*. A sua instalação é fundamental para o bom funcionamento da aplicação, pois é onde serão guardados os dados inseridos na mesma.
- *Apache Tomcat* – Servidor aplicacional do *Time Link*. É também dos servidores mais utilizados em aplicações *Web* (vasta comunidade e livre de encargos). A sua instalação é também fundamental para o funcionamento da aplicação pois é onde vai ser efetuado o *deployment* da mesma, tornando o *Time Link* acessível através do *Web browser*.
- SWI-Prolog – O *Time Link* permite o *upload* ficheiros na notação *Kleio* (extensão .cli), tradução dos mesmos *online* e posterior importação para a base de dados. Este tradutor está implementado na linguagem de programação Prolog, e a instalação deste ambiente é também

fundamental para o bom funcionamento da aplicação, pois de outra forma não será possível traduzir os ficheiros *Kleio* em informação a ser inserida na base de dados.

- jEdit – Tem sido o editor de texto utilizado na escrita dos ficheiros *Kleio*. A sua instalação não é necessária para o funcionamento da aplicação mas caso o utilizador (por exemplo Historiador) queira produzir os seus ficheiros será muito útil.
- Ficheiro mhk.war – Ficheiro do tipo *Web application resource* que contém a aplicação em si. Este ficheiro necessita de estar presente na pasta de *Web applications* do servidor e não difere entre sistemas operativos.

Para definir melhor as versões das componentes utilizadas (*Windows* e *MAC*), na tabela seguinte apresentamos esta distinção.

Componente	Versão *Windows*	Versão *MAC*
JRE	7	N/A
MySQL	5.5.31	5.5.29
Apache Tomcat	7.0.35	7.0.34
SWI-Prolog	5.8.3	5.8.3
jEdit	5.0.0	5.0.0
mhk.war	2.1	2.1

Tabela 2 – Versões das componentes por sistema operativo.

Com todos estes pré-requisitos instalados na máquina temos as condições necessárias para executar a aplicação em sistemas operativos *Windows* e *MAC*, passando agora à identificação dos casos de uso.

4.3.2. Casos de uso

Apesar dos *packages* de instalação serem desenvolvidos para sistemas operativos distintos, os casos de uso identificados coincidem, pois estes derivam da dificuldade de instalação e utilização das componentes identificadas anteriormente. Assim sendo, passamos a descrever as dificuldades identificadas pelos utilizadores.

- Início e paragem de serviços – Como foi referido anteriormente, o *Time Link* utiliza um servidor aplicacional e uma base de dados de informação. Estes serviços necessitam de ser iniciados para a aplicação funcionar corretamente, e parados para não consumirem recursos na máquina. O início e paragem destes serviços não são triviais para o utilizador comum e como tal necessitam de ser efetuados através de um processo agilizado.
- Consistência com aplicações por sistema operativo – Para melhorar a experiência do utilizador na utilização da aplicação e facilitar o processo de aprendizagem seria usual que o *Time Link*

seguisse o exemplo de outras aplicações usadas nos sistemas operativos *Windows* e *MAC*, como por exemplo atalhos para iniciar a aplicação, para desinstalação, entre outros.

- Mensagens de erro do servidor e base de dados – Como é habitual neste tipo de aplicações, por vezes são geradas mensagens de erro que podem derivar do início ou paragem der serviços, exceções, etc. Assim sendo seria interessante abstrair o utilizador destas mensagens, pois este não necessita deste tipo de informação.

Posteriormente estes casos identificados foram verificados pelo orientador da empresa e pelo Historiador, traduzindo-se em requisitos dos *packages* de instalação. Para resolver estes problemas decidiu-se utilizar as linguagens de *scripting* descritas nas fases de implementação respetivas.

4.3.3. Levantamento de requisitos do *package* de instalação *Windows*

Após identificados alguns problemas a ultrapassar na instalação e utilização do *Time Link*, devemos também ter em conta o *software* de instalação a utilizar, pois apesar da vasta gama disponível, o *Time Link* é um projeto *open source*, logo este *software* de instalação deve ser livre de encargos. Devido a estas restrições e aos problemas a resolver, foi efetuado o levantamento de requisitos do *package* de instalação *Windows* que passamos a descrever.

1. Implementação de um mecanismo que inicia os serviços de base de dados e servidor numa só ação;
2. Implementação de um mecanismo que termine a execução dos serviços de base de dados e servidor numa só ação;
3. Implementação de um mecanismo que configure a diretoria da variável de ambiente JAVA_HOME automaticamente;
4. Disponibilizar atalhos para a execução dos mecanismos de início e paragem de serviços, desinstalação e início do jEdit;
5. Não apresentar a janela de execução do servidor e da base de dados.
6. Utilização de um *software* de instalação livre de encargos;
7. Implementação de um mecanismo que elimine as componentes instaladas do disco numa só ação;

Com estes requisitos conseguimos obter os resultados pretendidos no que toca à agilização do processo de instalação e execução do *Time Link*. Estes requisitos são definidos como funcionais e não funcionais e também classificados quantitativamente com prioridades, como podemos observar na tabela seguinte. É importante referir que a escala atribuída à prioridade dos requisitos está definida entre 0 e 5.

Requisitos	Funcional	Prioridade
1. Início de serviços	✓	5
2. Paragem de serviços	✓	5
3. Configuração JAVA_HOME	✓	5
4. Atalhos	✓	4
5. Janela de execução	✓	2
6. Utilização de um *software* livre de encargos	×	5
7. Eliminação das componentes do disco	✓	5

Tabela 3 – Requisitos do *package* de instalação *Windows* do *Time Link*.

Os requisitos identificados anteriormente foram verificados e validados com o Historiador, o orientador da empresa e com gestor de *software* da mesma.

4.3.4. Levantamento de requisitos do *package* de instalação *MAC*

Dentro do mesmo contexto do *package* de instalação anterior, mas respetivo ao sistema operativo *MAC* e tendo em conta todos os problemas identificados, foi também efetuado o levantamento de requisitos do instalador que passamos a descrever.

1. Implementação de um mecanismo que inicia os serviços de base de dados e servidor numa só ação;
2. Implementação de um mecanismo que termine a execução dos serviços de base de dados e servidor numa só ação;
3. Implementação de um mecanismo que configure as permissões necessárias da base de dados automaticamente;
4. Disponibilizar atalhos para a execução dos mecanismos de início e paragem de serviços, desinstalação e início do jEdit;
5. Utilização de um *software* de instalação livre de encargos;
6. Implementação de um mecanismo que elimine as componentes instaladas do disco numa só ação;
7. Não apresentar a janela de execução do servidor e da base de dados.

Estes requisitos são novamente definidos como funcionais e não funcionais e classificados quantitativamente com prioridades, como podemos observar na tabela seguinte, com a escala atribuída à prioridade dos requisitos definida entre 0 e 5.

Requisitos	Funcional	Prioridade
1. Início de serviços	✓	5
2. Paragem de serviços	✓	5
3. Configuração das permissões	✓	5
4. Atalhos	✓	4
5. Utilização de um *software* livre de encargos	×	5
6. Eliminação das componentes do disco	✓	5
7. Janelas de execução	✓	2

Tabela 4 – Requisitos do *package* de instalação *MAC* do *Time Link*.

Estes requisitos foram também verificados e validados com o Historiador, o orientador da empresa e o com gestor de *software* da mesma. Após esta verificação e validação, passou-se à fase de levantamento de requisitos dos *softwares* de instalação a utilizar.

4.3.5. Levantamento de requisitos dos *softwares* de instalação

Nesta fase foi efetuado o levantamento dos requisitos que o *software* de instalação deveria preencher para o comportamento dos *packages* fosse de encontro aos nossos objetivos. Estes requisitos foram definidos e validados juntamente com o orientador da empresa, com o Historiador, e consoante o tipo de requisito (orientador e Historiador assumem papeis diferentes), para que o produto final fosse de encontro ao desejado. Passamos então a enumerar e descrever os mesmos.

1. *Software* livre de encargos – Existe uma variada gama de *softwares* de instalação para sistema operativo *Windows* e *MAC*, o que nos oferece várias possibilidades livres de pagamento.
2. Compressão – Visto que estes *packages* foram elaborados para serem disponibilizados na internet e posterior descarregamento, seria importante que oferecesse algum tipo de compressão.
3. Portabilidade entre sistemas operativos do mesmo tipo – Devido à existência de diversos sistemas operativos *Windows* e *MAC* a serem utilizados hoje em dia, era importante que este *software* de instalação fornecesse portabilidade entre os mesmos.
4. Instalação transparente para o utilizador – Devido aos vários pré-requisitos a serem instalados na máquina, é importante que esta instalação seja transparente para o utilizador e sem qualquer tipo de configuração necessária.

Na tabela seguinte podemos observar o tipo de requisitos (funcionais e não funcionais), classificando-os em termos de prioridade de implementação, numa escala de 0 a 5.

Requisitos	Funcional	Prioridade
***1. Software* livre de encargos**	×	5
2. Compressão	✓	3
3. Portabilidade	×	5
4. Transparência de instalação	×	5

Tabela 5 – Requisitos dos *softwares* de instalação *Windows* e *MAC*.

Após este levantamento foi efetuada uma análise ao *software* (*Windows* e *MAC*) de instalação para que este preenche-se os requisitos necessários permitindo alcançar os nossos objetivos, e que, no âmbito de projetos *open source,* fosse livre de encargos, seguindo assim toda a lógica do *Time Link*.

4.3.6. *Software* de instalação *Windows* analisado

Entre os diversos *softwares* disponíveis para elaboração deste *package* em sistema operativo *Windows*, vamos enumerar apenas os que consideramos ter mais potencial para atingir os nossos objetivos e explicar o porquê da opção tomada.

O *Advanced Installer* foi o primeiro *software* testado para a elaboração do *package* de instalação *Windows* devido à vasta gama de funcionalidades fornecidas, documentação base e facilidade de utilização. Apesar destas mais-valias apenas a versão *trial* deste *software* é livre de encargos e ainda assim só pode ser utilizada para avaliação, o que nos levou a procurar uma alternativa.[19]

O *software Install Editor* foi o *software* que se seguiu. Este permite a construção de um *package* de instalação MSI (sistema de instalação *Microsoft* para sistema operativo *Windows*), é também de fácil utilização e fornece funcionalidades que poderiam ser suficientes para a elaboração do *package*. Mas, tal como o *software* anterior, a versão completa não é livre de encargos, o que nos levou a procurar outra solução. [20]

O *software Inno Setup* foi uma das opções com mais potencial que analisámos. Este é baseado em *scripting*, o que aumenta a complexidade de desenvolvimento mas também oferece grande liberdade. Disponibiliza uma vasta gama de funcionalidades devido a esta complexidade, contém alguma documentação base e é totalmente livre de encargos. No entanto dentro deste tipo de *softwares* (baseados em *scripting*) temos várias opções, e conseguimos encontrar uma que em termos de documentação é mais completa. [21]

O *software* escolhido foi o NSIS (*Nullsoft Scriptable Install System*) pois este preenche todos os requisitos que definimos anteriormente. Tal como o anterior é baseado em *scripting*, o que fornece liberdade de implementação e uma vasta gama de funcionalidades, é *open source* totalmente livre de encargos, tem uma vasta documentação base devido à sua grande utilização, contém algoritmo de compressão e fornece portabilidade entre sistemas operativos *Windows*. [22]

Na tabela seguinte podemos observar um resumo da análise do *software* de instalação, segundo os requisitos identificados anteriormente.

Requisitos/*Software*	***Advanced Installer***	***Install Editor***	***Inno Setup***	***NSIS***
***1. Software* livre de encargos**	×	×	✓	✓
2. Compressão	✓	✓	✓	✓
3. Portabilidade	✓	✓	✓	✓
4. Transparência de instalação	✓	✓	✓	✓

Tabela 6 – Comparação entre *softwares* de instalação *Windows*.

Depois de efetuado o levantamento de requisitos foi então escolhido o *software* de instalação a utilizar, como referido anteriormente, o NSIS.

4.3.7. *Software* de instalação *MAC* analisado

Tal como para o *package* de instalação *Windows,* foram também analisados alguns *softwares* de instalação para o sistema operativo *MAC*, que passamos a descrever.

O *software VISE X* é provavelmente o melhor instalador para o sistema operativo *MAC*, permitindo a criação de um ficheiro binário com todas as componentes a serem instaladas. Para além deste aspeto, fornece uma vasta gama de funcionalidades, documentação base completa e bem organizada, e facilidade de utilização. Apesar destas mais-valias, este *software* não é livre de encargos, o que nos levou a continuar a nossa pesquisa. [23]

O *software Install Builder* é baseado na linguagem *Java* e tem a particularidade de ser *cross plataform*, o que permitiria a instalação em sistemas operativos diferentes (por exemplo *Windows* e *MAC*). Para além deste aspeto, fornece uma vasta gama de funcionalidades, documentação base bem organizada e facilidade de utilização. Apesar destas mais-valias, este *software* não é livre de encargos, o que nos levou a procurar uma solução diferente. [24]

O *software Iceberg* foi o escolhido para a implementação deste *package* de instalação. Apesar de ser especifico da plataforma *MAC*, permite portabilidade entre este tipo de sistemas operativos, oferece uma vasta gama de funcionalidades (por exemplo execução de *scripts* pré e pós-instalação), contém documentação base completa e bem organizada (exemplos de utilização e problemas encontrados), é de fácil utilização e diferencia-se dos anteriores na medida em que é *freeware,* ou seja, de utilização livre de encargos. Apesar de não conter suporte para desinstalação, conseguimos efetuar esta através de *scripts*, o que não se traduz propriamente numa limitação. Para além destes aspetos, este *software* utiliza o instalador nativo do sistema operativo *MAC*.[25]

Requisitos/*Software*	*VISE X*	*Install Builder*	*Iceberg*
1. Livre de encargos	×	×	✓
2. Compressão	✓	✓	✓
3. Portabilidade	✓	✓	✓
4. Transparência na instalação	✓	✓	✓

Tabela 7 – Comparação entre *softwares* de instalação *MAC*.

Depois de efetuado o levantamento de requisitos foi então escolhido o *software* de instalação a utilizar, como referido anteriormente, o *Iceberg*.

4.4. Requisitos – Importação GEDCOM

Nesta fase, através de testes efetuados pelo estagiário, contato com Historiadores que utilizam a aplicação e ideias da empresa, foram identificados os requisitos e processos a implementar (derivados dos requisitos) dentro do contexto da funcionalidade de importação GEDCOM, começando com uma breve descrição do *software*.

4.4.1. Descrição do *software*

Inicialmente o *Time Link* apenas permitia a importação de dados no formato *Kleio*, apesar da maioria das aplicações nesta área permitirem importar e exportar dados no formato GEDCOM. Assim sendo o objetivo desta funcionalidade é a implementação de um *parser* em *Java* que permita a conversão GEDCOM para XML.

A opção pelo formato XML deve-se ao facto da já existir um importador implementado na aplicação que recebe este tipo de ficheiros. Mais especificamente, os ficheiros *Kleio* são primeiro traduzidos para este formato segundo uma notação específica e depois importados. Gerando o ficheiro XML através do *parser Java*, e sendo este consistente com os gerados pelo tradutor *Kleio*, conseguimos importar a informação utilizando o importador já existente.

4.4.2. Casos de uso

Definido o objetivo desta funcionalidade vamos então identificar alguns casos de uso que nos ajudaram a definir os requisitos. Assim sendo, dentro do processo de requisitos foram identificados os seguintes casos através das técnicas já referidas anteriormente.

- O utilizador faz *upload* do ficheiro GEDCOM para a pasta local mhk_docs, pré-defina pelo *Time Link*;
- O utilizador traduz o ficheiro na pasta mhk_docs;
- O utilizador importa o ficheiro XML para a base de dados.

Tendo em conta estes casos de uso foi então elaborado o levantamento de requisitos do *software* a implementar.

4.4.3. Levantamento de requisitos

Antes de identificados os requisitos, é importante referir que existem várias versões do formato GEDCOM, sendo que a mais utilizada pelas aplicações anteriormente identificadas é a versão 5.5.1. Apesar da versão 6 seguir o formato *standard* do XML, esta não tem uma especificação completa, pelo que não é aconselhado o desenvolvimento de *software* de suporte à mesma.

Para esta fase do desenvolvimento de *software* foi efetuado o levantamento de requisitos por parte do estagiário, verificado e validado pelo gestor de *software* da iClio, João Carvalho.

1. *Upload* de ficheiros GEDCOM (extensão .ged) através do interface gráfico da página *Web* do *Time Link*, para a pasta local mhk_docs pré-definida pelo *Time Link*;
2. Tradução de ficheiros GEDCOM 5.5.1 (de codificação ANSEL, pois é o *standard* deste formato) para XML (na mesma pasta mhk_docs) através do interface gráfico da página *Web* do *Time Link*;
3. Importação dos ficheiros XML gerados utilizando o importador já implementado através do interface gráfico da página *Web* do *Time Link*.

Na tabela seguinte podemos verificar quais destes requisitos são funcionais e a sua prioridade. É importante referir que a escala atribuída à prioridade é idêntica à seguida até aqui, ou seja, entre 0 e 5.

Requisitos	Funcional	Prioridade
1. *Upload* GEDCOM	✓	3
2. Tradução GEDCOM 5.5.1	✓	5
3. Importação XML gerado	✓	5

Tabela 8 – Requisitos importação GEDCOM.

Para além deste requisitos gerais identificados, se tivermos em conta a especificação da versão GEDCOM 5.5.1, podemos ainda particularizar alguns requisitos a este nível, pois pode existir informação que não é suportada pelo modelo de dados ou pelo importador atual do *Time Link*. [26]

Como se pode verificar na especificação, esta versão do GEDCOM segue um modelo em árvore, sendo que cada linha do ficheiro tem o seguinte formato <*level*> <*tag*> <*value*>. Dentro de cada *level* existem várias *tags* definidas na especificação que permitem armazenar a informação.

Num primeiro nível (*level* 0), como é apresentado na especificação do GEDCOM 5.5.1, temos as principais *tags*:

- 0 <HEADER> – Contém informação sobre o ficheiro GEDCOM, como por exemplo o sistema de onde foi exportado.
- 0 <RECORD> – *Tag* que contém a maior parte da informação presente no ficheiro GEDCOM como por exemplo indivíduos, famílias etc.
- 0 <TRLR> – Significa o fim da informação GEDCOM presente no ficheiro.

