Mestrado em Engenharia Informática
Estágio

Relatório Final

TIME LINK Developer
(Aplicações Web em Java)

Paulo Alexandre Duarte da Silva
padsilva@student.dei.uc.pt

Orientador DEI:
Prof. Doutor Mário Zenha Rela

Orientador iClio:
Prof. Doutor Joaquim Ramos de Carvalho

Data: 2 de Setembro de 2015

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

Mestrado em Engenharia Informática
Estágio

Relatório Final

TIME LINK Developer
(Aplicações Web em Java)

Paulo Alexandre Duarte da Silva
padsilva@student.dei.uc.pt

Orientador DEI:
Prof. Doutor Mário Zenha Rela

Orientador iClio:
Prof. Doutor Joaquim Ramos de Carvalho

Júri Arguente:
Prof. Raul Barbosa

Júri Vogal:
Prof. Paulo Simões

Data: 2 de Setembro de 2015

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

Resumo

O presente documento descreve os resultados do trabalho desenvolvido no âmbito do estágio do Mestrado em Engenharia Informática. O âmbito deste estágio consistiu no desenvolvimento de uma aplicação que possibilite ao utilizador acelerar o processo de introdução dos dados no sistema TIME LINK.

O TIME LINK é um sistema informático desenvolvido pelo orientador externo, Doutor Joaquim Ramos de Carvalho, e que serve para apoiar investigações académicas. Este sistema é baseado em interfaces *web* e é bastante eficaz na recolha e tratamento de dados biográficos presentes em diversas fontes e no cruzamento de toda esta informação. Para que a informação seja populada na base de dados, é necessária a transcrição dos dados biográficos através de uma notação especial denominada de Kleio. Esta notação segue uma estrutura de modelo de dados que se baseia em três conceitos: grupos, unidades nucleares de recolha de informação; elementos, descrevem os grupos; e aspetos, atribuem valores aos elementos.

A transcrição dos dados biográficos não é um processo simples e é objetivo deste estágio criar uma aplicação que interaja com o utilizador para facilitar todo este processo, através de um editor de código que suporte a notação Kleio e permita manipular de forma intuitiva os dados que são transcritos.

Palavras-Chave

"TIME LINK", "Kleio", "Editor", "Aplicação *Web*", "Engenharia de *Software*"

Abstract

This report relates to the results of the work developed in the curricular internship of the Master's Degree in Computer Science. The aim of this internship was to develop an application that allows the user to accelerate the process of data input into the TIME LINK system.

The TIME LINK is a software application that supports academic research and is developed by the external mentor, Dr. Joaquim de Carvalho Ramos. The application is based on web interfaces and is quite effective in collecting and processing biographical data from different sources and in crossing all this information. In order to information be added into the database, it is necessary the transcription of biographical data to a special notation called Kleio. This notation follows a data model structure which is based on three concepts: groups, the nuclear unit of information collection; elements, which describe the groups; and aspects, which assign values to the elements.

The transcription of the biographical data is not a simple process and it was the aim of this internship to create an application that interacts with the user in order to facilitate this process. This aim is achieved through a code editor that supports Kleio notation and allows the manipulation of the transcribed data in an intuitive manner.

Keywords

"TIME LINK", "Kleio", "Editor", "Web Application", "Software Engineering"

Agradecimentos

Durante o meu percurso académico tive contato com diversas pessoas e de, alguma forma, todas contribuíram para que este fosse concluído com sucesso.

Gostaria de começar por agradecer ao *Chief Executive Officer* (CEO) da iClio, Alexandre Pinto, por toda a disponibilidade. Ao Exmo. Senhor Vice-Reitor da Universidade de Coimbra, Doutor Joaquim Ramos de Carvalho, pela confiança que depositou em mim para a realização deste projeto. À Doutora Ana Isabel Ribeiro pela disponibilidade e paciência.

Fica também o agradecimento ao Professor Doutor Mário Rela por toda a ajuda e compreensão ao longo desta etapa. Caro Professor, obrigado por toda a disponibilidade e por ter acreditado em mim.

Aos meus amigos de sempre, pela paciência e por nunca se terem esquecido de mim. Mesmo que passem dias, semanas ou meses sem vos ver, vocês são parte importante da minha vida!

Aos meus familiares, pelo apoio constante e por me terem proporcionado esta experiência enriquecedora.

À Helena, pela paciência, compreensão, palavra amiga, carinho e amizade.

Obrigado a todos!

Índice

Capítulo 1 Introdução 1
1.1. Contexto 1
1.2. Objetivos 2
1.3. Estrutura do Documento 2
Capítulo 2 Metodologia e Planeamento 4
2.1. Metodologia 4
2.2. Planeamento 4
2.2.1. Primeiro Semestre 4
2.2.2. Segundo Semestre 6
2.2.3. Desvios 7
2.3. Análise de Riscos 8
Capítulo 3 Estado da Arte 11
3.1. Estudo do TIME LINK 11
3.1.1. Características do Sistema 11
3.1.2. Descrição da Notação Kleio 14
3.1.3. Necessidades Encontradas 16
3.2. Estudo das Aplicações Concorrentes 17
3.2.1. IDEs *Web-based* 17
3.2.2. Análise Comparativa 18
3.2.3. Ideias Retiradas 19
3.3. Soluções Encontradas 19
Capítulo 4 Especificação 21
4.1. Requisitos 21
4.1.1. Levantamento e Validação 21
4.1.2. Especificação 21
4.2. Diagrama de Casos de Uso 23
4.3. *Mockups* 24
4.4. Arquitectura do Sistema 25
4.4.1. Modelo Arquitetural Final 25
4.4.2. Modelo Arquitetural com Edição Colaborativa 27
Capítulo 5 Desenvolvimento 29