Dentro da *tag* RECORD referida anteriormente, podemos encontrar novamente novas *tags* que distinguem o tipo de informação. Como podemos observar na especificação da versão 5.5.1 do GEDCOM, esta divide-se em:

- n <FAM_RECORD> – Contém informação sobre famílias. Mais especificamente a cada família é atribuído um id, e esta família contém ids dos seus membros, marido, mulher e filhos.
- n <INDIVIDUAL_RECORD> – Contém informação sobre os indivíduos. Mais especificamente a cada individuo é atribuído um id, pode conter variados atributos como nome, morada etc., e informação sobre eventos em que esteve presente (batismo, casamento, nascimento etc.). Estes atributos e eventos estão também especificados.
- n <MULTIMEDIA_RECORD> – Contém informação sobre recursos multimédia. Mais especificamente a cada recurso é atribuído um id e contém também vários atributos. Este id é referenciado através do individuo, pois este pode conter vários MULTIMEDIA_RECORDS.
- n <NOTE_RECORD> – Segue a mesma lógica da *tag* MULTIMEDIA_RECORD. Contém informação sobre indivíduos, a cada nota é atribuído um id e estas notas são referenciadas a partir dos indivíduos.
- n <REPOSITORY_RECORD> – Contém informação sobre a localização de uma ou várias fontes.
- n <SOURCE_RECORD> – Contém informação sobre a fonte de dados genealógicos GEDCOM.
- n <SUBMITTER_RECORD> – Contém informação sobre o *submitter* da informação GEDCOM.

Definidos as *tags* de mais alto nível, podemos identificar quais contêm informação que consideramos útil para o *Time Link* e consistente com o modelo de dados do mesmo. Assim sendo temos:

- HEADER – Podemos tratar como informação da fonte. Não está representada esta entidade do diagrama UML, mas é criada uma entrada na mesma para cada ficheiro importado no *Time Link*.

- FAM_RECORD – Apesar to *Time Link* não conter entidades de suporte a famílias, estas são representadas como relações de parentesco (inseridas na entidade relações). Estas famílias podem ser decompostas em relações deste tipo.
- INDIVIDUAL_RECORD – Insere-se diretamente como uma nova pessoa, com atributos, relações e eventos (esta última correspondente à entidade Ato no *Time Link*).
- MULTIMEDIA_RECORD – Apesar de poder ser inserido com um atributo genérico, este *record* contém vários atributos. Teria de ser criada uma nova tabela na base de dados e alterado importador.
- NOTE_RECORD – Podemos inserir como atributo de um individuo pois este *record* apenas contém o valor da própria nota.
- REPOSITORY_RECORD – Não temos suporte no modelo de dados para esta informação.
- SOURCE_RECORD – Tal como a *tag* HEADER, pode ser traduzida em informação da fonte.
- SUBMITTER_RECORD – Poderia ser inserido como um individuo, mas apenas está relacionado com fonte de informação.

Esta análise, validada com o Doutor Joaquim Ramos de Carvalho e com o gestor de *software* da empresa deu então origem a requisitos particulares da funcionalidade de tradução GEDCOM para XML que podemos observar na tabela seguinte, classificados como funcionais e não funcionais, e com prioridade na escala de entre 0 e 5.

Requisitos GEDCOM	Funcional	Prioridade
1. Tratamento *tag* HEADER	✓	3
2. Tratamento *tag* FAM_RECORD	✓	5
3. Tratamento *tag* INDIVIDUAL_RECORD	✓	5
4. Tratamento *tag* NOTE_RECORD	✓	4
5. Tratamento *tag* SOURCE_RECORD	✓	3

Tabela 9 – Requisitos particulares da tradução GEDCOM.

Apesar de algumas *tags* não serem tratadas, desta forma conseguimos abranger a informação mais importante presente nos ficheiros GEDCOM.

4.5. Requisitos – Famílias

Tal como na funcionalidade anterior, foram identificados os requisitos e processos a implementar dentro do contexto da funcionalidade de famílias, começando com uma breve descrição do *software*.

4.5.1. Descrição do *software*

Mais especificamente com esta funcionalidade o utilizador consegue agora visualizar a árvore genealógica de um individuo genérico. Esta informação sobre famílias está decomposta na base de dados segundo relações do tipo individuo x é pai de y ou individuo y é filho de x, pelo que foi implementado um algoritmo o que calcule as famílias presentes na base de dados para armazenamento numa nova tabela de famílias. A partir desta informação de famílias é então calculada a árvore genealógica de um individuo genérico, e apresentada ao utilizador por meio de uma biblioteca *JavaScript*.

4.5.2. Casos de uso

Definido objetivo da funcionalidade a implementar, vamos então identificar alguns casos de uso que nos ajudaram no processo de identificação de requisitos.

- O utilizador está na página *Web* de um individuo genérico e pretende visualizar a sua árvore genealógica;
- O utilizador pretende limitar o nível da árvore (superior e inferior);
- O utilizador pretende explorar a árvore genealógica gerada.

Tendo em conta estes casos de uso foi então elaborado o levantamento de requisitos do *software* através do qual foi desenvolvida a solução.

4.5.3. Levantamento de requisitos

Antes de definir os requisitos para esta funcionalidade, é importante referir que para a visualização da árvore genealógica no *Web Browser* optou-se por utilizar a biblioteca *JavaScript* FamilyTree.js própria para este tipo de representações. [27]

Apesar da biblioteca *JavaScript* ser implementada especificamente para árvores de famílias, esta contém algumas limitações que podemos ultrapassar através da apresentação de informação adicional nos nós da árvore. Por exemplo, uma das limitações desta biblioteca verifica-se quando temos o caso de um homem x que teve filhos com a mulher y e com a mulher z. Nesta situação não é possível distinguir graficamente quem é mãe de quem, e o mesmo se verifica no caso contrário, em que uma mulher x teve filhos com o homem y e com o homem z.

Apesar deste tipo de limitações, através desta informação adicional, como por exemplo relações do tipo: tem como pai ou tem como filho, conseguimos transmitir a informação correta ao utilizador. Assim sendo, para esta fase de desenvolvimento de *software* foi efetuado o levantamento de requisitos pelo estagiário, verificado e validado pelo gestor de *software* da iClio, João Carvalho.

1. Implementação de um algoritmo para o cálculo de famílias;
2. Criação de uma nova tabela famílias na base de dados para armazenamento da informação calculada, contendo os ids dos indivíduos e os seus nomes;
3. Escrita das famílias na base de dados na tabela de famílias criada;
4. Implementação de um algoritmo que calcule e devolva a árvore genealógica de um determinada pessoa, recebendo o seu id;
5. Apresentação da árvore genealógica do individuo na sua página Web respetiva às famílias, utilizando a biblioteca FamilyTree.js.

Na tabela seguinte podemos verificar quais destes requisitos são funcionais e a sua prioridade. É importante referir que a escala atribuída à prioridade é idêntica à seguida até aqui, ou seja, entre 0 e 5.

Requisitos	Funcional	Prioridade
1. Algoritmo cálculo famílias	✓	5
2. Tabela de famílias	✓	5
3. Escrita das famílias	✓	5
4. Algoritmo cálculo árvore	✓	5
5. Apresentação da árvore	✓	5

Tabela 10 – Requisitos famílias.

Os requisitos identificados foram validados pelo gestor de *software* de empresa, dando então início às fases de arquitetura e desenvolvimento da solução.

4.6. Requisitos – Edição *inline*

Tal como nas funcionalidades anteriores, foram identificados os requisitos e processos a implementar dentro do contexto da funcionalidade de edição *inline*, começando com uma breve descrição do *software*.

4.6.1. Descrição do *software*

Mais especificamente implementou-se uma solução que permite ao utilizador alterar os ficheiros já presentes na sua pasta mhk_docs (no nosso caso está presente no cliente, pois o cliente e o servidor são a mesma máquina) e gravar os mesmos no disco. Devido ao facto desta notação *Kleio* ter uma sintaxe bem definida, introduzimos também o *highlight* (palavras ou frases a cores) em tempo real configurado para esta notação, permitindo a leitura de ficheiros *Kleio* mais transparente e intuitiva para o utilizador. Existem várias ferramentas disponíveis que permitem configuração da sintaxe a tratar, pelo que no nosso caso decidi-mos utilizar a biblioteca *JavaScript* CodePress que é especializada no *highlight* de linguagens de programação, mas permite configuração de novos módulos. [28]

4.6.2. Casos de uso

Dentro do processo de requisitos foram identificados os seguintes casos de uso, através das técnicas já referidas anteriormente.

- O utilizador está na página *Web* de visualização de ficheiros e pretende alterar ou visualizar o seu conteúdo clicando no nome correspondente;
- O utilizador pretende guardar o ficheiro alterado na mesma pasta escolhendo o nome do ficheiro.

Tendo em conta estes casos de uso foi então elaborado o levantamento de requisitos do *software* desta funcionalidade.

4.6.3. Levantamento de requisitos

Para esta fase de desenvolvimento foi efetuado o levantamento de requisitos pelo estagiário, verificados e validados pela empresa.

1. Visualização de ficheiros *Kleio* na página *Web* do *Time Link*;
2. Edição de ficheiros *Kleio* na página *Web* do *Time Link*;
3. Armazenamento local através do servidor (pois este está na mesma máquina que o cliente) dos ficheiros editados na pasta mhk_docs;
4. Criação de um novo módulo na biblioteca *JavaScript* CodePress configurado para a notação *Kleio*.

Na tabela seguinte podemos verificar quais destes requisitos são funcionais e a sua prioridade. É importante referir que a escala atribuída à prioridade é idêntica à seguida até aqui, ou seja, entre 0 e 5.

Requisitos	Funcional	Prioridade
1. Visualização	✓	5
2. Edição	✓	5
3. Armazenamento	✓	5
4. *Highligh* da sintaxe	✓	3

Tabela 11 – Requisitos edição *inline*

Os requisitos identificados foram validados pelo gestor de *software* de empresa, dando então início ás fases de arquitetura e desenvolvimento da solução.

4.7. Requisitos – Introdução de dados no *Web Browser*

Seguindo a estrutura dos pontos anteriores, foram identificados os requisitos e processos a implementar dentro do contexto da funcionalidade de introdução de dados no *Web Browser*, começando com uma breve descrição do *software*.

4.7.1. Descrição do *software*

Como tem sido referido até aqui, o *Time Link* inicialmente apenas permitia a introdução de dados na aplicação através de ficheiros *Kleio*, e agora também de ficheiros GEDCOM. Apesar da importação deste novo formato de ficheiros, o utilizador está ainda obrigado à aprendizagem da linguagem *Kleio* para produção destes tipos de ficheiros, ou elaboração/exportação de ficheiros GEDCOM através de outras aplicações. Este facto levou-nos a introduzir esta funcionalidade, por forma a abstrair o utilizador tanto da linguagem *Kleio* como do formato GEDCOM.

Assim sendo, com esta funcionalidade o utilizador pode criar fontes de informação através de uma página *Web* do *Time Link*. Estas fontes contêm pessoas, atributos, relações e atos ou eventos com pessoas e funções associadas, que após a criação do documento são exportadas para XML (segundo o mesmo formato utilizado pelo *parser* GEDCOM) e importadas na base de dados.

4.7.2. Casos de uso

Dentro do processo de requisitos foram novamente identificados casos de uso, através das técnicas já referidas anteriormente.

- O utilizador está na página *Web* de criação de fontes e pretende adicionar/eliminar pessoas, atributos sobre essas pessoas, relações entre as mesmas, atos e funções;
- O utilizador pretende importar a fonte de informação criada na base de dados.

Tendo em conta estes casos de uso foi então elaborado o levantamento de requisitos do *software* a desenvolver.

4.7.3. Levantamento de requisitos

Para esta fase de desenvolvimento foi efetuado o levantamento de requisitos pelo estagiário, verificados e validados pela empresa.

1. Página *Web* que permita a produção e importação de fontes de informação, nomeadamente adicionar pessoas, atributos, relações, atos e funções;
2. Criação do ficheiro XML numa só ação com a informação adicionada à fonte, num formato consistente com o gerado na tradução *Kleio* e GEDCOM e armazenado na pasta mhk_docs do *Time Link*.

3. Importação do ficheiro XML automática após a criação do mesmo utilizando o importador da aplicação.

Na tabela seguinte podemos verificar quais destes requisitos são funcionais e a sua prioridade. É importante referir que a escala atribuída à prioridade é idêntica à seguida até aqui, ou seja, entre 0 e 5.

Requisitos	Funcional	Prioridade
1. Criação de fontes	✓	5
2. Criação do XML	✓	5
3. Importação do XML	✓	5

Tabela 12 – Requisitos introdução de dados no *Web Browser*.

Os requisitos identificados foram validados pelo gestor de *software* de empresa, dando então início ás fases de arquitetura e desenvolvimento da solução.

5. Arquitetura

Nesta fase vamos apresentar a arquitetura do *software* desenvolvido durante o estágio, tendo em conta a descrição das funcionalidades e os seus requisitos anteriormente especificados.

5.1. Arquitetura – *Packages* de instalação

Relativamente ao desenvolvimento dos *packages* de instalação (*Windows* e *MAC*), foram elaboradas as arquiteturas respetivas que passamos a descrever nos próximos pontos.

5.1.1. Arquitetura do *package* de instalação *Windows*

O *software* NSIS é baseado em *scripting*, o que permite criar uma lógica complexa dependendo do tipo de instalação que se pretende implementar. Como o *package* a ser elaborado requer alguma complexidade, nomeadamente na execução de *scripts*, com este *software* consegue-se alcançar os resultados pretendidos.

Na figura seguinte podemos observar a arquitetura do *package* de instalação, dividida por secções, sendo que se pode equivaler cada uma destas a um procedimento de uma linguagem de programação dentro do paradigma procedimental. Neste caso, a execução destes procedimentos é sequencial.

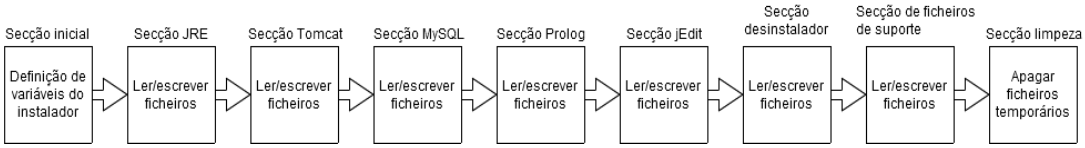

Figura 6 – Arquitetura do *package* de instalação *Windows*.

Durante a compilação do *package* de instalação, em cada secção de leitura/escrita são lidos os ficheiros da máquina de desenvolvimento. As diretorias nas quais a aplicação irá ser instalada na máquina do utilizador são definidas na secção inicial, sendo esta definição utilizada na secção de limpeza durante a desinstalação. Durante a instalação, são escritos os ficheiros necessários na máquina do utilizador, e apagados os ficheiros temporários.

5.1.2. Arquitetura do *package* de instalação *MAC*

Apesar do *software* utilizado para a elaboração deste *package* de instalação não ser baseado em *scripting*, pois fornece um interface gráfico onde podemos adicionar e retirar componentes, internamente o seu funcionamento é semelhante, pelo que podemos também dividir esta instalação em secções, mas sem qualquer garantia sobre a ordem em que as componentes são instaladas. Como este *software* fornece um interface gráfico para a elaboração do *package* não é necessária a elaboração de código para a sua construção. Na figura seguinte podemos observar a arquitetura do mesmo.

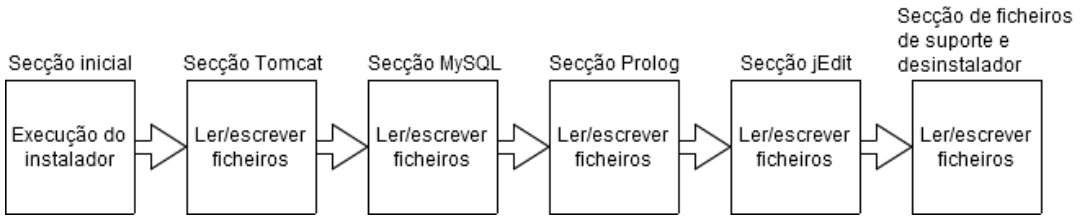

Figura 7 – Arquitetura do *package* de instalação *MAC*.

Durante a *build* do *package* de instalação, em cada secção de leitura/escrita são lidos os ficheiros da máquina de desenvolvimento. As diretorias nas quais a aplicação irá ser instalada na máquina do utilizador são definidas no interface gráfico e durante a instalação são escritos os ficheiros necessários na máquina do utilizador segundo o instalador nativo do sistema operativo *MAC*.

5.2. Arquitetura – Importação GEDCOM

Neste ponto vamos descrever a arquitetura da solução relativa à importação GEDCOM, tendo em conta a forma como esta foi integrada no *Time Link*. Como foi referido anteriormente na arquitetura de sistema, esta aplicação disponibiliza uma *thread* onde são executadas tarefas mais custosas em termos temporais, como a tradução GEDCOM para XML.

Para integrarmos esta nova componente, temos apenas de implementar a classe ou classes que resolvem o problema (tradução) e executar estas a partir de uma nova classe Worker também implementada. Com estas classes temos apenas de registar o novo Worker na *thread* e agendar a tarefa na mesma.

Após definida a forma como foi integrada a solução no *Time Link*, podemos então descrever a arquitetura da solução em si. Na resolução do problema, para além da alteração de algumas páginas HTML (*views*) e de algumas outras classes presentes na arquitetura, foram criadas duas adicionais (GedcomParser e o Worker correspondente). Na figura seguinte podemos visualizar a arquitetura da solução implementada.

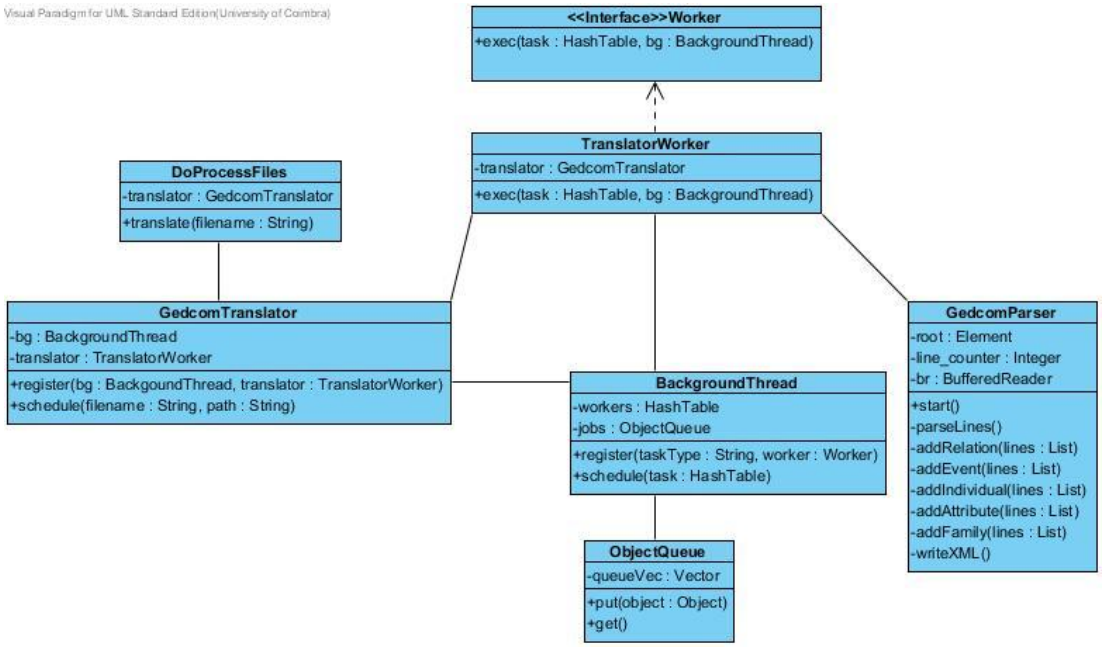

Figura 8 – Arquitetura da solução GEDCOM.

É importante referir que nas classes BackgroundThread e ObjectQueue não foi necessário efetuar qualquer alteração pertencendo estas ao estado inicial do *Time Link*.

5.3. Arquitetura – Famílias

Neste ponto vamos descrever a arquitetura da solução implementada na funcionalidade de famílias onde, tal como na funcionalidade anterior, foi utilizada a *thread* disponível para tarefas custosas em termos temporais, nomeadamente o cálculo de famílias.

Tal como no desenvolvimento de *software* anterior, foram implementadas as classes que resolvem o problema (cálculo de famílias) e instanciadas a partir de uma nova classe Worker também implementada. Com estas classes temos apenas de registar o novo Worker na *thread* e agendar a tarefa na mesma. É importante também referir que o cálculo das famílias é efetuado após a importação de dados na aplicação, de forma a garantir a atualização da informação no sistema.

Após definida a forma como foi integrada a solução no *Time Link*, podemos então descrever a arquitetura da solução em si. Na resolução do problema, foram criadas três classes adicionais (FamilyManagement, que faz o cálculo da árvore e devolve a mesma, o Worker que calcula as famílias e grava na base de dados e a classe DoExecFamily que dá início a todo o processo de criação da árvore). Na figura seguinte podemos visualizar a arquitetura da solução implementada.

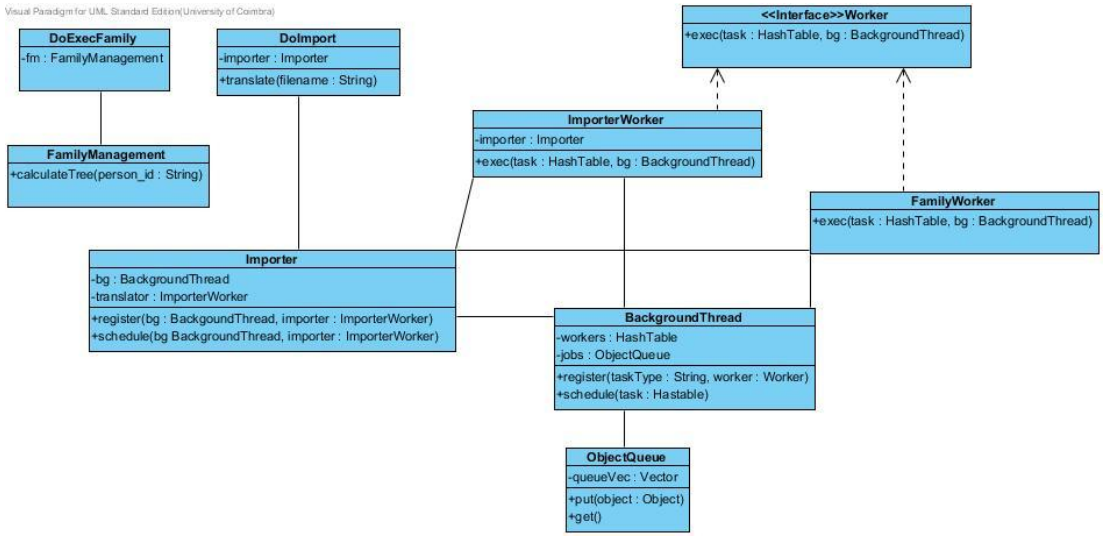

Figura 9 – Arquitetura solução famílias.

A arquitetura desta solução segue a mesma lógica da tradução GEDCOM, pois é novamente criado um Worker (FamilyWoker), sendo este registado e agendado na *thread* para execução. É importante referir também que, dependendo do tipo de ação a executar, cálculo de famílias ou cálculo da árvore genealógica, são instanciadas as classes DoImport e DoExecFamily, respetivamente.

5.4. Arquitetura – Edição *Inline*

Seguindo a estrutura dos pontos anteriores, vamos descrever a arquitetura da solução implementada relativa à funcionalidade de edição *inline*.

Para esta solução foram implementadas duas novas classes *Java*, DoEditFiLe e DoSaveFile que respondem aos pedidos do utilizador aquando a visualização do ficheiro e o armazenamento, respetivamente. Foi também implementado um novo módulo para a biblioteca CodePress a importar para a página de visualização e edição do ficheiro.

Na figura seguinte podemos observar a arquitetura da solução tendo em conta que uma destas duas classes é instanciada a partir da *Servlet* mediante o tipo de pedido ao servidor.

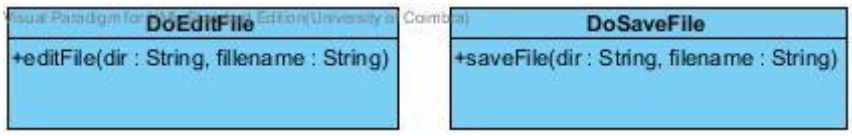

Figura 10 – Arquitetura edição *inline*.

Após esta fase foi então elaborado o módulo de *highlight* de sintaxe utilizando a biblioteca *JavaScript* CodePress. Esta biblioteca permite a introdução de módulos para qualquer tipo de linguagem ou notação implementados em *JavaScript* e a serem importados na página *Web* da aplicação. Para o módulo *Kleio* foram criados dois ficheiros ou bibliotecas (kleio.js e kleio.css), que contêm as expressões

regulares e as cores a utilizar, respetivamente. Na fase de implementação iremos descrever todo o processo de edição e desenvolvimentos efetuados.

5.5. Arquitetura – Introdução de dados no *Web Browser*

Seguindo a estrutura dos pontos anteriores, vamos descrever a arquitetura da solução implementada relativa à funcionalidade de introdução de dados no *Web Browser*.

Foram implementadas três novas classes *Java*, DoCreateSource e DoSaveSource que respondem aos pedidos do utilizador aquando a criação da fonte e a importação do ficheiro XML, respetivamente, sendo que a classe SaveWebSource produz o ficheiro XML resultante da informação introduzida.

Na figura seguinte podemos observar a arquitetura da solução tendo em conta que as classes DoCreateSource e DoSaveSource são instanciadas a partir da *servlet* consoante o tipo de pedido ao servidor, e a classe SaveWebSource é instanciada a partir da classe DoSaveSource para criação do ficheiro XML.

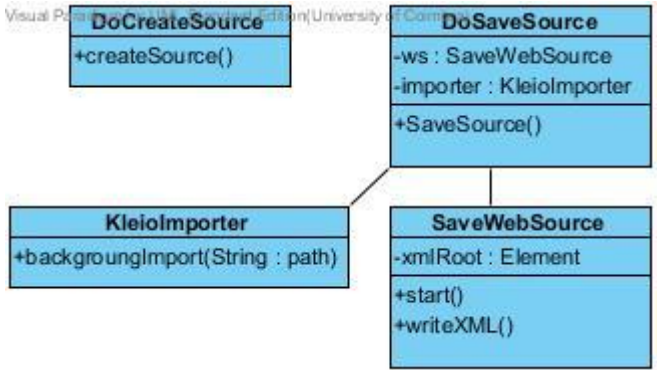

Figura 11 - Arquitetura introdução de dados no *Web Browser*.

É importante também referir que, que a classe de importação (KleioImporter) é instanciada a partir da classe DoSaveSouce após a criação do ficheiro XML, registando e agendando a tarefa na *thread*.

6. Implementação

Após as fases de requisitos e arquitetura vamos então apresentar a implementação dos *packages* de instalação e das funcionalidades planeadas para este estágio.

6.1. Implementação – *Packages* de instalação

A implementação dos *packages* de instalação é descrita neste ponto, dividido para cada tipo de instalador desenvolvido. É importante referir que para além da implementação do instalador em si, são também descritos os *scripts* utilizados para resolver o problema (em sistema operativo *Windows* e *MAC*).

6.1.1. Implementação de *scripts* do *package* de instalação *Windows*

Para além dos pré-requisitos já descritos é importante que seja transparente para o utilizador o início e paragem de alguns serviços que a aplicação utiliza. Assim sendo vamos descrever os ficheiros de suporte ou *scripts* utilizados em sistema operativo *Windows*.

Neste *package* foram desenvolvidos *scrips* na linguagem *batch* que permitem iniciar os serviços necessários e a paragem dos mesmos, bem como definição das variáveis de ambiente necessárias. No processo de paragem é preciso ter em conta que os serviços consomem recursos na máquina do utilizador mesmo quando este não está a utilizar a aplicação, o que torna muito importante a transparência desta ação. Assim sendo passamos a enumerar os ficheiros (*scripts*) *batch* e os seus propósitos.

- *Time Link startup* – Este *script* permite ao utilizador iniciar os serviços de base de dados e servidor. Posteriormente abre a página *Web* local do *Time Link* no *Web Browser* pré-definido da máquina. Este ficheiro é convertido para um executável (extensão .exe) e é criado um atalho no ambiente de trabalho que permite a sua execução com duplo-clique.
- *Time Link shutdown* – Este *script* permite a paragem dos serviços de base de dados e servidor para que não consumam recursos na máquina. Quando a paragem terminar é exibida uma mensagem informativa ao utilizador. Este ficheiro é convertido para um executável (extensão .exe) e é criado um atalho no ambiente de trabalho que permite a sua execução com duplo-clique.
- *SetJavaHome* – Este *script* é executado aquando a instalação do *Java Runtime Environment,* definindo assim a localização da variável de ambiente JAVA_HOME na máquina. Esta configuração é necessária pois o servidor utilizado é executado sobre este ambiente (*Java Runtime Environment*).

Com estes ficheiros de suporte conseguimos alguma consistência e transparência de utilização, pois normalmente as aplicações *Windows* utilizam atalhos para dar início à sua execução e desinstalação.

É importante também referir que alterando as componentes de servidor ou base de dados para versões diferentes (normalmente mais recentes), caso a forma de execução destes serviços seja idêntica, os *scripts* de início e paragem de serviços continuam funcionais, caso contrário terão que ser alterados.

6.1.2. Implementação do *package* de instalação *Windows*

Inicialmente começou-se pela leitura da documentação base e estudo de exemplos de instalação com estes *softwares*, de modo a agilizar o processo de implementação.

Após esta fase foram desenvolvidos dois *scripts* (segundo as normas NSIS): de instalação, que escrevem todos os pré-requisitos e ficheiros de suporte na máquina do utilizador, e de desinstalação, que remove todos esses ficheiros e pré-requisitos da máquina. Cada *script* desenvolvido produz um ficheiro executável (extensão .exe), sendo que o ficheiro de desinstalação é incluído no ficheiro de instalação, por forma a ficar presente na máquina do utilizador. Na figura seguinte podemos observar a estrutura do *package* de instalação.

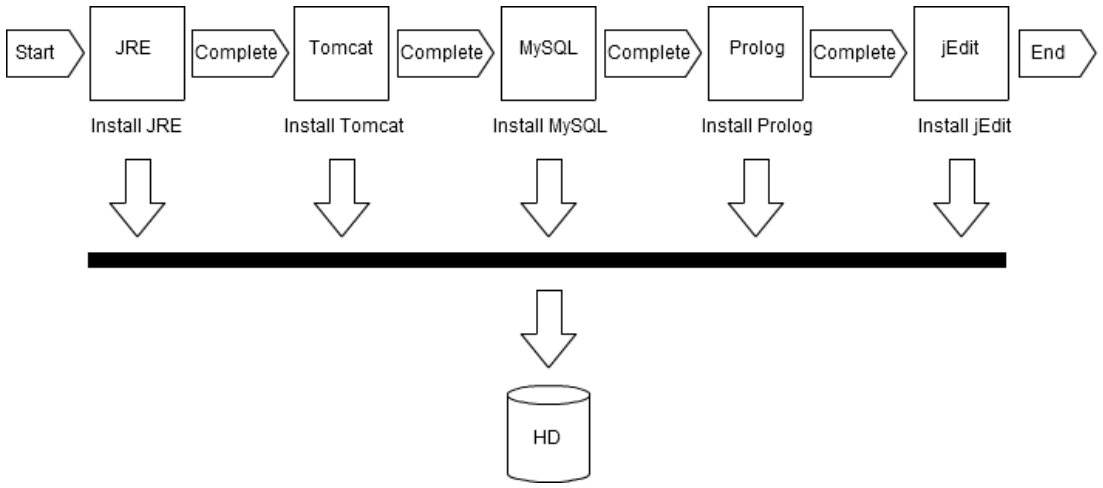

Figura 12 – Diagrama do *package* de instalação *Windows*.

Inicialmente, na instalação de alguns dos pré-requisitos, era utilizada a versão de instalação que utiliza o *Windows installer* (executável), necessitando de configuração por parte do utilizador. Esta solução não foi a melhor, pois exigia configurações por parte do mesmo aquando a instalação do pré-requisito em questão. Este foi um dos grandes desafios nesta fase, pois, seria necessário substituir esta instalação por uma transparente, ou seja, já configurada e independente da máquina.

A fase posterior, que levou a vários problemas no início e paragem de serviços, pois deveria ser transparente para o utilizador. Com os ficheiros de suporte incluídos no *package* e já descritos anteriormente, conseguimos alcançar os objetivos definidos.

Em casos de exceção durante a execução do instalador, o próprio *software* de instalação fornece suporte para abortar a instalação ou ignorar o problema.

6.1.3. Implementação de *scripts* do *package* de instalação *MAC*

Tal como no *package* anterior, nesta versão foram também desenvolvidos alguns *scripts* (neste caso na linguagem *Shell Script*) com o objetivo de iniciar e parar os serviços necessários à execução do *Time Link*, e desinstalar as componentes da máquina. Para além destes, temos também um *script* que configura as permissões necessárias para a execução do serviço de base de dados. Passamos então a enumerar estes *scripts* e as suas funções.

- Start Time Link – Este *script* é semelhante ao implementado para o *package Windows* e assim sendo tem como objetivo iniciar os serviços (base de dados e servidor) necessários para a execução do *Time Link*, com permissões de administrador. As permissões deste ficheiro são alteradas de modo a permitir a sua execução com duplo-clique por parte do utilizador.
- Stop Time Link – Também semelhante ao *script* implementado no *package Windows,* tem como objetivo a paragem dos serviços (base de dados e servidor) necessários à execução do *Time Link*. Tal como no *script* anterior, são alteradas as suas permissões de modo permitir a sua execução com duplo-clique.
- Install_db – A função deste *script* é a configuração das permissões necessárias à execução de serviços da base de dados bem como o *path* para o mesmo.
- jEdit launch – Este script funciona como um atalho para a componente jEdit. Verificou-se como necessária a sua implementação, pois esta componente tem configurações (como *templates*, *plugins*, etc.) previamente definidas por utilizadores do *Time Link*. Estas configurações necessitam de ser copiadas para diretoria correta, e no ato de execução da componente, pois o utilizador pode apagar esta pasta ou instalar outra versão da mesma. É importante referir que caso a pasta já exista na máquina esta cópia não é efetuada, mas caso esta não esteja presente é copiada para a diretoria correta.
- Time Link uninstaller – Este *script* tem apenas a função de desinstalar as componentes instaladas pelo *package*, pois algumas destas estão em diretorias distintas, o que exigia algum trabalho por parte do utilizador. Conseguimos assim agilizar este processo.

É importante também referir que, no caso de alteração das componentes de servidor ou base de dados para versões diferentes, se a forma de execução destes serviços for idêntica, os *scripts* de início e paragem de serviços continuam funcionais, caso contrário terão que ser alterados, tal como no *package* anterior.

6.1.4. Implementação do *package* de instalação *MAC*

Nesta fase de implementação foram adicionadas as componentes necessárias para o bom funcionamento do *Time Link,* bem como os ficheiros de suporte (*scripts*) necessários. Todas estas componentes foram adicionadas através do interface gráfico do *software* de instalação, definindo as diretorias destino das mesmas na máquina.

Após esta fase, é efetuada a *build* do *package*, que resulta num ficheiro único de extensão .pkg que contém toda a informação a ser instalada na máquina do utilizador. Este ficheiro, depois de executado, utiliza o instalador nativo do sistema operativo *MAC*, permitindo ao utilizador visualizar um interface gráfico consistente com outras aplicações para este sistema. Na figura seguinte podemos observar a estrutura do *package* de instalação.

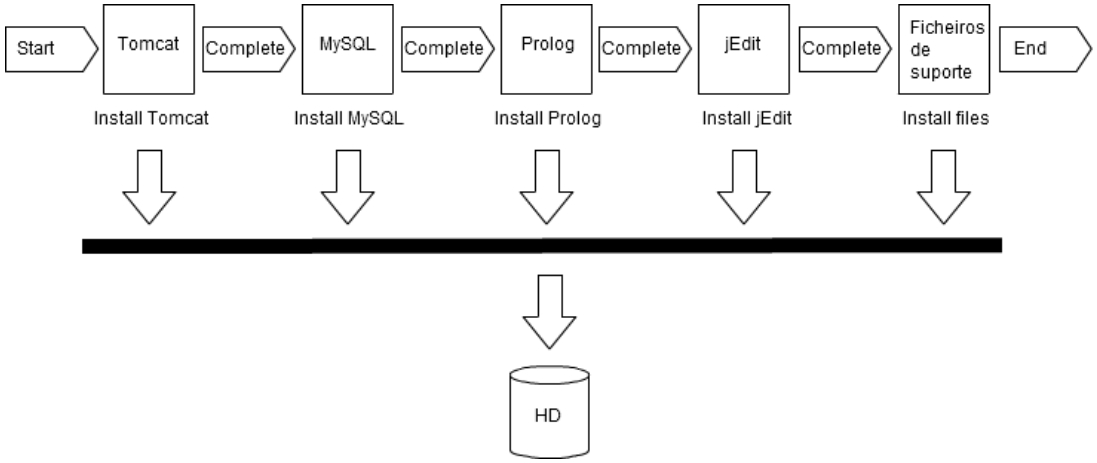

Figura 13 – Diagrama do *package* de instalação *MAC*.

Como no *package* de instalação para *Windows* verificámos que a utilização de instalações em formato "executável" necessitava de configuração por parte do utilizador. Assim sendo, na versão *MAC* foi seguida uma abordagem idêntica, utilizando uma instalação das componentes já configuradas.