5.1. Pedidos ao Servidor 29
5.2. Funcionalidades 30
5.2.1. Editor de Texto 30
5.2.2. Editar Ficheiros 30
5.2.3. Gravar Ficheiros 30
5.2.4. *Autocomplete* 31
5.2.5. Realce da Sintaxe 32
5.2.6. Sugestões 32
5.2.7. Edição Colaborativa 33
5.3. Testes e Validação 33
5.3.1. Testes Unitários e de Integração 33
5.3.2. Testes de Sistema 34
5.3.3. Testes de Aceitação dos Requisitos Funcionais 34
5.3.4. Testes de Aceitação dos Requisitos Não-Funcionais 35
Capítulo 6 Conclusões 37
6.1. Reflexões Finais 37
6.2. Contribuições 39
6.3. Trabalho Futuro 39
Referências 40

Lista de Figuras

Figura 1: Arquitetura do sistema TIME LINK 12
Figura 2: Diagrama de casos de uso 23
Figura 3: Mockup para editar um ficheiro 24
Figura 4: Arquitetura da aplicação 25
Figura 5: Lógica de negócio do Ace 26
Figura 6: Arquitetura da solução com edição colaborativa 27
Figura 7: Exemplo de utilização do editor 31
Figura 8: Exemplo de sugestões 33

Lista de Tabelas

Tabela 1: Risco nº1 – Local de Trabalho 8
Tabela 2: Risco nº2 - Comunicação 9
Tabela 3: Risco nº3 - Documentação....9
Tabela 4: Risco nº4 - Situação Académica 10
Tabela 5: Análise comparativa das aplicações concorrentes estudadas. 18
Tabela 6: Requisitos Funcionais 22
Tabela 7: Requisitos não-funcionais....23
Tabela 8: Regras de realce da sintaxe 32
Tabela 9: Tabela de modelo dos testes de sistema....34
Tabela 10: Resultados dos testes de aceitação dos requisitos funcionais 34
Tabela 11: Resultado dos testes do requisito não funcional - Portabilidade....35

Acrónimos

AJAX - Asynchronous Javascript and XML

BSD - Berkeley Software Distribution

CD - Compact Disc

CEO - Chief Executive Officer

CSS - Cascading Style Sheets

FCFS - First-Come, First-Served

FCTUC - Faculdade de Ciências e Tecnologia da Universidade de Coimbra

FLUC - Faculdade de Letras da Universidade de Coimbra

FTP - File Transfer Protocol

GUI - Graphical User Interface

HTML - HyperText Markup Language

HTTP - HyperText Transfer Protocol

ID - IDentifier

IDE - Integrated Development Environment

IPN - Instituto Pedro Nunes

JSON - JavaScript Object Notation

MIT - Massachusetts Institute of Technology

MVC - Model–View–Controller

PC - Personal Computer

PHP - PHP: Hypertext Preprocessor

SFTP - Secure File Transfer Protocol

URI - Uniform Resource Identifier

URL - Uniform Resource Locator

XML - eXtensible Markup Language

Capítulo 1
Introdução

Este relatório serve de suporte ao trabalho efetuado pelo aluno Paulo Alexandre Duarte da Silva no âmbito da unidade curricular anual Dissertação/Estágio do Mestrado em Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra (FCTUC), no ano letivo de 2014/2015.

O estágio teve a duração de dois semestres, sendo que no primeiro foi definida a metodologia e o planeamento, estudado o estado da arte, realizado o levantamento e especificação de requisitos. No segundo semestre foi desenhada a arquitetura da aplicação e procedeu-se ao desenvolvimento do projeto.

Este capítulo está dividido em três secções: contexto, objetivos pretendidos e termina com um resumo da estrutura deste documento.

1.1. Contexto

A iClio [1], empresa sediada no Instituto Pedro Nunes (IPN), que é incubadora de empresas focadas na área da tecnologia, foi fundada em 2010 por investigadores da Faculdade de Letras da Universidade de Coimbra (FLUC) e é uma empresa dedicada à criação e publicação de conteúdos relacionados com a história, património e cultura.

"iClio's mission is to build bridges between content and technology, focusing on accuracy, usability, design quality and effective project management. Clio, the muse of history, is now ready to discover new environments, becoming iClio – the muse of history, culture and heritage for the new digital media." [2]

Tal como a sua missão diz, a iClio compromete-se em construir ligações entre a informação e a tecnologia, focando-se no rigor, usabilidade e qualidade do desenvolvimento dos projetos.

O sistema TIME LINK [3], no qual a iClio contribui ativamente com código, assenta sobre a recolha e tratamento de informação biográfica dispersa por várias fontes e suporta um processo reversível de reconstrução dessa informação. Este sistema tem sido usado por investigadores na área da história social, por instituições culturais e em projetos a nível municipal visando a reconstituição de comunidades de pessoas que viveram no passado. De seguida é descrito um cenário típico do TIME LINK.

Cenário Típico

Depois da prévia recolha das fontes que vão fornecer a informação biográfica e de determinado o âmbito cronológico que determina os limites temporais dessa recolha, é iniciada a fase de transcrição documental para suporte informático. Esta transcrição é baseada em textos anotados, isto é, seguem uma notação especial Kleio (que será abordada na subsecção 3.1.2.), e escrita mediante um processador de texto comum.

Tomemos, como exemplo, o seguinte texto anotado referente a um batismo:

```
n$joao/m
        pn$manuel
                ls$morada/coimbra
        mn$maria
        pad$paulo
                ls$morada/lisboa
        mad$ana
                ls$titulo/dona
                ls$morada/coimbra
```

Este exemplo diz respeito ao batismo de João (filho de Manuel, que reside em Coimbra, e Maria) e que tem como padrinhos Paulo, que mora em Lisboa, e Dona Ana, residente em Coimbra.

Como podemos constatar, esta notação permite conservar na transcrição informática uma forma semelhante à fonte original. No entanto, a transcrição destas fontes de informação biográfica tem uma curva de aprendizagem muito alta.

Depois de finalizado o processo de transcrição é necessário adicionar os dados no sistema TIME LINK, ou seja, popular a base de dados com a informação biográfica, para que possam ser posteriormente consultados e manipulados. Para isso o utilizador terá que carregar os ficheiros, que se encontram localmente e que contêm a informação biográfica transcrita na notação Kleio, no sistema. Todo este processo tem que ser repetido quando existem alterações ao ficheiro que se quer adicionar, o que constitui uma limitação.

1.2. Objetivos

O objetivo deste estágio passa por encontrar soluções que otimizem a transcrição dos dados biográficos, dado que, como foi referenciado anteriormente, todo este processo tem uma curva de aprendizagem muito acentuada, o que facilita a geração de erros, tornando-o num processo demorado e de difícil compreensão.

O utilizador irá conseguir editar código através de um interface web que produza feedback e com elevado grau de interatividade. Para além disso, o utilizador não terá que se preocupar em carregar os ficheiros locais sempre que estes são alterados, visto ser possível, aquando da edição, gravar o conteúdo no ficheiro que está a ser editado.

1.3. Estrutura do Documento

Definido o âmbito do projeto e os seus principais objetivos é agora feito um resumo da estrutura do presente documento e dos principais temas abordados, de modo a facilitar a compreensão deste relatório.

Capítulo 1 – introdução ao projeto, dando a conhecer a entidade acolhedora, o seu enquadramento e quais os objetivos pretendidos. Também é abordada a estruturação deste documento;

Capítulo 2 – neste capítulo é abordada a metodologia usada, definido o planeamento para o estágio e ainda é feita uma análise aos riscos inerentes a este tipo de projeto;

Capítulo 3 – o estado da arte encontra-se no terceiro capítulo. Neste capítulo é feita a especificação do sistema TIME LINK, analisado o estudo das aplicações concorrentes à aplicação que será desenvolvida e apresentadas as soluções encontradas;

Capítulo 4 – neste capítulo encontra-se a especificação da aplicação, nomeadamente o levantamento e especificação de requisitos, diagramas de casos de uso, mockups da aplicação e arquitetura do sistema;

Capítulo 5 – são apresentadas em detalhe todas as decisões de implementação referentes a cada requisito funcional e os respetivos testes efetuados;

Capítulo 6 – refere-se às conclusões que se tiram do trabalhado realizado, às contribuições do estagiário e é feita uma reflexão sobre o trabalho futuro.

Capítulo 2
Metodologia e Planeamento

Neste capítulo será descrita a metodologia adotada, abordado todo o planeamento feito para o 1º e 2º semestre, divididos, respetivamente, por duas secções. Todos e quaisquer desvios ao planeamento serão também reportados devidamente. Na última secção serão enumerados os riscos associados ao projeto de desenvolvimento da aplicação.

2.1. Metodologia

A metodologia de trabalho aplicada neste estágio segue o modelo *Scrum* [4], isto é, segue uma abordagem ágil e iterativa que divide o projeto em pequenos ciclos (*sprints*). Esta permite assim um acompanhamento mais eficiente da evolução do projeto. Destas divisões resultaram cinco fases distintas: análise das funcionalidades existentes e soluções encontradas, levantamento e especificação de requisitos, desenvolvimento da solução, realização de testes e elaboração do relatório final de estágio. Ao longo destas fases foram agendadas reuniões mensais onde o estagiário apresentou o trabalho desenvolvido e delineado o trabalho a ser realizado na fase seguinte.

Referir ainda que a fase de desenvolvimento da solução foi dividida em pequenos *sprints*. Nesta fase foi utilizado o sistema de controlo de versões Git [5], através do repositório de código GitHub [6].

2.2. Planeamento

Nesta secção constam os planeamentos para cada semestre e as respetivas alterações/desvios dos mesmos.

2.2.1. Primeiro Semestre

Nesta subsecção são listadas e descritas as fases que foram delineadas para o primeiro semestre.

Enquadramento e Objetivos

Nesta fase foi realizada uma reunião formal com o cliente, onde foi apresentada a proposta, feito o enquadramento do projeto e discutidos os objetivos pretendidos para o estágio.

Análise e Estudo do TIME LINK

Nesta fase o pretendido era que o estagiário ficasse a par das funcionalidades do sistema TIME LINK pois é à volta deste que o estágio se vai realizar. Para isso foram fornecidas algumas teses sobre o mesmo [7] [8] e realizadas algumas experiências/testes na aplicação, para assim perceber quais as funcionalidades que a mesma oferece aos seus utilizadores.

Formação de Tecnologias *FrontEnd*

Devido aos objetivos do estágio estarem intimamente ligados a interfaces web e estes provavelmente utilizarem linguagens como o *HyperText Markup Language* (HTML), *Cascading Style Sheets* (CSS) e JavaScript sentiu-se a necessidade de fazer uma semana de formação, pois o estagiário não se sente à vontade com as mesmas. O estagiário seguiu alguns tutoriais que se encontram na Internet. [9] [10]

Análise e Levantamento do Estado da Arte

Toda a análise e levantamento do estado da arte ajuda-nos a perceber o que as aplicações, líderes na área em estudo, oferecem aos seus utilizadores e que fazem delas um exemplo a seguir. Também foi importante para encontrar a proposta de solução de implementação da aplicação.