Seguindo a mesma abordagem do *package* de instalação para *Windows*, com os *scripts* elaborados, conseguimos uma utilização transparente da aplicação, no que toca ao início e paragem de serviços, alcançando assim os objetivos previamente definidos.

É importante também referir que no caso de ocorrerem exceções durante a execução do instalador, estas ficam a cargo *software* nativo de instalação do sistema operativo *MAC*.

6.2. Implementação – Importação GEDCOM

Na fase de implementação desta solução anteriormente definida, foi analisada a forma como a informação contida no ficheiro GEDCOM pode ser traduzida para XML de uma forma consistente

com o formato gerado pela tradução *Kleio*. A par da implementação do *parser,* esta foi a tarefa mais complexa pois existe grande variedade de informação nos ficheiros GEDCOM, e sua tradução para um XML consistente com o gerado pela tradução *Kleio* deve ser realizada de forma a perder o mínimo de informação possível.

Como está definido nos requisitos deste *software*, existem várias *tags* no ficheiro GEDCOM que vão ser tratadas com o objetivo de gerar o XML correspondente. Cada uma destas *tags* pode conter uma vasta variedade de outras *tags* (atributos, relações, etc. um *level* acima), como está definido na especificação do GEDCOM 5.5.1. Por exemplo, dentro do *record* INDIVIDUAL_RECORD que corresponde à *tag* INDI no ficheiro GEDCOM, podemos encontrar *tags* que representam atributos do individuo (sexo, id, etc.), sendo que o *level* destas *tags* contidas vai ser sempre superior ao *level* do individuo (por exemplo, se individuo *level* 1 os seus atributos vão começar no *level* 2), sendo que estas *tags* podem ainda subdividir-se sem qualquer limite de *level*. Desta forma conseguimos delimitar onde começa e acaba a informação sobre cada *tag*.

É importante também referir que existem atributos de cardinalidade 1 para 1 (por exemplo, no caso do individuo este só pode ter um id), de 0 para n (por exemplo um individuo pode ter várias notas ou nenhuma) e de 0 para 1 (no caso de atributos não obrigatórios). Definido então o problema a resolver, vamos descrever a abordagem seguida para a sua resolução.

Numa descrição de mais alto nível, optou-se pela implementação de uma classe *Java* denominada GedcomParser que recebe a localização do ficheiro a traduzir e escreve o resultado (ficheiro XML traduzido) na pasta local mhk_docs, pré-definida pelo *Time Link*. A um nível mais baixo, e seguindo a lógica descrita anteriormente, o *parser* implementado lê as linhas do ficheiro GEDCOM até encontrar uma *tag* que seja tratável para o nosso caso. Assim que uma destas *tags* é encontrada, são lidas e guardadas temporariamente todas as linhas contidas nessa *tag* (ou seja, de *level* superior). Estas linhas são então enviadas para o método correspondente que faz o *parsing* da *tag*. Este processo é seguido para todas as *tags* a serem tratadas pelo *parser,* de forma genérica sempre que possível.

À medida que o *parsing* é efetuado, é escrito num ficheiro local o relatório da tradução que ajuda a visualizar informação resultante do *parsing*, e é também efetuada a construção do ficheiro XML, sendo este escrito no disco no final do processo. Para a criação deste ficheiro XML, utilizou-se a biblioteca *Java* JDOM muito utilizada e que contém vasta documentação. [29]

Após o desenvolvimento do *parser* foram implementadas as classes *Java* TranslatorWorker e GedcomTranslator que dão início ao processo de *parsing*. Na figura seguinte podemos observar o funcionamento da solução implementada.

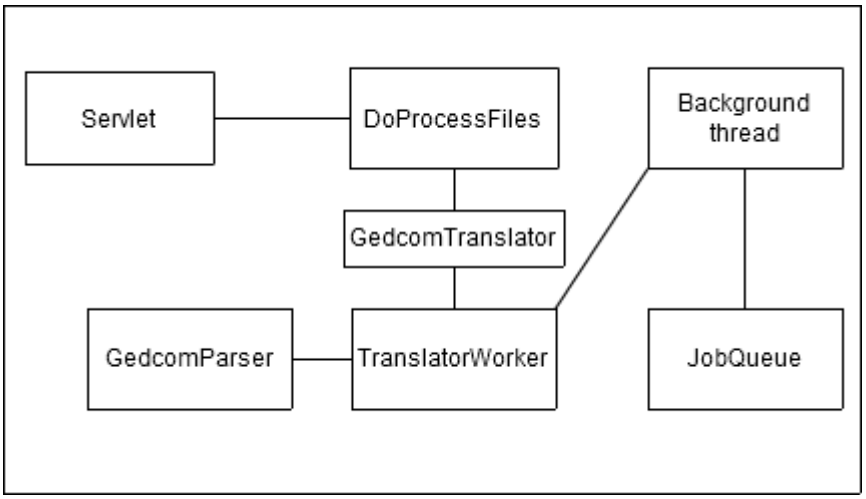

Figura 14 – Diagrama de funcionamento solução GEDCOM.

Como podemos verificar no diagrama anterior, os pedidos efetuados ao servidor são numa primeira fase tratados pela classe *Java* do tipo *Servlet*. Esta redireciona o pedido para a classe correspondente (neste caso DoProcessFiles), que cria uma instancia da classe GedcomTranslator. Esta sua vez cria uma instância da classe TranslatorWorker que contém um método exec (implementado pela interface Worker) dando inicio ao processo de tradução (incluído na classe GedcomParser). Após a criação do Worker, é efetuado o registo do mesmo na *thread*, e agendada a tradução através da classe GedcomTranslator. É importante também referir que os ficheiros XML e relatório de tradução resultantes são escritos no disco pela classe GedcomParser no final deste processo.

Como foi referido anteriormente, esta *thread* contém uma fila de tarefas FIFO (JobQueue) que permite a gestão e agendamento de tarefas a executar, estando esta gestão a cargo da classe BackgroundThread.

6.3. Implementação – Famílias

Após as fases de requisitos e arquitetura, foi então analisado o algoritmo de cálculo de famílias a utilizar, tendo em conta o formato do resultado esperado para posterior armazenamento na tabela famílias na base de dados. Assim sendo, cada entrada nesta tabela contém informação sobre três indivíduos (terceto), pai, mãe e filho. Ou seja para cada pessoa existente na base de dados, temos então de calcular todas as famílias em que é pai ou mãe e em que é filho, caso exista esse tipo de informação, construindo assim todas as famílias dos indivíduos. Efetuado o cálculo destes tercetos (pai mãe e filho), é então armazenada a informação relativa aos ids de cada individuo e os seus nomes.

Após a implementação do cálculo de famílias anterior, foram então armazenadas as mesmas na base de dados e iniciada a implementação do algoritmo que calcula a árvore genealógica de um

individuo genérico. A partir desta informação presente na base de dados, temos apenas que recursivamente percorrer as ligações entre filhos, pais e mães. É importante referir que neste cálculo é armazenado, a cada o nó, o seu nível (por exemplo, o individuo sobre o qual é efetuado o cálculo é de nível 0, enquanto os seus pais e filhos são de nível 1 e -1, respetivamente). Esta informação é necessária devido à forma como a biblioteca *JavaScript* utilizada funciona, pois esta começa por construir o nível mais alto da árvore e a partir daí adiciona filhos aos nós. É importante referir que,tal como o cálculo de famílias, este algoritmo tem em conta não só a informação GEDCOM, mas também a informação *Kleio*. Implementado o cálculo da árvore, demos então início à construção gráfica da mesma.

Como foi referido anteriormente, foi utilizada uma biblioteca *JavaScript* específica para este tipo de representações. Inicialmente, quando o utilizador entra na página de famílias, é efetuado um pedido ao servidor para o cálculo da árvore completa, sobre o individuo escolhido. Quando o utilizador carrega no botão de desenho da árvore, esta é truncada segundo o nível limite (inferior e superior) inserido pelo utilizador. Após a criação da árvore, a biblioteca *JavaScript* permite centrar a mesma num nó específico, sendo que nosso caso será no indivíduo sobre o qual é construída a árvore (nível 0).

Descrita a forma como foi efetuada a implementação da solução, podemos observar nas figuras seguinte o funcionamento da mesma integrada no *Time Link*, aquando a importação de dados (cálculo de famílias) e visualização da árvore (cálculo da árvore genealógica), respetivamente.

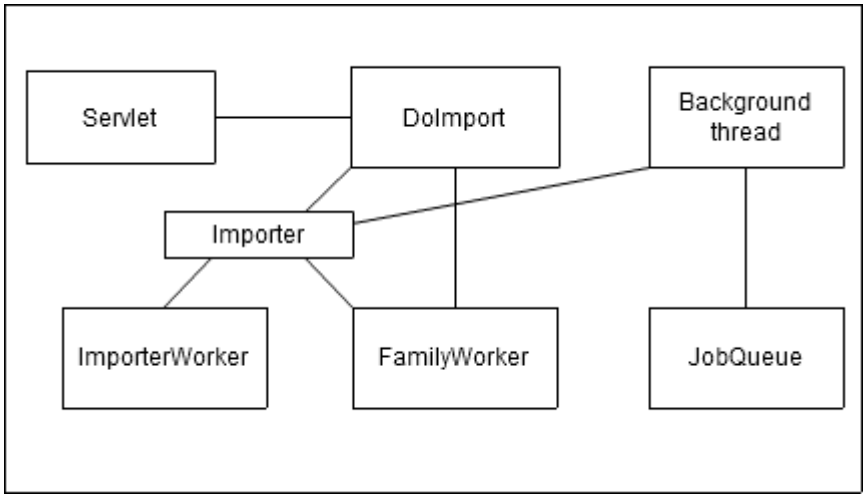

Figura 15 – Diagrama de funcionamento cálculo de famílias.

Efetuado o pedido de importação à *Servlet*, este é redirecionado para a classe DoImport que instancia os Worker's correspondentes e agendada a execução da importação e do cálculo de famílias, respetivamente. No caso de se verificar um pedido para construção da árvore genealógica, este é redirecionado para a classe FamilyManagement que efetua o cálculo da árvore e devolve o resultado, como exemplificado na figura seguinte.

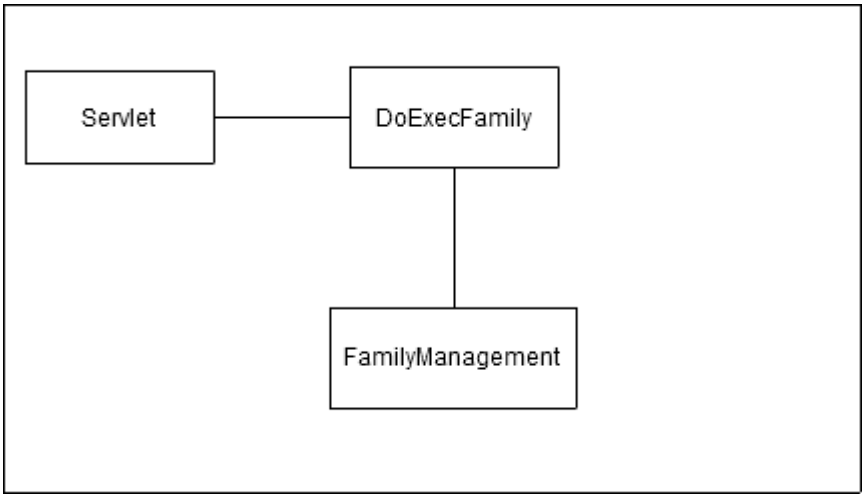

Figura 16 – Diagrama de funcionamento cálculo árvore genealógica.

Terminada esta implementação passamos então a descrever a funcionalidade que se seguiu, dentro do processo de desenvolvimento.

6.4. Implementação – Edição *inline*

Nesta fase de implementação foram elaboras as classes *Java* descritas na arquitetura respetiva a este desenvolvimento, tanto como as bibliotecas *JavaScript* e *CSS*.

Derivado dos requisitos foi implementado um processo ao qual será dado início com o *click* no nome do ficheiro (na página *Web*), sendo o utilizador então redirecionado para uma nova página onde poderá visualizar, editar e gravar o ficheiro previamente escolhido. Antes da apresentação da página *Web* de edição é instanciada no servidor a classe DoEditFile que devolve o conteúdo do ficheiro. Introduzindo o nome do novo ficheiro originado pela edição e clicando no botão "guardar", este ficheiro é armazenado no disco do servidor (no nosso caso servidor e cliente são a mesma máquina) através da instanciação da classe DoSaveFile.

Na figura seguinte podemos observar o diagrama de funcionamento da solução implementada para esta funcionalidade.

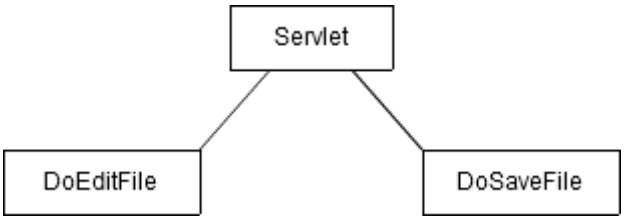

Figura 17 – Diagrama de funcionamento edição *inline*.

Após esta fase, demos então início à implementação das bibliotecas *JavaScript* e *CSS* para o *highlight* de sintaxe, sendo estas importadas na página *Web* onde é efetuada a edição do ficheiro.

Com foi referido, a biblioteca CodePress permite a implementação de novos módulos, ou seja, *highlight* de sintaxe de linguagens distintas das já existentes, sendo apenas necessário implementar um novo ficheiro *JavaScript* que contenha as expressões regulares e *keywords* que permitam identificar as palavras ou frases que devem ser coloridas, e um ficheiro *CSS* que contenha a definição das cores a atribuir a cada expressão regular ou *keyword* definida.

As *keywords* a identificar já se encontravam descriminadas (num ficheiro com o nome mhk.xml implementado anteriormente a este estágio, para o editor de texto jEdit e presente no repositório SVN do *googlecode* anteriormente referenciado), pelo que foram então elaboradas as expressões regulares que aceitassem e diferenciassem estas *keywords*, sendo depois definidas as cores a atribuir (também consistente com as cores do módulo jEdit).

6.5. Implementação – Introdução de dados no *Web Browser*

Nesta fase de implementação, foram elaboras as classes *Java* descritas na arquitetura respetiva a este desenvolvimento, que passamos agora a descrever.

Derivado dos requisitos foi implementado um processo ao qual será dado início com o *click* na tabulação de criação de fontes (no menu principal da página *Web*), sendo o utilizador então redirecionado para uma nova página onde poderá criar a sua fonte de informação. Com este *click* é instanciada no servidor a classe DoCreateSource que direciona o utilizador para a página *Web* de criação de fontes. Nesta página o utilizador tem a possibilidade de criar as suas fontes de informação e exporta-las para XML com um simples *click* que dá início a este processo (instanciando no servidor a classe DoSaveSource que por sua vez instancia a classe SaveWebSource), sendo a importação efetuada automaticamente a partir desta classe e após a criação do ficheiro.

É importante também referir que os dados inseridos pelo utilizador são enviados da camada de apresentação para a camada de controladores através de um objeto do tipo JSONObject. Este tipo de objetos permite o armazenamento de pares nome/valor, que no nosso caso contêm as listas de pessoas, atributos, relações, atos, funções e a informação sobre a fonte. [30]

Enviado este objeto para a camada de controladores através de um pedido AJAX (*Asynchronous Javascript and XML*) que permite contactar o servidor assincronamente e receber respostas do mesmo (por exemplo, e no nosso caso, importar os dados sem atualizar a página *Web*, podendo o utilizador continuar a editar a fonte e substituir os dados importados anteriormente). Este objeto é acedido no servidor através da classe *Java* JSON.simple, que permite retirar toda a informação contida no mesmo. [31]

A informação é depois utilizada para criar o ficheiro XML de uma forma consistente para o importador da aplicação. Na figura seguinte podemos observar o diagrama de funcionamento da solução implementada.

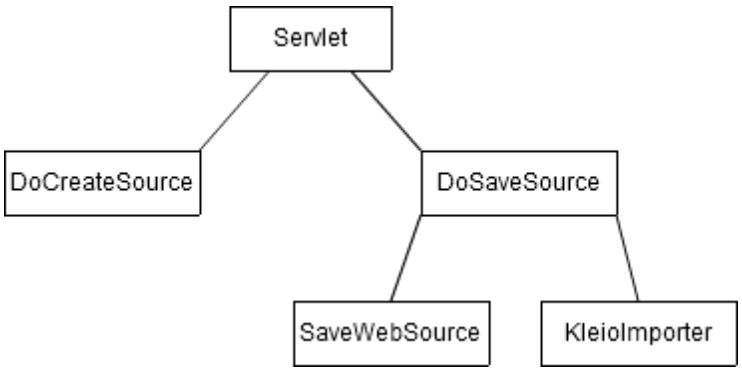

Figura 18 – Diagrama de funcionamento introdução de dados *Web Browser*.

Como podemos observar na figura anterior, a solução implementada segue a mesma estrutura das funcionalidades anteriores e das que já existiam no *Time Link*, sendo que desta vez o importador é automaticamente instanciado após a criação de uma fonte de informação.

7. Testes e validação

A fase de testes é uma das etapas mais importantes de um projeto de desenvolvimento de *software*, pois esta permite melhorar a qualidade do produto final. Para cada desenvolvimento efetuado durante o ano letivo será descrita esta fase referindo os métodos e tipos de testes desenvolvidos.

7.1. Testes – *Package* de instalação *Windows*

Para estes desenvolvimentos foram efetuados testes unitários e de integração, testes de sistema e testes de aceitação, que passamos a descrever nos próximos pontos.

7.1.1. Testes unitários e de integração

Durante o desenvolvimento do *package* de instalação, foram realizados testes unitários e de integração. Mais especificamente, antes de se introduzir uma nova secção no *package*, foi testado seu comportamento (instalação) individual, estando assim independente das secções anteriores. Estando o comportamento individual da secção dentro do esperado, passamos então à integração da mesma com as secções anteriores, testando novamente o comportamento do *package*. Este processo repetiu-se iterativamente sobre todas as secções implementadas.

7.1.2. Testes de sistema

Nesta fase de testes, foram efetuados testes de instalação e utilização do *Time Link* em sistemas operativos *Windows*, pois este é um dos requisitos definidos anteriormente. Assim sendo, a aplicação foi instalada usando o *package* de instalação em máquinas com sistemas operativos *Windows* distintos, testando depois a execução da mesma. Estes testes foram avaliados pelo programador e pelo orientador da empresa. Na tabela seguinte podemos verificar os sistemas operativos abrangidos e os resultados obtidos.

Sistema Operativo	Resultado
Windows XP *Service Pack* 3 (instalação limpa atualizada)	✓
Windows Vista *Service Pack* 2 (cerca de 1 ano de utilização)	✓
Windows 7 *Service Pack* 1 (cerca de 4 meses de utilização)	✓

Tabela 13 – Testes de instalação e utilização em sistemas operativos *Windows*.