Levantamento de Requisitos

Esta é uma fase muito importante pois leva-nos a perceber quais os requisitos funcionais e não-funcionais desejados para a aplicação. Para além disso, a priorização dos mesmos também é um fator determinante para perceber o esforço que devemos aplicar a cada requisito.

Escrita do Relatório Intermédio

O relatório intermédio é escrito desde uma fase inicial para que no final do semestre o peso do mesmo não tenha implicações em atrasos de outras tarefas.

Criação do Diagrama de Casos de Uso

Nesta fase é desenhado o diagrama de casos de uso. Este serve para descrever o comportamento do sistema na perspetiva do utilizador, isto é, para identificar quais as funcionalidades do sistema.

Desenho das *Mockups*

Esta fase tem como propósito ilustrar alguns protótipos do produto final com todas as funcionalidades enumeradas no levantamento de requisitos.

Alteração e Validação dos Requisitos, Diagramas e *Mockups*

Depois das tarefas relacionadas com os requisitos, foi agendada uma reunião com o cliente com o propósito de mostrar toda a documentação técnica, fazer a validação e, eventualmente, algumas alterações à mesma.

Arquitetura

Esta fase contempla toda a arquitetura da aplicação, onde foi possível perceber como cada componente do sistema interage com as restantes, e consequentemente identificar quais as tecnologias necessárias para a implementação da solução final deste estágio.

Revisão do Relatório Intermédio

Depois de finalizado o relatório intermédio, este é enviado para os orientadores, será dado algum *feedback* e poderão ser feitos alguns ajustes.

Correção e Entrega do Relatório Intermédio

Depois do *feedback* recebido, caso fosse necessário, seriam feitos alguns ajustes.

Preparação da Apresentação Intermédia

Nesta fase é elaborada e preparada a apresentação intermédia.

2.2.2. Segundo Semestre

Para o 2º semestre foram planeadas as fases de desenvolvimento, testes, elaboração do relatório final de estágio e preparação da apresentação final. Segue uma breve descrição das fases referidas anteriormente.

Desenvolvimento

Nesta fase de desenvolvimento é utilizada a metodologia de trabalho descrita na secção 2.3. Esta escolha permite fazer uma lista das funcionalidades a implementar e divide a fase em pequenos *sprints*, seguindo assim um processo de desenvolvimento incremental e iterativo. Foi utilizado o sistema de controlo de versões Git, através do repositório de código GitHub.

Testes

Sempre que uma fase terminar o seu ciclo de desenvolvimento serão feitos testes de integração, para verificar se existem falhas na integração dos componentes no sistema. Também serão feitos testes de sistema que têm como finalidade conseguir perceber se existem ou não falhas, do ponto de vista do utilizador. Ainda serão feitos testes de aceitação que servem para fazer uma validação final do desenvolvimento e dos requisitos definidos. Por fim, serão feitos testes de usabilidade para se perceber se o sistema é facilmente compreensível a um utilizador comum.

Escrita do Relatório Final

Tal como no 1º semestre, o relatório final será escrito desde uma fase inicial do segundo semestre, para assim não atrasar algumas tarefas agendadas para mais tarde.

Revisão do Relatório Final

O relatório final será entregue aos orientadores com uma semana de antecedência, para que assim possam fazer revisões e dar algum *feedback* que possa ser útil para completar o relatório e não ficar aquém do esperado.

Correção e Entrega do Relatório Final

Depois do feedback recebido, caso seja necessário, serão feitos alguns ajustes. Nesta fase será necessário cumprir várias normas, como imprimir um dado nº de relatórios, encaderná-los e gravar alguns *Compact Disks* (CDs), bem como entregar todo este material na secretaria. Para além disso, a elaboração dos diapositivos da apresentação e a preparação do discurso também são levados em conta.

Preparação da Apresentação Final

Nesta fase é a preparada a apresentação final, para isso são elaborados os diapositivos de apresentação e estruturado o discurso.

2.2.3. Desvios

De seguida são enumerados os desvios que existiram ao longo do estágio.

Primeiro Semestre

Devido à época de exames de Janeiro não ter corrido como se esperava as fases que coincidiam com esta não foram cumpridas. Entre elas, a arquitetura da aplicação e todas as fases referentes ao relatório intermédio de estágio. Foi então necessário passar a fase de arquitetura da aplicação para o segundo semestre.

Segundo Semestre

Como nem tudo estava implementado até ao início de Junho e o estagiário ainda ter que realizar no mínimo quatro exames, foi decidido adiar a entrega e defesa para Setembro. Sendo assim os meses de Junho e Julho ainda foram dedicados à implementação de algumas funcionalidades e aperfeiçoamentos nas mesmas.

2.3. Análise de Riscos

Existem sempre vários riscos inerentes ao desenvolvimento de *software*, riscos esses que se não forem identificados e devidamente tratados poderão afetar o desenvolvimento do projeto. Abaixo estão detalhados os riscos que foram detetados e a forma como foram mitigados.

Antes de se definir os riscos existentes no decurso do projeto, é importante especificar os classificadores usados na definição dos mesmos e o que estes implicam. Desta forma, é necessário definir as métricas de Impacto, Probabilidade e Prazo.

Para o Impacto, os possíveis classificadores são:

- Elevado - Não se cumprem os critérios de sucesso;
- Médio - Cumprem-se os critérios de sucesso mas com grande dificuldade;
- Baixo - Cumprem-se os critérios de sucesso sem grande dificuldade.

No que diz respeito à Probabilidade, os classificadores existentes são:

- Elevada - Mais de 70% de probabilidade de ocorrer;
- Média - Entre 40% e 70% de probabilidade de ocorrer;
- Baixa - Menos de 40% de probabilidade de ocorrer.

Por fim, os classificadores existentes para caracterizar o Prazo são:

- Longo - Data da defesa intermédia;
- Médio - 1 mês;
- Curto - 2 semanas.

Definidos os parâmetros de classificação, foram estudados os possíveis riscos e foi feita a sua descrição, indicando para cada um a sua condição, consequência e classificação.

Tabela 1: Risco nº1 – Local de Trabalho

	Risco nº1
Condição	Local de trabalho é remoto.
Consequências	Afeta o processo de produção de *software*; Falta de acompanhamento; Desvios ao planeamento.
Impacto	Médio.
Probabilidade	Média.
Prazo	Longo.
Mitigação	Planeamento feito com o máximo rigor; Marcação de reuniões mensais.

Tabela 2: Risco nº2 - Comunicação

	Risco nº2
Condição	Pouca comunicação com os orientadores externos.
Consequências	Tomadas de decisão precipitadas; Falta de acompanhamento; Estimativas erróneas para determinadas tarefas.
Impacto	Médio.
Probabilidade	Alta.
Prazo	Longo.
Mitigação	Promover a comunicação entre as duas partes; Agendar reuniões todos os meses.

Tabela 3: Risco nº3 - Documentação

	Risco nº3
Condição	Escassa documentação e suporte à notação Kleio.
Consequências	Compreensão errada da notação; Funcionalidades implementadas não acrescentam valor à aplicação.
Impacto	Médio.
Probabilidade	Média.
Prazo	Curto.
Mitigação	Quando existirem dúvidas, estas devem ser esclarecidas com o orientador externo; Requisitos validados sempre com o cliente.

Tabela 4: Risco nº4 - Situação Académica

	Risco 4
Condição	Situação académica do estagiário (a frequentar 5 disciplinas no ano de estágio).
Consequências	Afeta o processo de produção de *software*; Desvios ao planeamento.
Impacto	Baixo.
Probabilidade	Alta.
Prazo	Longo.
Mitigação	Planeamento feito com o máximo rigor e cumprido à risca; Utilizar *software* Toggl [11] (monitorização de tempo despendido para uma dada tarefa).

Capítulo 3
Estado da Arte

Neste capítulo será detalhado o sistema TIME LINK e analisados os resultados da pesquisa efetuada sobre aplicações concorrentes. Tais estudos foram divididos em duas secções: Estudo do TIME LINK e Estudo das Aplicações Concorrentes.

3.1. Estudo do TIME LINK

Nesta secção que é apresentado o TIME LINK. Na primeira subsecção serão caracterizadas as especificidades do sistema; na subsecção 3.1.2. será feita uma descrição detalhada da linguagem de transcrição usada pelo sistema, denominada de Kleio; e na terceira, e última, subsecção serão identificadas as potenciais melhorias a implementar no âmbito deste estágio.

3.1.1. Características do Sistema

Nesta subsecção serão especificadas as finalidades do TIME LINK, a sua arquitetura de sistema e as principais funcionalidades do mesmo.

Propósito

O TIME LINK é uma aplicação implementada com base em interfaces *web*, no entanto, é necessário ser instalada localmente, isto é, nas máquinas dos utilizadores, pois o TIME LINK ainda não é um produto comercializado e não se encontra disponibilizado para livre utilização na Internet. A instalação local permite executar operações que as outras bases de dados de teste, disponibilizadas na Internet, não permitem executar. Foi desenvolvido principalmente para suportar investigações académicas (como, por exemplo, estudos genealógicos), mas também pode ser utilizado por qualquer outro utilizador interessado. As suas particularidades são: a eficaz recolha e tratamento de informação biográfica dispersa em variadíssimas fontes, o acesso a mecanismos de navegação através de redes complexas de relação interpessoais e a reconstrução biográfica por cruzamento de informação, sendo possível, interativamente, adicionar e remover informação.

Arquitetura do TIMELINK

O sistema TIME LINK segue um modelo de implementação *Model–View–Controller* (MVC) cujo principal objetivo é a separação da lógica interna de negócio de uma aplicação informática, da sua lógica de apresentação e interação. Assim as alterações feitas a *layouts* não afetam a manipulação de dados e estes podem ser reorganizados também sem afetar os *layouts*. Para além disso, a melhor organização que este modelo oferece é uma vantagem para que equipas de diferentes áreas possam trabalhar em simultâneo e melhora a escalabilidade do código produzido.

Este modelo divide a aplicação em três camadas distintas:

- Apresentação: é responsável por toda a interface da aplicação e obtém a informação que necessita solicitando a camada de modelos, mediante a camada de controlo;
- Controlo: serve como intermediário entre a camada de apresentação e de modelo. É aqui que todas as decisões são tomadas, de acordo com a lógica de negócio;
- Modelo: nesta camada acede-se a toda a informação armazenada na base de dados. Aqui é manipulada essa informação e realizadas operações como, por exemplo, consultas.

Na figura 1 é apresentada a esquematização da arquitetura do sistema e, de seguida, a explicação da interação entre as várias componentes do mesmo.