Como podemos observar obtivemos resultados positivos no que toca à instalação e utilização da aplicação em sistemas operativos *Windows*. De notar que a tabela anterior demonstra o comportamento da versão final do *package* de instalação, pois à medida que se introduzia um sistema operativo *Windows* distinto, foram corrigidos vários erros ao nível de variáveis de ambiente, início e paragem de serviços e localizações pré-definidas de instalação, sendo estes problemas resolvidos de forma genérica.

De seguida passamos a uma fase em que supomos que o utilizador já possa ter ferramentas semelhantes (por exemplo bases de dados e servidor) instaladas no seu computador. Estes testes foram efetuados em várias máquinas, com sistemas operativos *Windows* distintos, adicionando e retirando instalações binárias das componentes e sendo avaliados pelo programador. Vamos então verificar na tabela seguinte o comportamento das ações de instalação dos pré-requisitos, tendo em conta que foram executados nos sistemas operativos descritos na tabela anterior.

Componente	Resultado
Apache Tomcat 7.0.35 já instalado	✓
MySQL 5.5.31 já instalado	✓
jEdit 5.0.0 já instalado	✓
Prolog 5.8.3 já instalado	✓
JRE 7 já instalado	✓

Tabela 14 – Testes de instalação dos pré-requisitos *do package Windows*.

Visto que a instalação destes componentes é efetuada independentemente do *software* já existente na máquina, a aplicação continua funcional mesmo que alguns destes componentes já se encontrem na mesma. Como estes pré-requisitos instalados pelo *package* necessitam de configuração específica para o funcionamento da aplicação, esta executa corretamente. É importante referir que a tabela anterior representa o comportamento da versão final do *package* de instalação.

Esta fase de testes permitiu resolver vários erros ao nível de registo de serviços, pois estes podem já existir na máquina (outras instalações de base de dados e servidor podem registar os seus serviços no gestor do sistema operativo *Windows*), o que nos fez optar por não registar os mesmos, executando-os através dos *scripts* já descritos anteriormente.

Por fim foi testada a desinstalação da aplicação em vários sistemas operativos *Windows*, utilizando o desinstalador implementado. Estes testes consistiram na instalação do *Time Link* (pré-requisitos e ficheiros de suporte) usando o *package* de instalação em máquinas distintas e posterior desinstalação das componentes, sendo depois avaliados pelo programador. Podemos observar os resultados obtidos na tabela seguinte.

Sistema Operativo	Resultado
Windows XP *Service Pack* 3	✓
Windows Vista *Service Pack* 2	✓
Windows 7 *Service Pack* 1	✓

Tabela 15 – Testes de desinstalação do *package Windows*.

No caso de o utilizador executar o desinstalador da aplicação e algum dos recursos a eliminar estar em utilização, este vai permanecer na máquina e tem que ser eliminado manualmente. Após o recuso ser libertado pode também ser efetuada desinstalação ou uma nova instalação, pois os ficheiros já existentes são apagados ou substituídos, respetivamente.

7.1.3. Testes de aceitação

Como referido anteriormente o público-alvo desta aplicação são Historiadores e utilizadores comuns, sendo que ambos não devem necessitar de conhecimentos de informática para instalar e executar a aplicação, o que é um dos objetivos do *package* de instalação.

Assim sendo decidimos agendar uma sessão de testes com os Historiadores João Pinho e Sónia Nobre que têm vindo a trabalhar com aplicação por forma a ter uma opinião sobre o que pode ser melhorado ao nível de interação com o utilizador. Esta sessão foi efetuada no dia 10/12/2012 e contou também com a presença do orientador da empresa, Alexandre Pinto.

Nesta sessão foi fornecido o *package* de instalação implementado e pedido ao Historiador que instalasse a aplicação sem qualquer tipo de informação prévia ou manual de instruções. À medida que a instalação prosseguia, foram apontadas as dificuldades com que se deparou, assim como a sua opinião em alguns pontos que deveriam ser melhorados para que o utilizador tivesse uma experiência mais agradável aquando a instalação e utilização do *Time Link*, pontos estes que passamos a enumerar.

- Instalações fora da janela do instalador – Algumas instalações (jEdit e *Apache Tomcat*) estavam a ser efetuadas através do instalador do *Windows*, o que aumentava a complexidade de instalação. Resolvido copiando para a máquina em questão versões de *software* que não utilizam instalador (*Apache Tomcat*, MySQL e jEdit). De notar que no caso da ferramenta MySQL houve necessidade de importar a estrutura de base de dados já criada em outras instalações em máquina do *Time Link* (através da exportação de um ficheiro *backup* de extensão .sql de uma base de dados configurada e vazia, fornecida pela empresa, conseguimos facilmente importar a estrutura para a nova instalação).
- Utilizador necessitava de reiniciar o servidor após instalação – O *deployment* do ficheiro mhk.war (um dos pré-requisitos da aplicação) era efetuado após a instalação do servidor. Resolvido utilizando um servidor já com o ficheiro *deployed*.
- Utilizador necessitava de mensagens informativas – Alguns passos, nomeadamente o *shutdown* da aplicação e o fim de instalação não eram transparentes e poderiam levar a equívocos por parte do utilizador. Resolvido através da apresentação destas mensagens, no término dos serviços e da instalação da aplicação.

Depois de corrigidos estes problemas voltou-se a testar o *package* de instalação no dia 18/12/2012 com o orientador da empresa no papel de utilizador comum, sendo então aprovado.

7.1.4. Exemplo de utilização

O resultado final do *package* de instalação é um ficheiro executável (extensão .exe) que é executado com um simples duplo-clique. De seguida é iniciado todo o processo de instalação numa janela, que ilustramos na figura seguinte.

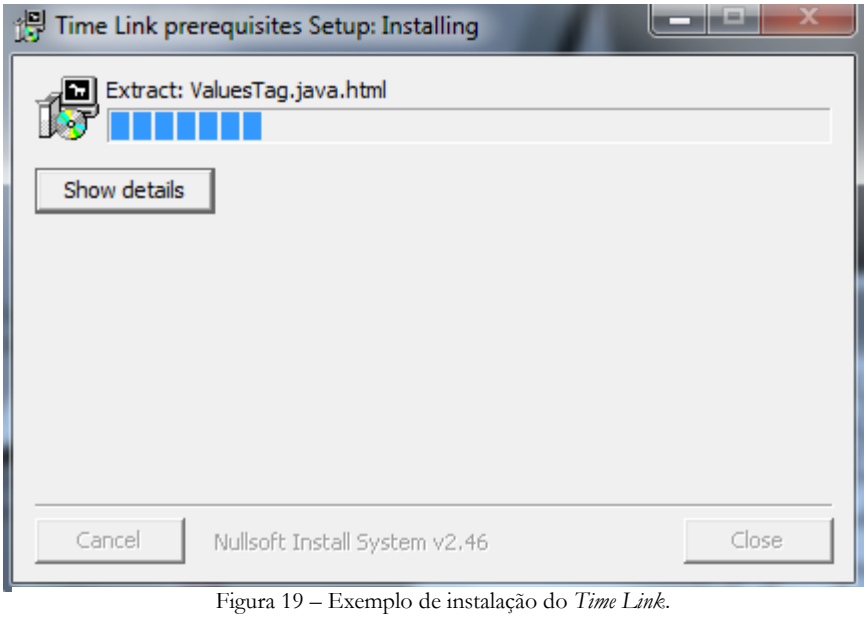

Figura 19 – Exemplo de instalação do *Time Link*.

A execução do instalador é sequencial e não exige qualquer configuração por parte do utilizador. Após a instalação são criados atalhos no ambiente de trabalho da máquina, que permitem iniciar e parar o *Time Link*, como é ilustrado na figura seguinte.

Figura 20 – Atalhos de início e paragem do *Time Link*.

Nesta fase a aplicação já está funcional e com um duplo-clique no ícone *Time Link Startup* podemos iniciar a aplicação, sendo o resultado final ilustrado na figura seguinte.

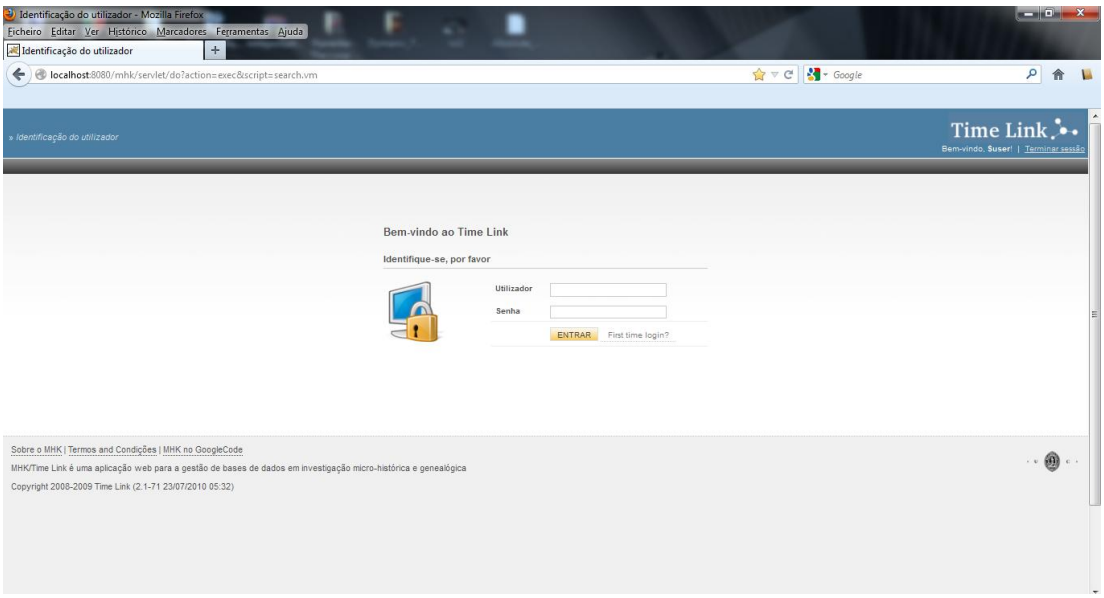

Figura 21 – Exemplo do *Time Link*.

A desinstalação pode ser efetuada clicando no atalho na barra iniciar, como é ilustrado na figura seguinte, tanto como o início e paragem de serviços anteriormente referido.

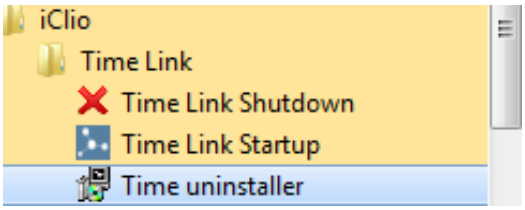

Figura 22 – Exemplo de desinstalação do *Time Link*.

Pelo exemplo de utilização podemos verificar que o *package* de instalação é consistente com as aplicações desenvolvidas para o sistema operativo *Windows* e consequentemente de fácil aprendizagem para o utilizador.

7.1.5. Problemas identificados

Durante os testes efetuados na fase de instalação, desinstalação e execução de serviços, identificou-se os seguintes problemas:

- Podem ocorrer exceções ao iniciar os serviços de base de dados e servidor, sem que o utilizador seja notificado, pois estas não são tratadas;
- Na execução do desinstalador, caso algum dos recursos a serem removidos da máquina estarem bloqueados (em utilização), estes não são removidos;
- Na execução do instalador, caso já existam os ficheiros a serem escritos na máquina, e alguns destes se encontrem em utilização, não é efetuada a escrita dos mesmos;

- Na execução dos *scripts*, estes podem ser bloqueados pelo antivírus instalado na máquina, pelo que devem ser garantidas as permissões necessárias neste caso.

Vistos estes problemas vamos agora fazer uma breve reflexão sobre trabalhos possíveis para melhorar esta componente.

7.1.6. Trabalhos possíveis

Apesar deste *package* de instalação do *Time Link* já resolver vários problemas de instalação e utilização existe ainda muitas funcionalidades que podem ser desenvolvidas em termos de suporte à aplicação. Assim sendo identificámos os pontos potencialmente mais importantes neste momento e sua justificação.

- Atualização automática do ficheiro mhk.war – Este ficheiro pode ser facilmente atualizado aquando a instalação do *Time Link*, podendo assim instalar sempre a ultima versão estável da aplicação, apenas necessitando de suporte do lado do servidor. Pode também ser efetuada a verificação de atualização na ação de iniciar o *Time Link*;
- Atualização de componentes – Esta atualização tem de ser efetuada manualmente, obrigando o programador a instalar o *software* pretendido e efetuar as alterações. Se estes componentes pudessem ser atualizados de uma forma mais agilizada seria interessante.

7.2. Testes – *Package* de instalação *MAC*

Seguindo a mesma ideia do *package* de instalação *Windows*, esta fase foi dividida em testes unitários e de integração, testes de sistema e testes de aceitação.

7.2.1. Testes unitários e de integração

Os testes unitários e de integração foram realizados durante o desenvolvimento da versão *MAC* do *package* seguem a mesma estrutura da fase de testes anteriormente referida introduzindo uma nova secção, testando o comportamento individual e integrado com as secções anteriores Este processo também se repetiu iterativamente sobre todas as secções implementadas.

7.2.2. Testes de sistema

Seguindo a mesma lógica, foram também efetuados testes de instalação e utilização do *Time Link* em sistemas operativos *MAC*. A aplicação foi instalada usando o *package* em máquinas com sistemas operativos *MAC* distintos, testando depois a sua execução. Estes testes foram avaliados pelo programador e pelo orientador da empresa. Na tabela seguinte podemos verificar os sistemas operativos abrangidos e os resultados obtidos.

Sistema Operativo	Resultado
MAC OS 10.8.3 (1 ano de utilização)	✓
MAC OS 10.8.2 (4 meses de utilização)	✓
MAC OS 10.7.3 (instalação limpa)	✓

Tabela 16 – Testes de instalação e utilização em sistemas operativos *MAC*.

Como podemos observar, nesta versão também obtivemos resultados positivos no que toca à instalação e utilização da aplicação em sistemas operativos *MAC*, sendo que a tabela anterior representa o comportamento final do *package* de instalação.

Seguindo a estrutura do testes ao *package* anterior vamos também supor que o utilizador já possa ter ferramentas semelhantes instaladas na máquina. Estes testes foram novamente efetuados em várias máquinas com sistemas operativos *MAC* distintos, adicionando e retirando instalações das componentes e sendo avaliados pelo programador. Na tabela seguinte podemos observar o comportamento das ações de instalação.

Componente	Resultado
Apache Tomcat 7.0.34 já instalado	✓
MySQL 5.5.29 já instalado	✓
jEdit 5.0.0 já instalado	✓
Prolog 5.8.3 já instalado	✓

Tabela 17 – Testes de instalação dos pré-requisitos do *package MAC*.

Tal como no *package* de instalação *Windows,* a instalação destes componentes é efetuada independentemente do *software* já instalado na máquina, a aplicação continua funcional mesmo que alguns destes componentes já existam na mesma. É importante também referir que a tabela anterior representa o comportamento da versão final do *package* de instalação e que esta fase de testes permitiu resolver vários comportamentos ao nível de permissões da base de dados e execução de serviços.

Por fim foi testada a desinstalação da aplicação nos vários sistemas operativos *MAC*, utilizando o *script* de desinstalação implementado. Estes testes consistiram na instalação do *Time Link* (pré-requisitos e ficheiros de suporte) usando o *package* de instalação em máquinas distintas e posterior a desinstalação das componentes, sendo depois avaliados pelo programador. Podemos observar os resultados obtidos na tabela seguinte.

Sistema Operativo	Resultado
MAC OS 10.8.3	✓
MAC OS 10.8.2	✓
MAC OS 10.7.3	✓

Tabela 18 – Testes de desinstalação do *package MAC*.

Tal como referido no *package* de instalação *Windows*, no caso de o utilizador executar o desinstalador da aplicação e algum dos recursos a eliminar estar em utilização, este vai permanecer na máquina e tem que ser eliminado manualmente. Após o recuso ser libertado pode também ser efetuada a desinstalação ou uma nova instalação, pois os ficheiros já existentes são apagados ou substituídos, respetivamente.

7.2.3. Testes de aceitação

Tal como foi efetuado para o *package* de instalação *Windows,* nesta versão *MAC*, decidimos também agendar uma sessão de testes com os Historiadores João Pinho e Sónia Nobre, que têm vindo a trabalhar com aplicação. Esta sessão foi efetuada no dia 06/02/2013 e contou também com a presença do orientador da empresa, Alexandre Pinto e do gestor de *software* da mesma, João Carvalho.

Nesta sessão foi fornecido o *package* de instalação implementado e pedido aos Historiadores que instalassem a aplicação sem qualquer tipo de informação prévia. À medida que a instalação prosseguia, foram apontadas as dificuldades com que se depararam, assim como as suas opiniões em alguns pontos a serem melhorados, que passamos a enumerar.

- Definições do jEdit – Visto que esta componente já vem configurada pelo estagiário (contém *templates* etc.), existiam algumas inconsistências no que toca a diretorias *default* da mesma.
- Bibliotecas Prolog em falta – Durante a execução da aplicação, verificou-se a inexistência de algumas bibliotecas utilizadas pela componente Prolog, necessárias para o bom funcionamento do tradutor *Kleio*, não incluídas no *package*.
- Problemas com permissões – Inicialmente o *package* de instalação não tinha em conta as permissões do utilizador para a execução de *scripts*, sendo que estre problema foi ultrapassado perguntando ao utilizador as suas credenciais no ato de execução da aplicação.

Depois de corrigidos estes problemas voltou-se a testar o *package* de instalação no dia 20/02/2013 com o orientador da empresa no papel de utilizador comum, sendo então aprovado.

7.2.4. Exemplo de utilização

O resultado final do *package* de instalação é um ficheiro único de extensão .pck que pode ser executado com um simples duplo-clique. De seguida é iniciado todo o processo de instalação numa janela, que ilustramos na figura seguinte.