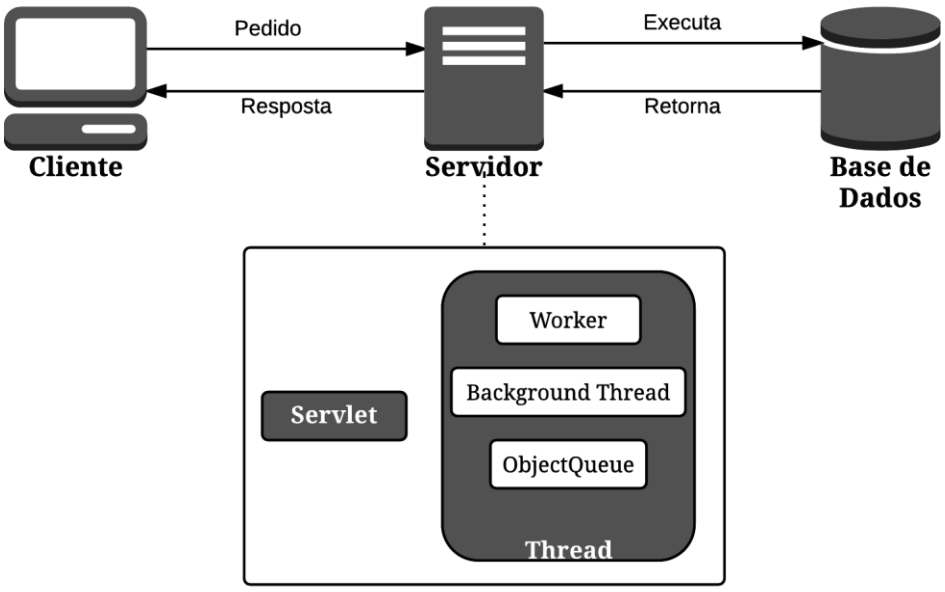

Figura 1: Arquitetura do sistema TIME LINK

Na figura 1 identificam-se, inequivocamente, três diferentes componentes: cliente, servidor e base de dados. Estas componentes são parte essencial do sistema e por isso serão analisadas e detalhadas nos parágrafos abaixo.

Como podemos constatar, o cliente executa pedidos *HyperText Transfer Protocol* (HTTP) ao servidor, através de um navegador comum. O servidor trata de todos os pedidos e executa-os, através de operações à base de dados. Este ainda processa os dados provenientes da base de dados e encaminha para o cliente, em forma de resposta HTTP. Na base de dados está contida toda a informação e esta é enviada ao servidor, consoante os pedidos executados.

Cliente

Esta componente, que corresponde à camada de apresentação do modelo MVC, trata da interface gráfica de utilizador ou *Graphical User Interface* (GUI), única face visível de todo o sistema. Aqui são apresentadas todas as informações ao utilizador final. A instanciação desta camada é feita pelo navegador.

Para que o navegador interprete toda a informação e que esta seja visualizada de forma compreensiva e intuitiva foi utilizada a linguagem HTML (*HyperText Markup Language*), que serve para produzir páginas *web*. Para aprimorar a apresentação da informação, formatando os estilos da página e não o conteúdo da informação, foram utilizadas as linguagens CSS e JavaScript. A linguagem JavaScript ajuda também a executar pedidos assíncronos ao servidor. Ainda são utlizados *templates* Apache Velocity para implementar esta componente.

Servidor

Corresponde à camada de controlo do modelo MVC e tem como principal função controlar todo o fluxo de informação que passa pelo sistema.

As *Servlets*, tipos de classes Java, recebem os pedidos, através do protocolo HTTP, da parte do utilizador. Depois de recebidos os pedidos, o *backend* da aplicação vai entrar em ação e executar vários comandos como, por exemplo, escrever e ler informações da base de dados. Depois esses comandos são transformados em pedidos que serão enviados como resposta ao cliente, novamente através do protocolo HTTP.

Alguns dos pedidos, referidos no parágrafo anterior, dão origem a tarefas simples de serem executadas como, por exemplo, *login* ou abrir uma das páginas de navegação. Por outro lado existem também várias tarefas de um grau de complexidade e morosidade maior, como, por exemplo, importação de dados *eXtensible Markup Language* (XML). Tal fator exige que o TIME LINK mantenha uma *thread* sempre a executar e que permita o registo destes pedidos, gerindo-os através do método *First-Come*, *First-Served* (FCFS). O que permite que se adicionem novas tarefas a serem executadas, mesmo que essas tarefas sejam executadas através de classes Java que implementem a interface *Worker*.

Base de Dados

Esta é a componente que armazena toda a informação do sistema e corresponde à camada de modelo do modelo MVC.

O TIME LINK utiliza um servidor de base de dados, MySQL [12], que permite adicionar novos dados e consultar, editar e remover dados existentes. Eles são armazenados em tabelas, seguindo um modelo relacional. Este modelo permite maiores níveis de *performance*, escalabilidade, flexibilidade e também assegura uma forte proteção dos dados existentes.

A estrutura de dados está dividida em três componentes: entidades, relações e atributos. As entidades dizem respeito a todos os tipos de eventos históricos e pessoas ou atos que a eles dizem respeito. As relações, tal como o nome indica, definem relacionamentos entre as entidades e podem ser, por exemplo, eventos, relações de parentesco, entre outros. Os atributos guardam informação sobre as entidades e podem ser armazenados, alterados ou removidos sem que a base de dados necessite de ser reestruturada.

Funcionalidades

O TIME LINK oferece aos seus utilizadores várias funcionalidades. Abaixo estão listadas as principais e/ou aquelas que são mais requisitadas:

- Mecanismos de pesquisa de uma qualquer expressão, como, por exemplo, um nome, uma profissão, uma localidade. Estas pesquisas podem ser restringidas por tipo de entidades e por intervalos de tempo;
- Navegar diretamente pela base de dados, através da página de navegação "Explorar". Nesta página são listados todos os atributos e relações existentes na base de dados;
- Apresentação de relações de qualquer tipo entre entidades, sendo estas relações agrupadas na ficha de pessoa real, isto é, página *web* com os vários acontecimentos na vida da pessoa em questão. Estes acontecimentos estão ordenados cronologicamente e têm definido: a função, os atributos, as relações e o nome sobre o qual a pessoa foi referida;
- Tradução de ficheiros com transcrições de textos anotados, segundo a notação Kleio, para XML. Este mecanismo de tradução produz um relatório de tradução que permite ao utilizador perceber se a tradução foi ou não bem-sucedida;
- Adicionar à base de dados o código XML que resultou da tradução dos ficheiros Kleio. Semelhante à tradução de ficheiros, também é produzido um relatório de importação, dando assim feedback ao utilizador do estado do processo.

3.1.2. Descrição da Notação Kleio

Nesta subsecção serão abordados os elementos base da linguagem Kleio e as especificidades que caracterizam a mesma. Para isso esta subsecção foi dividida em dois tópicos: Conceitos Gerais e Regras Gerais da Notação.

Conceitos Gerais

Tal como já foi referido anteriormente, o sistema TIME LINK permite a importação de ficheiros de transcrição de uma fonte. Estes ficheiros são constituídos por texto, que segue uma notação denominada de Kleio (esta teve origem no *Max-Plank Institute für Geschichte*), e são gravados com a extensão ".cli", como, por exemplo, "ob1688.cli".

A notação que é utilizada (Kleio) segue uma estrutura de modelo de dados que se baseia em três conceitos: grupos, elementos e aspetos. Um grupo é a unidade nuclear de recolha de informação e pode ser, por exemplo, uma pessoa ou propriedade. Este é descrito por elementos, por exemplo, o nome da pessoa ou a área da propriedade. Os aspetos servem para registar informação adicional (valor base, grafia original e comentário) e atribuir valores aos elementos acima mencionados.

Através deste modelo de dados, o TIME LINK considera as fontes como conjuntos de *actos*. Um *acto* é um acontecimento num dado momento e num determinado ponto, como, por exemplo, batismos, casamentos, óbitos, entre outros. A cada *acto* estão associados atributos próprios (data, localização, entre outros). No entanto, a informação mais pertinente dos mesmos é composta pelas pessoas ou entidades que neles participam, por exemplo, propriedade que é vendida. Essas pessoas ou entidades são definidas por um conjunto de funções, atributos e de relações. Isto é, quando estas são referidas numa fonte regista-se: a função desempenhada no *acto* os atributos a si associados e as relações com outras pessoas não deduzíveis da sua função.

Regras Gerais da Notação

Intrinsecamente ligado a este tipo de notação de documentos históricos estão regras. Abaixo encontram-se listadas algumas dessas regras, seguidas de um exemplo simples:

- Todas as transições começam com a declaração geral Kleio e logo de seguida pelo grupo fonte;
- Todo e qualquer grupo é sempre seguido por um "$", isto é, a palavra imediatamente anterior ao "$" corresponde a um grupo;
- Cada grupo tem elementos obrigatórios e outros opcionais, no entanto, todos os elementos desse grupo são separados por "/";
- Estes elementos têm também uma ordem pré-definida, essa ordem pode diferir consoante o tipo de grupo;
- Cada pessoa é registada como um novo grupo. Os atributos relativos a essas pessoas são notados com a fórmula "ls$", que vem de *life story*, e podem ser, por exemplo, morada, residência, profissão, entre outros;
- Sempre que existe um sinal "--" sucedido de um texto, é considerado como um comentário introduzido pelo utilizador;
- É o sinal "%" que introduz a linguagem na forma original, isto é, a linguagem que estava presente na fonte original;
- Comos os sinais "$", "/", "--" e "%" são regras específicas da notação, caso se queria introduzir esses sinais como parte do texto é estritamente necessário escrevê-los imediatamente a seguir a "\".

Exemplo:

```
kleio$gacto2.str
fonte$obitos 1577-1736/localizacao-A.U.C./loc=igreja paroquial da lousa

    obito$o1714-1/30/1/1714/fl.40v./manuel lopes serra (prior)
      n$manuel quaresma/m/id=o1714-1-per1
        ls$morada/vila da lousa%lousa vila
      sacr$sim
      testamento$nao
      locs$nao refere

    obito$o1714-2/5/2/1714/fl.41/manuel lopes serra (prior)
      n$jose freire/m/id=o1714-2-per1
        ls$morada/ponte do areal
        ls$idade/0.25#3 ou 4 meses
      sacr$sim
      testamento$nao
      locs$nao refere
```

Análise do Exemplo:

A primeira linha é de carácter obrigatório. Esta identifica a linguagem de notação especial e faz referência ao ficheiro que contém os actos da mesma.

De seguida é especificada a fonte. Neste caso são os óbitos entre 1577 e 1736 na localização A.U.C. e cuja missa tenha sido realizada na Igreja Paroquial da Lousã.

Nas restantes linhas é especificado cada óbito. Tomando, como exemplo, a seguinte entrada:

```
obito$o1714-1/30/1/1714/fl.40v./manuel lopes serra (prior)
        n$manuel quaresma/m/id=o1714-1-per1
          ls$morada/vila da lousa%lousa vila
        sacr$sim
        testamento$nao
        locs$nao refere
```

Pode-se verificar que a data do óbito é 30 de Janeiro de 1714 e que o prior foi Manuel Lopes Serra. Posteriormente, é identificado o falecido, nomeadamente o seu nome, sexo, morada e *IDentifier* (ID). Depois são adicionadas outras informações, como, por exemplo, se possui testamento ou não.