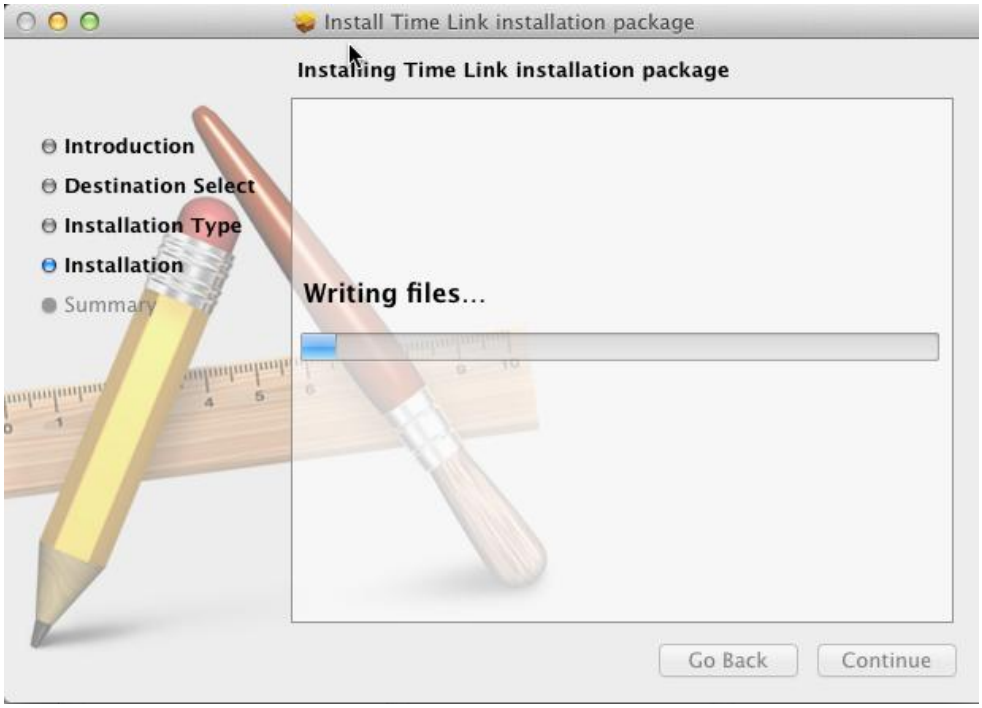

Figura 23 – Exemplo de instalação *package* MAC.

A execução do instalador é sequencial e não exige qualquer configuração por parte do utilizador. Após a instalação são criados atalhos na diretoria /Applications/Time Link da máquina, que permitem iniciar e parar o *Time Link*, tal como iniciar o editor de texto jEdit e desinstalar a aplicação. Na figura seguinte podemos observar estes atalhos.

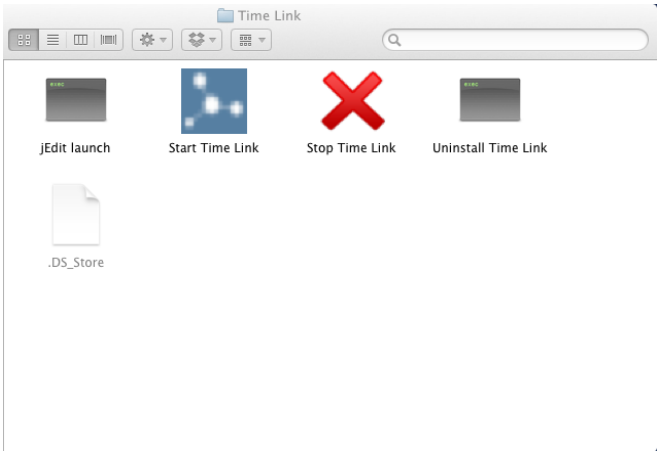

Figura 24 – Atalhos *package MAC*.

Nesta fase a aplicação já está funcional e com um duplo-clique no ícone *Start Time Link* podemos iniciar a aplicação, sendo o resultado final ilustrado na figura seguinte.

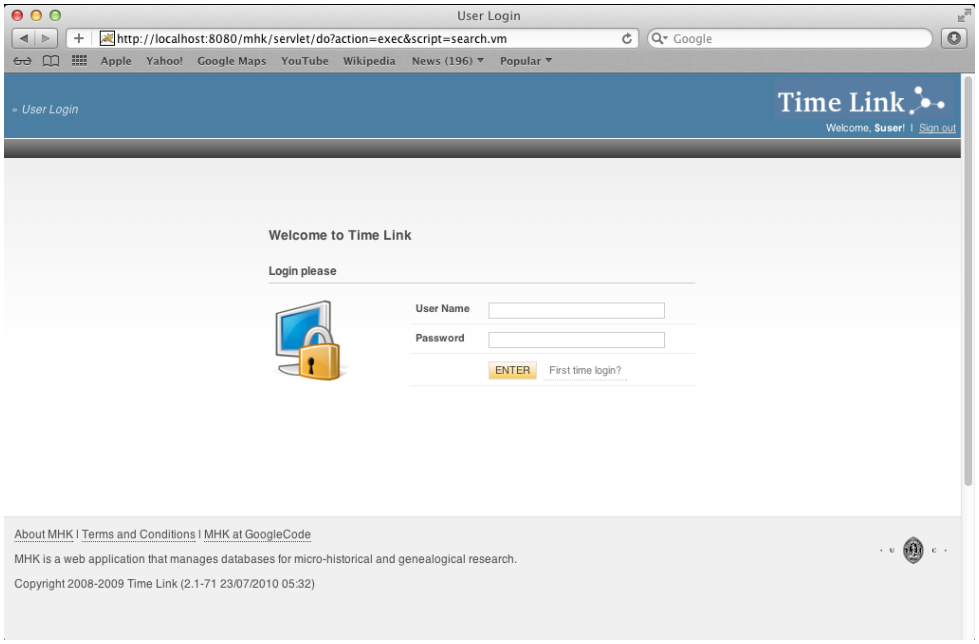

Figura 25 – Exemplo do *Time Link*.

7.2.5. Problemas identificados

Tal como no *package* de instalação *Windows,* durante os testes efetuados na fase de instalação, desinstalação e execução de serviços, identificou-se os seguintes problemas:

- Podem ocorrer exceções ao iniciar os serviços de base de dados e servidor, sem que o utilizador seja notificado, pois estas não são tratadas;
- Na execução do desinstalador, caso algum dos recursos a serem removidos da máquina estarem bloqueados (em utilização), estes não são removidos;
- Na execução do instalador, caso já existam os ficheiros a serem escritos na máquina, e alguns destes se encontrem em utilização, não é efetuada a escrita dos mesmos.

Vistos estes problemas vamos agora fazer uma breve reflexão sobre trabalhos possíveis para melhorar esta componente.

7.2.6. Trabalhos possíveis

Seguindo a mesma estrutura do *package* de instalação *Windows* identificamos os seguintes pontos como mais importantes para a aplicação neste momento.

- Atualização automática do ficheiro mhk.war – Este ficheiro pode ser facilmente atualizado aquando a instalação do *Time Link*, podendo assim instalar sempre a ultima versão estável da aplicação. Apenas necessita de suporte do lado do servidor. Pode também ser efetuada a verificação de atualização na ação de iniciar o *Time Link*;
- Atualização de componentes – Esta atualização tem de ser efetuada manualmente, obrigando o utilizador a instalar o programa pretendido e efetuar as alterações. Se estes componentes fossem atualizados de uma forma mais agilizada seria interessante.

7.3. Testes – Importação GEDCOM

Nesta fase, para além dos testes efetuados pelo programador durante a fase de desenvolvimento do *software*, foi utilizada a *Framework open source* JUnit para a realização de testes unitários e de integração. [32]

Esta *Framework* permite a programação casos de teste (ficheiros com código *Java*) e a sua execução através de um interface gráfico (no nosso caso o ambiente de desenvolvimento Eclipse IDE), automatizando assim o processo *testing*. [33]

Nos próximos pontos descrevemos então os testes efetuados, o seu resultado e tipo em que se inserem.

7.3.1. Testes unitários e de integração

Utilizando a *Framework* JUnit, foram elaborados algumas classes de teste e executadas sobre ficheiros GEDCOM distintos. Esta variedade de ficheiros GEDCOM utilizada tem como objetivo garantir que o *parser* implementado cobre realmente todos os casos contidos no tipo de informação a tratar (requisitos da tradução GEDCOM).

Na tabela seguinte podemos observar os testes unitários e de integração efetuados, bem como os ficheiros GEDCOM utilizados (segundo codificação ANSEL), sendo alguns destes ficheiros bastante utilizados em aplicações que importam e exportam este formato (ficheiros com o prefixo TGC e TGP, no nosso caso), pois cobrem todas as possibilidades do formato GEDCOM (mais especificamente cobrem qualquer *tag* possível em qualquer localização possível). [34][35]

Teste	Ficheiro GEDCOM	Resultado
Upload do ficheiro GEDCOM	TGC551.ged	✓
Tradução do ficheiro GEDCOM	TGC551.ged	✓
Importação do ficheiro XML gerado	TGC551.ged	✓
Integração dos 3 testes anteriores	TGC551.ged	✓
Upload do ficheiro GEDCOM	TGP-Gedccom551Test-v1.03.ged	✓
Tradução do ficheiro GEDCOM	TGP-Gedccom551Test-v1.03.ged	✓
Importação do ficheiro XML gerado	TGP-Gedccom551Test-v1.03.ged	✓
Integração dos 3 testes anteriores	TGP-Gedccom551Test-v1.03.ged	✓

Tabela 19 – Testes unitários e de integração GEDCOM.

Esta fase de testes permitiu resolver vários problemas ao nível da tradução GEDCOM para XML, pois os ficheiros de teste utilizados permitiram verificar casos específicos de tradução que não estavam a ser tratados. Durante os testes unitários efetuados tentamos garantir que a resolução de cada problema é efetuada de forma correta, integrando depois todas as componentes e verificando o seu comportamento geral.

Para além destes testes foram também testadas algumas codificações distintas para perceber as limitações do *parser* a este nível.

Codificação	Resultado
ANSEL	✓
ASCII	✓
Unicode	x
UTF-8	x

Tabela 20 – Testes codificações de ficheiros GEDCOM.

Como podemos observar, só algumas codificações de ficheiros são aceites pelo *parser*, mas estas incluem a versão considerada *standard* (ANSEL) para ficheiros GEDCOM.

7.3.2. Testes de sistema

Os testes de sistema efetuados tiveram em conta a integração do *parser* no *Time Link*, pois após a tradução implementada é também necessário importar o ficheiro XML gerado para o sistema e verificar se a informação está corretamente relacionada. Para além de vários ficheiros, foi também utilizada a aplicação local *MyFamilyTree* que permite a importação, edição e exportação de ficheiros

GEDCOM segundo um interface gráfico. Na tabela seguinte podemos então observar os ficheiros utilizados e os testes efetuados.[36]

Teste	Ficheiro GEDCOM	Resultado
Comparação com a informação *MyFamilyTree*	TGC551.ged	✓
Comparação com a informação *MyFamilyTree*	TGP-Gedccom551Test-v1.03.ged	✓

Tabela 21 – Teste de sistema GEDCOM.

Esta fase de testes permitiu resolver vários problemas de atributos em falta, relações em duplicado etc., comparando o resultado dos dados importados no *Time Link* com os importados na aplicação *MyFamilyTree*.

7.3.3. Testes de aceitação

Tal como foi efetuado com os *packages* de instalação, decidimos também efetuar uma sessão de testes com o gestor de *software* da empresa, João Carvalho. Esta sessão foi efetuada no dia 22/03/2013. Resultante desta sessão foi a necessidade de corrigir alguns aspetos da tradução que passamos a descrever.

- Criação de ocorrências de pessoas – É especifico do *Time Link* a criação destas ocorrências sempre que existe informação de eventos relacionados com indivíduos (batismo, nascimento, crisma etc.). A geração destas ocorrências não estava a ser efetuada, tendo então de se adicionar esta função.
- Relações do tipo "Same As" – Dentro do modelo de dados do *Time Link* existe a necessidade de definir indivíduos como "pessoas reais", que resultam do *linking* de informação. Para este efeito é necessário a introdução de uma relação deste tipo sempre que sabemos que o individuo que estamos a referenciar corresponde a uma "pessoa real". A geração destas relações não estava a ser efetuada, tendo então de se adicionar esta função.

Depois de corrigidos estes problemas e de efetuados novos testes unitários, de integração e de sistema, o *software* foi então aprovado. É importante referir também que o *software MyFamilyTree* foi usado como modelo de comparação durante esta fase.

7.3.4. Exemplo de utilização

Após a descrição das fases do processo de desenvolvimento de *software*, vamos apresentar um exemplo de utilização da funcionalidade implementada integrada no *Time Link*. Na figura seguinte podemos observar a página *Web* de tradução e importação de ficheiros. É importante referir que anteriormente apenas existia uma tabela de ficheiros (relativa aos ficheiros *Kleio* de extensão .cli), sendo que agora temos uma adicional que lista os ficheiros do tipo GEDCOM (de extensão .ged).

Figura 26 – Página importação no processo de tradução.

Clicando no botão de tradução ou na referência na linha do ficheiro correspondente, damos início ao processo de tradução. O relatório deste processo é escrito em ficheiro e apresentado na página representada na figura seguinte.

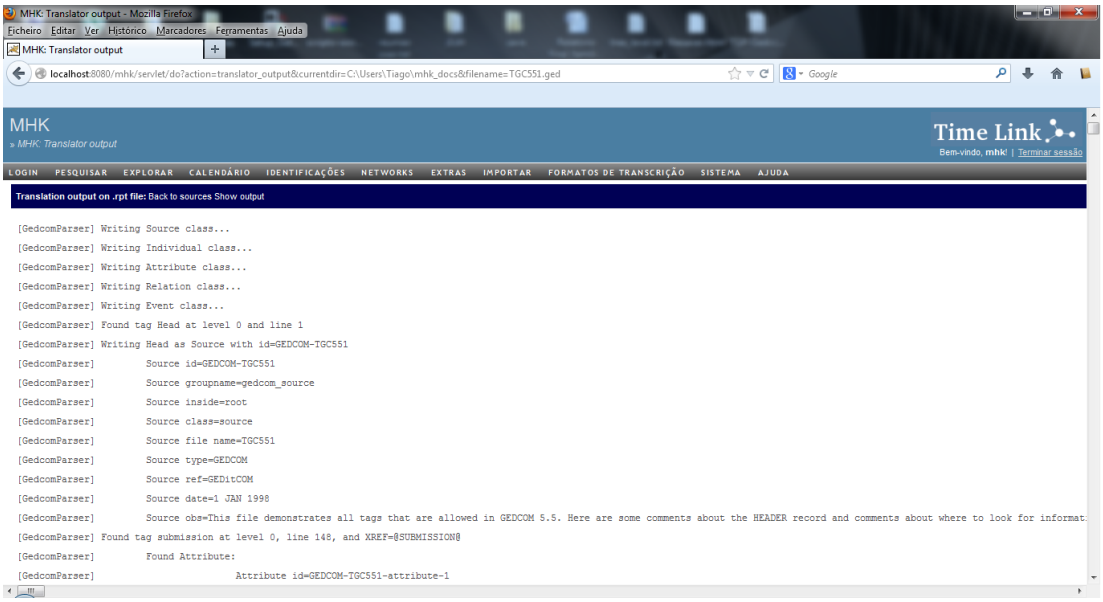

Figura 27 – Página de relatório de tradução.

Após esta tradução, voltando à página de importação, podemos verificar que o ficheiro foi traduzido com sucesso e que processo de importação está agora disponível, como representado na figura seguinte.

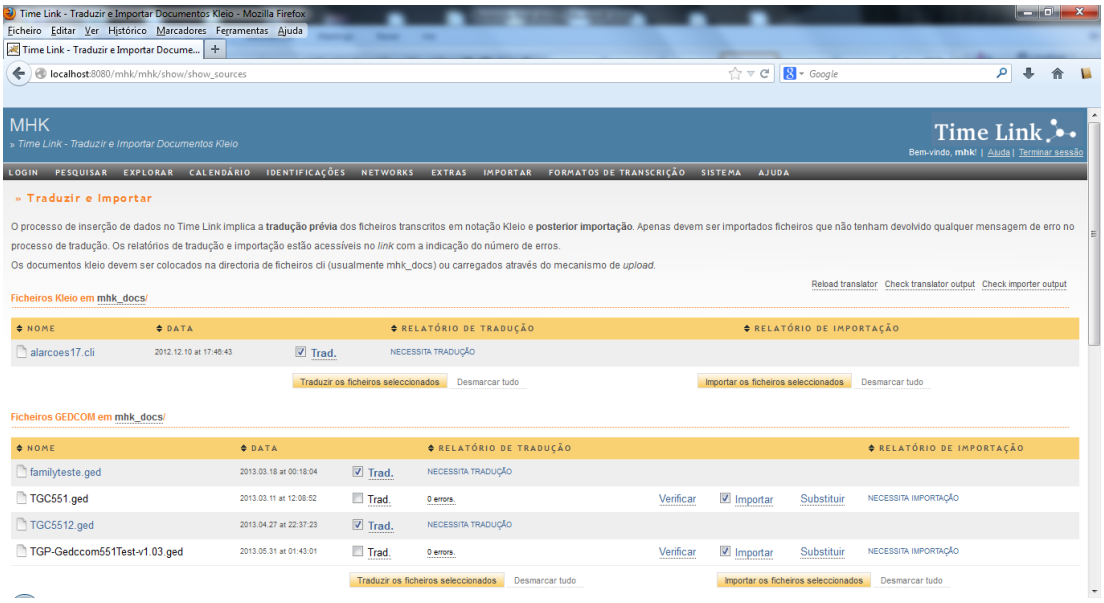

Figura 28 – Página de importação no processo de importação.

Seguindo a mesma lógica para o início do processo de importação (sendo que este já estava implementado *no Time Link*), é apresentada a página de relatório de erros, como podemos verificar na figura seguinte.

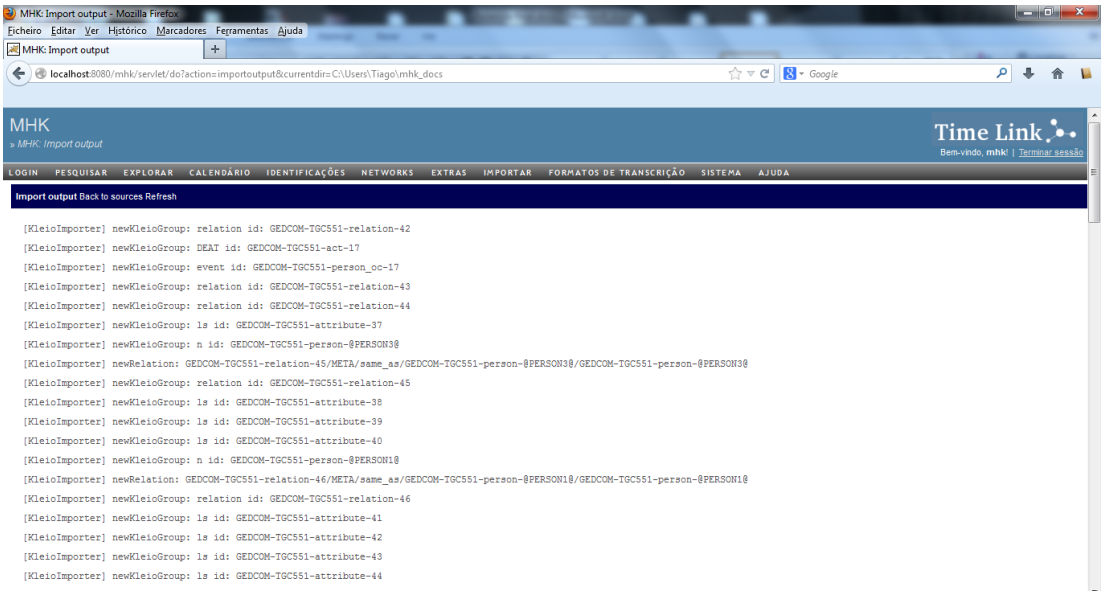

Figura 29 – Página de relatório de importação.

Terminada a importação e voltando à página de importação, podemos verificar que esta foi efetuada com sucesso e o número de erros identificados no processo, como apresentado na figura seguinte.

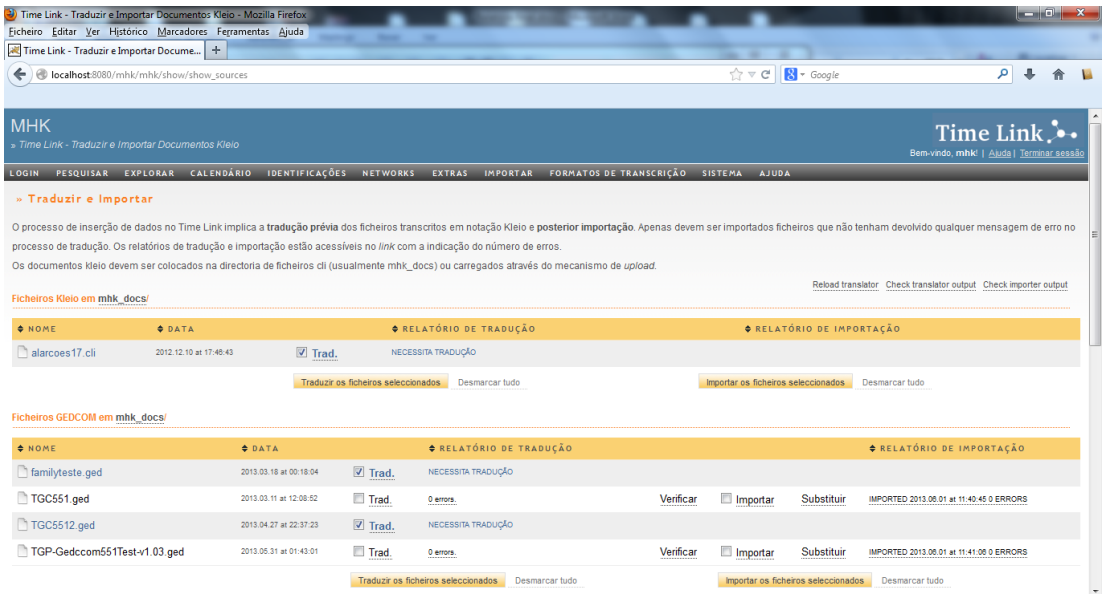

Figura 30 – Página após processo de tradução e importação.

Clicando no tipo de atributos, funções, relações ou procurando o individuo podemos então visualizar a informação inserida na base de dados, como exemplificado na figura seguinte.

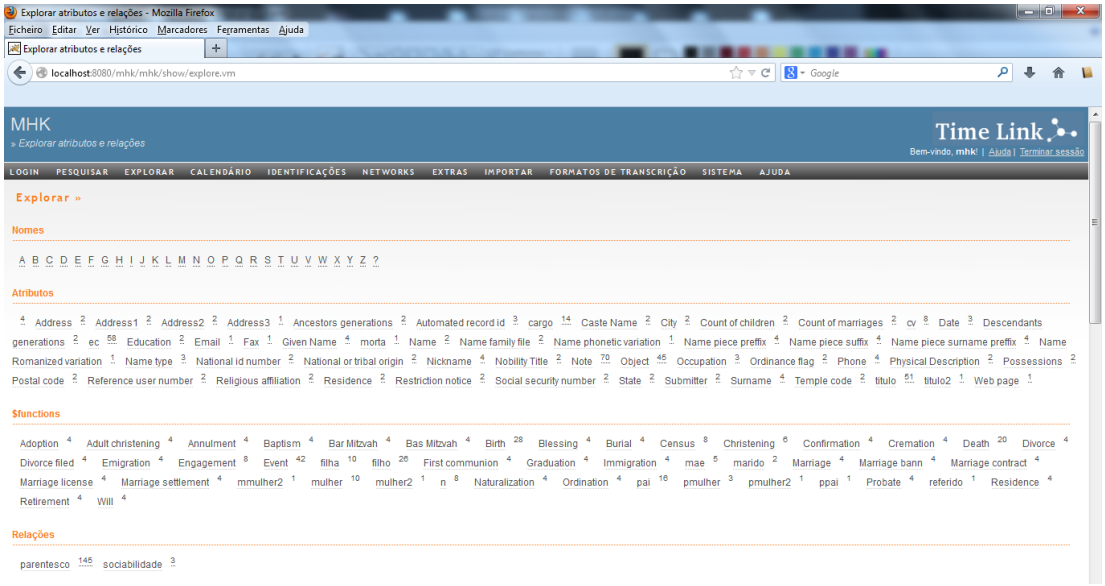

Figura 31 – Página de exploração de informação.

É importante referir que nesta página é apresentada informação proveniente de ficheiros *Kleio* e GEDCOM importada na base de dados.

7.3.5. Problemas identificados

Durante as fases de desenvolvimento de *software* e testes foram verificados os seguintes problemas:

- Só alguns tipos de codificação, nomeadamente ANSEL e ASCII são aceites pelo *parser*;

Visto este problema vamos agora fazer uma breve reflexão sobre trabalhos possíveis para melhorar esta componente.

7.3.6. Trabalhos possíveis

Nesta fase vamos identificar alguns pontos que podem ser melhorados, tanto em termos de experiência de utilização como de funcionalidade do *software*.

- É possível traduzir as famílias (*tag* FAM) sem ser efetuada a decomposição por relações de parentesco. Seria necessário alterar o importador;
- Permitir outros tipos de codificações, por exemplo UTF-8 e Unicode;

- Existem mais algumas *tags* que não estão a ser tratadas pelo *parser* devido à falta de suporte do modelo de dados e do importador. Sendo estes alterados poderemos importar toda a informação presento no ficheiro GEDCOM.

7.4. Testes – Famílias

Seguindo a mesma estrutura, foi novamente utilizada a *Framework* JUnit para a realização de testes unitários e de integração.

7.4.1. Testes unitários e de integração

Seguindo a mesma lógica, nesta fase foram implementadas classes de teste e executadas sobre ficheiros GEDCOM distintos. Esta variedade de ficheiros GEDCOM utilizada tem como objetivo garantir que a criação das famílias e da árvore genealógica cobre o maior número possível de casos.

Na tabela seguinte podemos observar os testes unitários e de integração efetuados, bem como o ficheiros GEDCOM utilizados.

Teste	Ficheiro GEDCOM	Resultado
Cálculo das famílias	TGC551.ged	✓
Escrita das famílias na base de dados	TGC551.ged	✓
Leitura das famílias da base de dados	TGC551.ged	✓
Cálculo da árvore	TGC551.ged	✓
Apresentação da árvore	TGC551.ged	✓
Integração dos testes 5 testes anteriores	TGC551.ged	✓
Cálculo das famílias	TGP-Gedccom551Test-v1.03.ged	✓
Escrita das famílias na base de dados	TGP-Gedccom551Test-v1.03.ged	✓
Leitura das famílias da base de dados	TGP-Gedccom551Test-v1.03.ged	✓
Cálculo da árvore	TGP-Gedccom551Test-v1.03.ged	✓
Apresentação da árvore	TGP-Gedccom551Test-v1.03.ged	✓
Integração dos testes 5 testes anteriores	TGP-Gedccom551Test-v1.03.ged	✓

Tabela 22 – Testes unitários e de integração famílias.

Da mesma forma que a funcionalidade anteriormente referida, é testada a solução do problema e a sua integração no processo de famílias, verificando assim o seu comportamento geral.

7.4.2. Testes de sistema

Os testes de sistema efetuados tiveram em conta a integração desta funcionalidade no *Time Link* tanto em termos do processo de importação e criação de famílias como do cálculo da árvore genealógica. Como anteriormente, foi utilizado o *software MyFamilyTree* como modelo de comparação

com a árvore construída no *Time Link*. Na tabela seguinte podemos observar os testes efetuados e os resultados obtidos.

Teste	Ficheiro GEDCOM	Resultado
Comparação com a árvore *MyFamilyTree*	TGC551.ged	✓
Comparação com a árvore *MyFamilyTree*	TGP-Gedccom551Test-v1.03.ged	✓

Tabela 23 – Testes de sistema famílias.

Esta fase de testes permitiu identificar casos específicos da árvore em que a sua construção não estava a ser efetuada de forma correta (por exemplo o caso de um homem ter filhos com várias mulheres e essas mulheres terem filhos com outros homens).

Para além deste processo, e derivado da utilização da linguagem *JavaScript* para desenho da árvore, esta representação foi testada a em vários *Web browsers* de forma a ter a certeza que resolvemos o problema de uma forma genérica, como podemos observar na tabela seguinte.

Browser	**Resultado**
Safari 5.1.7	✓
Google Chrome 27.0.1453.94	✓
Mozilla Firefox 21.0	✓
Internet Explorer 10.0.5	✓

Tabela 24 – Testes *Web browsers* famílias.

Como podemos observar na tabela anterior, a construção da árvore genealógica na camada de visualização é efetuada de uma forma genérica para os principais *Web Browsers* utilizados hoje em dia.

7.4.3. Testes de aceitação

Tal como foi efetuado nos desenvolvimentos anteriores, decidimos também efetuar uma sessão de testes com o gestor de *software* da empresa, João Carvalho. Esta sessão foi efetuada no dia 7/05/2013. Resultante desta sessão foi a necessidade de corrigir alguns aspetos da tradução que passamos a descrever.

- Limitações da biblioteca *JavaScript* – Como já foi referido, a biblioteca utilizada revela algumas limitações, pelo que a introdução de mais informação a cada nó da mesma é necessária.

Depois de corrigidos este problema e de efetuados novos testes unitários, de integração e de sistema, o *software* foi então aprovado.

7.4.4. Exemplo de utilização

Na figura seguinte podemos observar a página *Web* que apresenta a informação relativa a um individuo (nome, sexo, atributos, funções e relações).

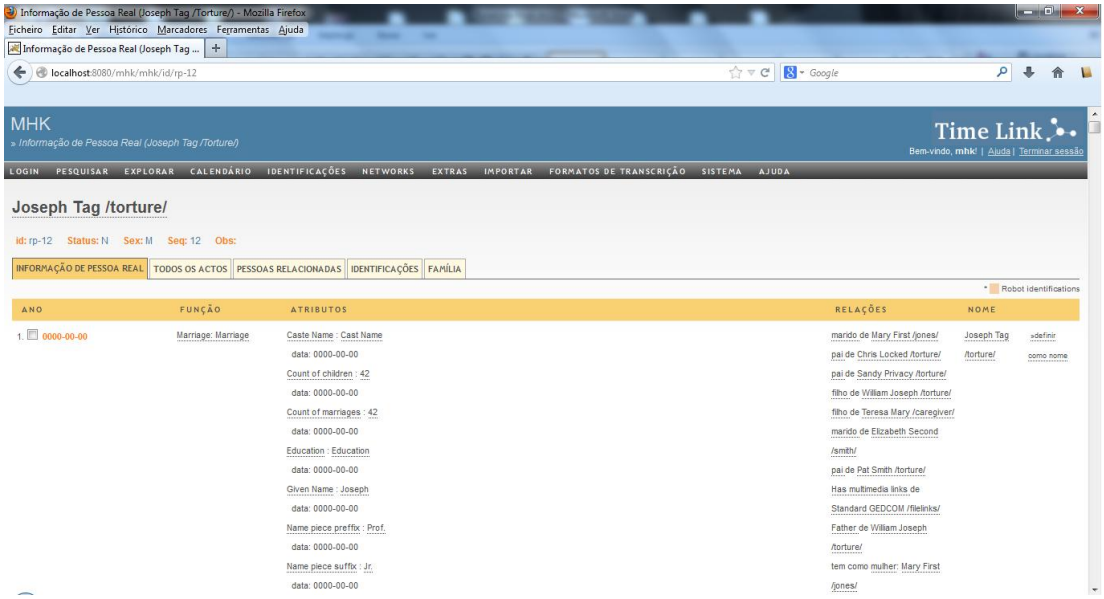

Figura 32 – Página de informação de um individuo.

Clicando na *tab* de famílias podemos então visualizar a árvore, sendo que podemos limitar o nível desta (superior e inferior). Na árvore apresentada na figura seguinte foi inserido o nível 0 (que significa a árvore completa).

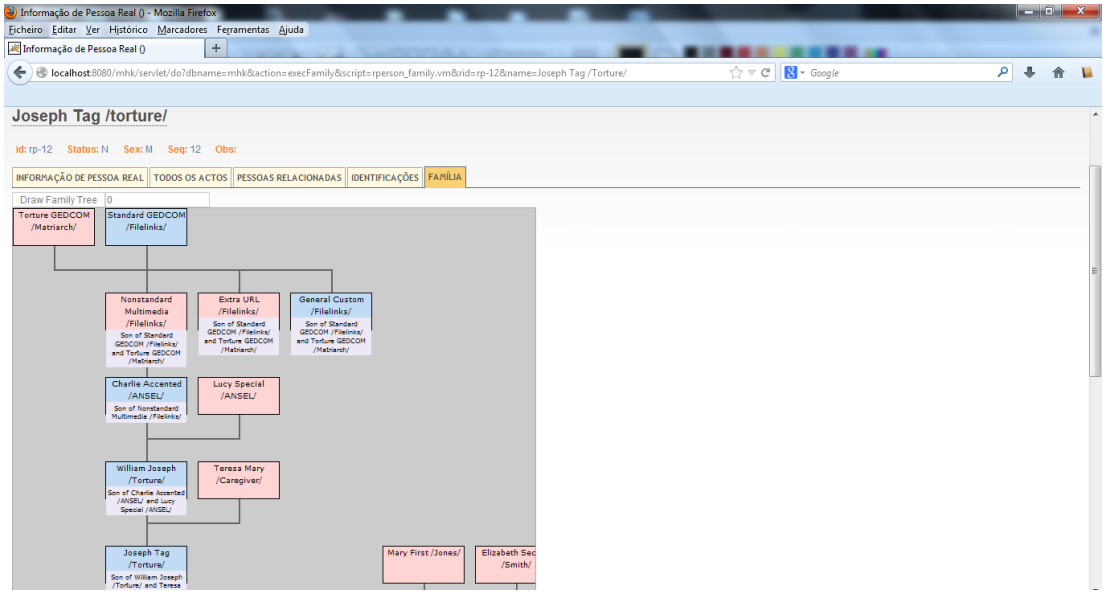

Figura 33 – Página árvore genealógica.

É importante referir que clicando na área da árvore conseguimos arrastar a mesma, visualizando assim o conteúdo que pode não ser apresentado devido ao limite da área de visualização.

7.4.5. Problemas identificados

Durante as fases de desenvolvimento de *software* e testes foram verificados os seguintes problemas:

- Relativamente à informação *Kleio*, a árvore é apresentada de forma correta mas devido à liberdade que esta linguagem de notação oferece (por exemplo, no ato de transcrição de um documento, caso exista informação que indique pais diferentes para o mesmo individuo, podem ser inseridos os dois pais), pode levar a árvores diferentes dependo do individuo sobre o qual esta é construída;
- Limitações de representação da biblioteca *JavaScript*.

Vistos estes problemas vamos agora fazer uma breve reflexão sobre trabalhos possíveis para melhorar esta componente.

7.4.6. Trabalhos possíveis

Como foi referido anteriormente, nesta fase vamos identificar alguns pontos que podem ser melhorados, tanto em termos de experiência de utilização como de funcionalidade do *software*.

- Como tem sido referido, o *Time Link* armazena informação proveniente de ficheiros *Kleio* e agora de também de ficheiros GEDCOM. Relativamente à informação *Kleio*, as famílias eram sempre definidas por relações de parentesco, o que levou à decomposição do *record* famílias respetivo ao GEDCOM. Estas famílias provenientes do ficheiro GEDCOM podem ser diretamente importadas sem ser necessária decomposição, alterando o importador da aplicação;
- Implementação de uma biblioteca *JavaScript*, ou de outro recurso (por exemplo aplicação flash) para o desenho de árvores genealógicas, permitindo uma representação completa e mais interativa.

7.5. Testes – Edição *inline*

Tal como nas fases de teste anteriores foi utilizada a *Framework* JUnit para a realização de testes unitários e de integração.

7.5.1. Testes unitários e de integração

Seguindo a estrutura das fases de testes anteriores, foram implementadas algumas classes do mesmo tipo, e utilizados ficheiros *Kleio* distintos fornecidos pela empresa.

Teste	Ficheiro *Kleio*	Resultado
Abrir ficheiro	alarcoes17.cli	✓
Gavar ficheiro	alarcoes17.cli	✓
Abrir ficheiro	santamaria-1785.cli	✓
Gavar ficheiro	santamaria-1785.cli	✓
Abrir ficheiro	governo-01.cli	✓
Gavar ficheiro	governo-01.cli	✓

Tabela 25 – Testes unitários e de integração edição *inline*.

Da mesma forma, é testada a solução do problema e a sua integração no processo de edição *inline*, verificando assim o seu comportamento geral.

Respetivamente a testes com *highlight* de sintaxe, para além a utilização das *keywords* presentes no ficheiro mhk.xml anteriormente referido, foram também testados alguns ficheiros de forma a tentar garantir o correto funcionamento do *software*.

Teste	Ficheiro *Kleio*	Resultado
Highlight	alarcoes17.cli	✓
Highlight	santamaria-1785.cli	✓
Highlight	governo-01.cli	✓

Tabela 26 – Testes *highlight* edição *inline*.

Para além destes testes foram também testadas algumas codificações diferentes para perceber as limitações a este nível.

Codificação	Resultado
ANSEL	✓
ASCII	✓
UTF-8	✓

Tabela 27 – Testes de codificações edição *inline*.

Como podemos verificar conseguimos abranger algumas codificações de ficheiros, suficiente para alcançar os objetivos pretendidos.

7.5.2. Testes de sistema

Nesta fase foi testada esta funcionalidade executando as funções abrir, editar e gravar ficheiro, tal como o *hightlight* da sintaxe, sobre os *Web Browsers* mais utilizados hoje em dia, pois pretendemos resolver o problema em questão de forma genérica.

Browser	Resultado
Safari 5.1.7	✓
Google Chrome 27.0.1453.94	✓
Mozilla Firefox 21.0	✓
Internet Explorer 10.0.5	x

Tabela 28 – Testes Web Browsers edição *inline*.

Apesar da maior parte dos *Web Browsers* testados funcionarem corretamente, a biblioteca CodePress não suporta versões do Internet Explorer superiores à 7.0, pelo que o funcionamento do *highlight* de sintaxe não funciona corretamente em versões recentes deste *Web Browser*.

7.5.3. Testes de aceitação

Em moldes diferentes das funcionalidades anteriores estes testes e sua validação foram efetuados pelo programador e pelo gestor de *software* da empresa em separado, pois a simplicidade do processo a implementar assim o permitiu.

Nesta fase não foram identificados problemas a corrigir, sendo então a componente de *software* aprovada.

7.5.4. Exemplo de utilização

Na figura seguinte podemos observar a página de *Web* do *Time Link* onde o utilizador pode visualizar o ficheiros existentes na sua pasta mhk_docs, podendo agora clicar no nome dos ficheiros.

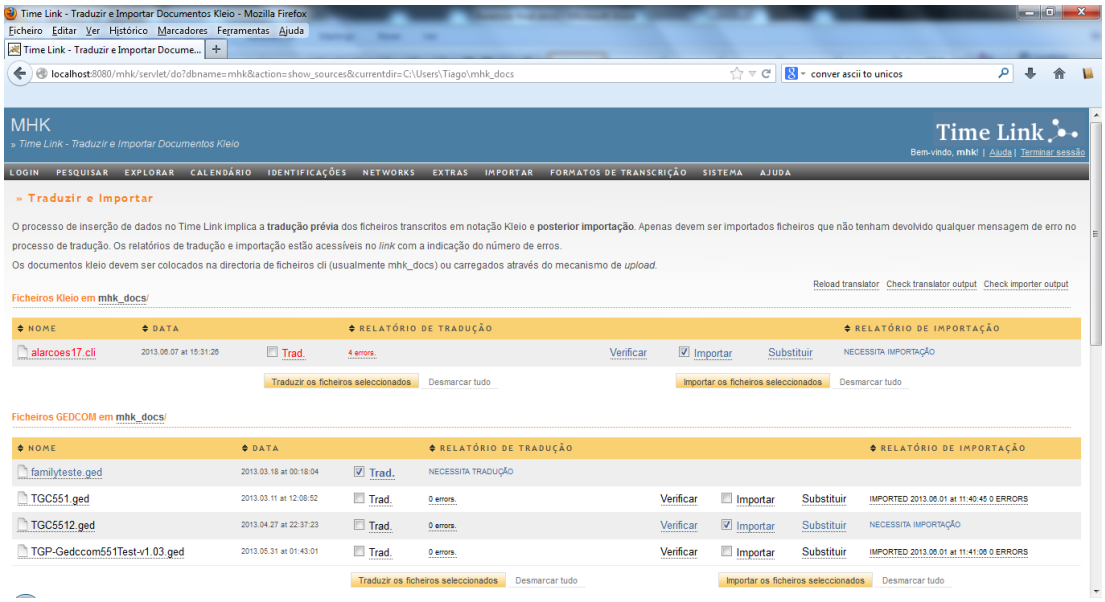

Figura 34 – Lista de ficheiros na pasta mhk_docs.

Após a escolha do ficheiro, o utilizador é direcionado para a página de edição com o conteúdo do ficheiro já carregado na área de texto, incluindo o *highlight* de sintaxe da linguagem.

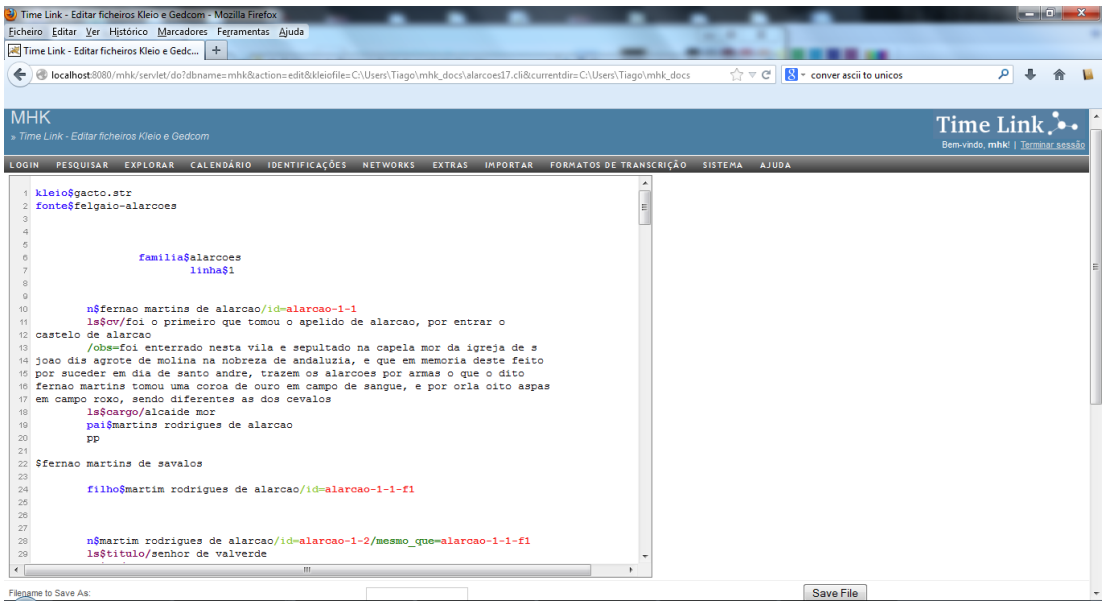

Figura 35 – Exemplo edição *inline*.

Escrevendo o nome do novo ficheiro e clicando no botão "guardar", este é armazenado no disco.

7.5.5. Problemas identificados

Durante as fases de desenvolvimento de *software* e testes foram verificados os seguintes problemas:

- No *Web Browser* Internet Explorer em versões acima da 7.0, o *highlight* da sintaxe não funciona corretamente pois a biblioteca CodePress não suporta versões recentes deste *Browser*.

Vistos estes problemas vamos agora fazer uma breve reflexão sobre trabalhos possíveis para melhorar esta componente.

7.5.6. Trabalhos possíveis

Nesta fase vamos identificar alguns pontos que podem ser melhorados em termos de funcionalidade do *software*.

- A biblioteca CodePress contém vários *engines* que são utilizados consoante o *User Agent* (*Web Browser*) através do qual é acedido. Assim sendo poderia ser implementado um novo ficheiro deste tipo que suportasse versões mais recentes do Internet Explorer (acima da 7.0).

7.6. Testes – Introdução de dados no *Web Browser*

Tal como nas fases de teste anteriores foi utilizada a *Framework* JUnit para a realização de testes unitários e de integração.

7.6.1. Testes unitários e de integração

Seguindo uma estrutura semelhante às fases de teste anteriores, foram implementadas algumas classes deste tipo que simulam a criação de objetos JSONObject e efetuam os de exportação de ficheiros com informação variada.

Teste	Objeto criado	Resultado
Exportar ficheiro	Simple	✓
Importar ficheiro	Simple	✓
Exportar ficheiro	Complex	✓
Importar ficheiro	Complex	✓

Tabela 29 – Testes unitários e de integração introdução dados *Web Browser*.

Foi também testada a integração de cada passo do processo de introdução de dados no *Web Browser*, verificando assim o seu comportamento geral. É importante referir que os objetos criados Simple e Complex têm em conta o número e variedade de pessoas, atributos, relações, atos e funções.

7.6.2. Testes de sistema

Nesta fase foi testada esta funcionalidade executando as funções de criação da fonte de informação e envio da informação para o servidor ficheiro sobre os *Web Browsers* mais utilizados hoje em dia, pois pretendemos resolver o problema em questão de forma genérica.

Browser	Resultado
Safari 5.1.7	✓
Google Chrome 27.0.1453.94	✓
Mozilla Firefox 21.0	✓
Internet Explorer 10.0.5	✓

Tabela 30 – Testes *Web Browsers* introdução dados *Web Browser*.

Como podemos observar na tabela anterior, a construção da fonte de informação na camada de visualização e o envio desta informação para o servidor através de um pedido AJAX, são efetuados de forma genérica para os principais *Web Browsers* utilizados hoje em dia.

7.6.3. Testes de aceitação

Semelhante à funcionalidade anterior, estes testes e sua validação foram efetuados pelo programador e pelo gestor de *software* da empresa em separado, pois a simplicidade do processo a implementar assim o permitiu.

Nesta fase não foram identificados problemas a corrigir, sendo então a componente de *software* aprovada.

7.6.4. Exemplo de utilização

Na figura seguinte podemos observar a página de *Web* do *Time Link* onde o utilizador pode criar as suas fontes e exportar estas para XML na pasta mhk_docs. Para cada tipo de tipo de informação a inserir, existe uma tabela à qual o utilizador vai adicionando linhas, construindo a fonte de informação com relações entre entidades (pessoas e atos), assim como atributos das mesmas.

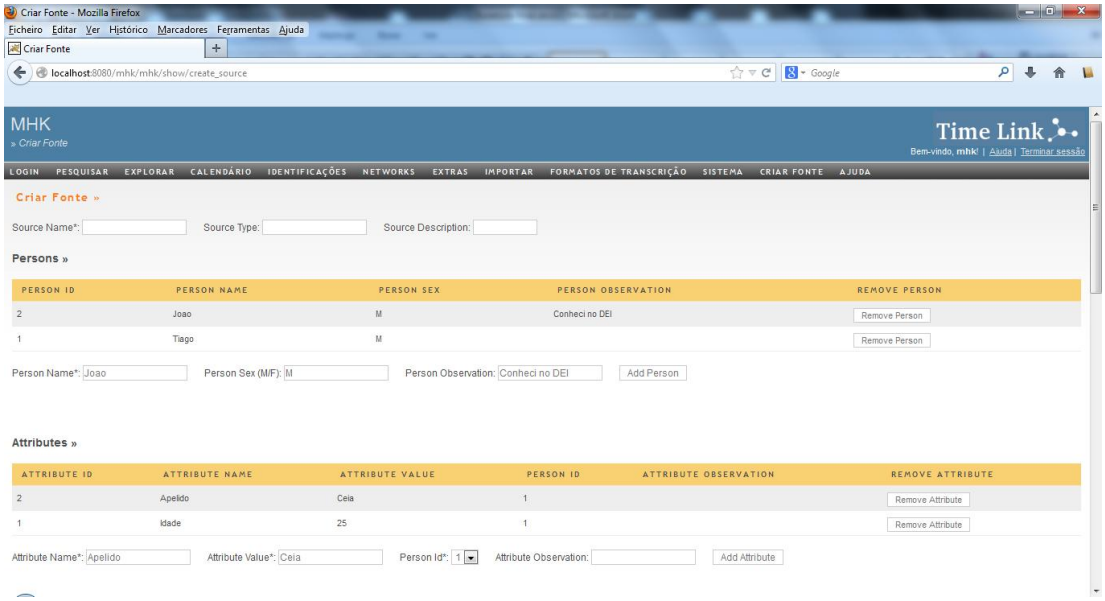

Figura 36 – Exemplo de utilização introdução de dados no *Web Browser*.

Após a criação desta fonte de informação o utilizador pode clicar no botão de importação, sendo então o ficheiro armazenado no dico através do servidor e importado na base de dados.

7.6.5. Problemas identificados

Nesta funcionalidade não foram encontrados problemas em termos das funções executadas durante o processo implementado.

7.6.6. Trabalhos possíveis

Para além da componente de *software* desenvolvida, seria interessante para o Historiador que página *Web* de criação de fontes oferecesse algum *feedback* sobre as normas utilizadas em cada tipo de documento. Apesar de neste momento oferecer-mos ao utilizador liberdade nos tipos de atributos e relações, caso o utilizador escolhesse um dos tipos de fontes *Kleio* bem conhecidas (estas estão definidos num ficheiro de nome gacto.str), poderia ser informado das normas em tempo real. [37]

8. Conclusão

Após uma redefinição do planeamento, este foi seguido sem desvios significativos. Ainda assim as propostas de elaboração de um *package* de instalação para sistema operativo *MAC*, assim como a implementação do suporte à importação de ficheiros no formato GEDCOM, que não estavam inicialmente planeadas, levaram a alguma adaptabilidade por parte do estagiário.

Sobre o trabalho desenvolvido neste estágio, achamos que está de acordo com o que foi planeado, conseguindo mostrar alguma autonomia por parte do estagiário bem como a complexidade do trabalho efetuado. Apesar dos riscos identificados, as estratégias de mitigação foram aplicadas com sucesso, fazendo com que os objetivos fossem alcançados e traduzindo estes desenvolvimentos numa boa experiência para o estagiário.

Com a introdução destas novas funcionalidades e dos *packages* de instalação consegui-mos quebrar barreiras ao nível da usabilidade/instalação da aplicação e importação/visualização de informação, justificando assim o trabalho desenvolvido ao longo do ano letivo.

Tendo em conta que a importação GEDCOM promove a utilização do *Time Link* não reduzindo este à informação *Kleio*, a visualização da árvore genealógica traz uma nova perceção sobre os dados na aplicação, a edição *inline* liberta os utilizadores da utilização do editor de texto jEdit e a instalação da aplicação em sistemas operativos *Windows* e *MAC* é agora trivial, podemos concluir que foram desenvolvidas mais-valias para a aplicação.

Numa perspetiva mais geral sobre o estágio, achamos que esteve dentro das espectativas, fazendo com que o estagiário assuma responsabilidades ao nível de decisões de engenharia de *software* e planeamento bem como experiência em todas as fazes de desenvolvimento de *software*, trazendo maior relevância a esta disciplina de estágio em engenharia informática.

9. Referências

[1] – Diário As Beiras, 2012. Disponível em: www.asbeiras.pt/2012/08/iclio-expande-se-para-o-brasil/, acedido em Setembro, 2012.

[2] – Colaborativa de código *googlecode*. Disponível em: http://code.google.com/p/mhk/, acedido em Setembro, 2012.

[3] – Plataforma de gestão *Basecamp*. Disponível em: http://basecamp.com/, acedido em Setembro, 2012.

[4] – *Issues googlecode*. Disponível em: http://code.google.com/p/mhk/issues/list , acedido em Setembro, 2012.

[5] – Imagem Google. Disponível em: Imagem, acedido em Outubro, 2012.

[6] – *Record linking*. Disponível em: http://en.wikipedia.org/wiki/Record_linkage, acedido em Outubro, 2012.

[7] – Estudos de aplicações sobre genealogia. Disponível em: http://familytreemagazine.com/article/best-big-genealogy-websites-2012, acedido em Outubro, 2012.

[8] – *Web page* ancestry.com. Disponível em: http://www.ancestry.com/, acedido em Outubro, 2012.

[9] – *Web page* archives.com. Disponível em: http://www.archives.com/ acedido em Outubro, 2012.

[10] – Boom de subscrições. Disponível em: http://familytreemagazine.com/article/best-big-genealogy-websites-2012, acedido em Outubro, 2012.

[11] – *Web page* genealogybank.com. Disponível em: http://www.genealogybank.com/gbnk/, acedido em Outubro, 2012.

[12] – *Web page* familysearch.org. Disponível em: https://familysearch.org/, acedido em Outubro, 2012.

[13] – *Web page* dyncoopnet.org disponível em: http://www.dyncoopnet-pt.org/, acedido em Outubro, 2012.

[14] – Estudo efetuado pelo *Web Site* toptenreview.com, 2012. Disponível em: http://genealogy-search-review.toptenreviews.com/, acedido em Outubro, 2012.

[15] – João Carvalho, 2009, Faculdade de Letras da Universidade de Coimbra, "Time Link: a evolução de uma base de dados prosopográfica".

[16] – Ramos de Carvalho, 1997, de Letras da Universidade de Coimbra, "Comportamentos Morais e Estruturas Sociais numa paróquia de Antigo Regime (Soure, 1680-1720)".

[17] – Joaquim Carvalho, Universidade de Coimbra, Rosário Campos, Escola Superior de Educação de Coimbra, 2005, "Interpersonal networks and the archaeology of social structures; using social positional events to understand social strategies and individual behavior".

[18] – Ramos de Carvalho, 2004, Faculdade de Letras da Universidade de Coimbra, "Micro-Historical Perspectives on Moral Choices: Case Studies from Early Modern Portugal".

[19] – S*oftware Advanced Installer*. Disponível em: http://www.advancedinstaller.com/, acedido em Novembro, 2012.

[20] – S*oftware Install Editor*. Disponível em: http://www.instedit.com/, acedido em Novembro, 2012.

[21] – S*oftware Inno Setup*. Disponível em: http://www.jrsoftware.org/isinfo.php, acedido em Novembro, 2012.

[22] – S*oftware NSIS*. Disponível em: http://nsis.sourceforge.net/Main_Page, acedido em Novembro, 2012.

[23] – S*oftware VISE X*. Disponível em: http://www.mindvision.com/vise_x.asp, acedido em Dezembro, 2012.

[24] – S*oftware Install Builder*. Disponível em: http://installbuilder.bitrock.com/, acedido em Dezembro, 2012.

[25] – S*oftware Iceberg*. Disponível em: http://s.sudre.free.fr/Software/Iceberg.html, acedido em Janeiro, 2013.

[26] – *GEDCOM 5.5.1*. Disponível em: http://wiki.webtrees.net/w/images-en/Ged551-5.pdf, acedido em Fevereiro, 2013.

[27] – Biblioteca FamilyTree.js. Disponível em: https://github.com/chandlerprall/FamilyTreeJS, acedido em Fevereiro, 2013.

[28] – Biblioteca *JavaScript* CodePress. Disponível em: http://sourceforge.net/projects/codepress/, acedido em Fevereiro, 2013.

[29] – Biblioteca *Java* JDOM Disponível em: http://www.jdom.org/, acedido em Fevereiro, 2013.

[30] – JSONObject. Disponível em: http://www.json.org/js.html, acedido em Março, 2013.

[31] – Biblioteca *Java* JSON.simple. Disponível em: http://code.google.com/p/json-simple/, acedido em Março, 2013.

[32] – *Framework* JUnit. Disponível em: http://junit.org, acedido em Março, 2013.

[33] – Eclipse IDE. Disponível em: http://www.eclipse.org/downloads/, acedido em Março, 2013.

[34] – Casos de teste GEDCOM. Disponível em: http://www.geditcom.com/gedcom.html, acedido em Março, 2013.

[35] – Casos de teste GEDCOM. Disponível em: http://timforsythe.com/tools/downloads, acedido em Março, 2013.

[36] – Aplicação *MyFamilyTree*. Disponível em: http://chronoplexsoftware.com/myfamilytree/, acedido em Abril, 2013.

[37] – Ficheiro gacto.str. Disponível em: https://code.google.com/p/mhk/source/browse/trunk/clio/src/gacto.str, acedido em Abril, 2013.

Anexos