3.1.3. Necessidades Encontradas

Nesta subsecção serão identificadas as potenciais necessidades do TIME LINK e como estas influenciaram o propósito do estágio e a definição dos objetivos do mesmo.

Potenciais Melhorias vs Mais-valias do Estágio

O processamento de documentos históricos transcritos sobre a forma de textos anotados, isto é, que seguem uma notação especial (Kleio), é um dos principais componentes que fazem parte do TIME LINK. No início deste estágio, o processo de introdução dos textos anotados era feito através de um simples editor de texto (jEdit [13]) e tinha uma curva de aprendizagem muito alta, facilitando a geração de erros, dificultando a adoção do programa e tornando tudo isto num processo cansativo.

As premissas anteriores serviram para identificar as potenciais melhorias e perceber as mais-valias que este estágio poderia acrescentar ao sistema já existente. Tendo em conta tudo isso, foram pensadas soluções para diminuir a tal curva de aprendizagem, facilitar assim a inserção dos textos anotados e acelerar todo o processo inerente à inserção.

Para ir de encontro às necessidades encontradas, foi pensado incorporar no sistema uma ferramenta baseada em interfaces web, que doravante vamos chamar de editor de código, que produza *feedback* ao utilizador e com um elevado grau de interatividade (automatismos, realce da sintaxe, indentação, entre outros). De referir que as características deste editor serão descritas mais à frente, mais concretamente na subsecção 4.1.2., onde são especificados os requisitos.

3.2. Estudo das Aplicações Concorrentes

Nesta secção são detalhadas algumas aplicações concorrentes à solução que se pretende implementar e que, como tal, poderão servir de exemplo. Este estudo encontra-se dividido em três subsecções que reportam o estudo de alguns dos editores de código web-based existentes, a análise comparativa entre eles e as ideias que foram retiradas deste estudo.

3.2.1. IDEs *Web-based*

Nesta subsecção são apresentados, individualmente, alguns *Integrated Development Environment* (IDE) web-based que foram estudados. De referir que os escolhidos não são os únicos existentes no mercado mas são os mais populares.

ShiftEdit [14]

O ShifEdit é um IDE *web-based* que oferece um vasto leque de funcionalidades aos seus utilizadores. Entre as várias funcionalidades destacam-se as seguintes: criar e editar código PHP: Hypertext Preprocessor (PHP), Ruby, Python, Perl, Java, HTML, CSS e JavaScript; *autocomplete* para CSS, HTML e PHP; análise em tempo real de sintaxe; acesso a ficheiros através de *File Transfer Protocol* (FTP), *Secure File Transfer Protocol* (SFTP), Dropbox, Google Drive, entre outros. A sua versão básica é gratuita mas para ter acesso a todas as funcionalidades que são oferecidas é preciso pagar uma mensalidade. Esta mensalidade é das mais baratas do mercado.

Codenvy [15]

Este IDE *web-based* foi pensado para programadores de aplicações *server-side*. São oferecidos ambientes pré-definidos de linguagens de programação como Java, JavaScript, PHP, Android, Python, Ruby, C++, entre outros. Por outro lado, os utilizadores podem criar um ambiente personalizável usando objetos *JavaScript Object Notation* (JSON) e Dockerfiles. Outro fator determinante é a integração com o GitHub e suportar os comandos associados a este. Tal como o ShifEdit, a versão básica do Codenvy é livre de encargos mas nem todas as funcionalidades estão disponíveis, para isso terá que ser paga uma mensalidade, esta bem mais cara que a do serviço mencionado anteriormente.

Cloud9 [16]

Tal como os IDEs anteriormente falados, o Cloud9 disponibiliza um serviço básico gratuito e um serviço pago (mensalmente) que oferece ao utilizador todas as funcionalidades existentes. Entre elas destacam-se: acesso a um terminal que executa comandos de compilação, sugestões à medida que o código é escrito, acesso a ferramentas de *debug* para inspecionar o código, personalizar a página (por exemplo, vista separada ou temas), múltiplas execuções de código e edição de imagens.

Codeanywhere [17]

Codeanywhere é uma plataforma que permite aos utilizadores ter acesso aos seus ficheiros e poder editá-los da maneira mais fácil possível, através de diversos dispositivos: *smartphone*, *Personal Computer* (PC) ou *tablet*. Esta aplicação oferece um cliente FTP e suporte às principais linguagens de programação (HTML, PHP, JavaScript, CSS, XML, entre outras). Outras funcionalidades também são disponibilizadas como: facilidade em personalizar o IDE, sugestões à medida que se programa, suporte a todos os navegadores, múltiplos *layouts* (em grelha ou vista separada), guardar alterações aos ficheiros automaticamente, associar a conta a outros clientes (GitHub, Google Drive, Dropbox, entre outros), uso de ferramentas de compilação e de *debug* e possibilidade de consultar o histórico de revisões para poder comparar as alterações, bem como para poder reverter para uma versão anterior.

3.2.2. Análise Comparativa

É apresentada de seguida, na tabela 5, a comparação das aplicações concorrentes estudadas. De notar que as características a ter em consideração passaram por ir de encontro com o que a solução visa incorporar no sistema: um editor de programação baseado em interfaces *web*, as ideias retiradas têm como base o estudo efetuado acerca dos IDEs *web-based*.

Tabela 5: Análise comparativa das aplicações concorrentes estudadas.

	ShiftEdit	*Codenvy*	*Cloud9*	*Codeanywhere*
Suporte a linguagens de programação (criação e edição de código)	Sim	Sim	Sim	Sim
Edição de ficheiros (carregados localmente)	Sim	Sim	Sim	Sim
Automatismos (indentação, highlighting, autocomplete, sugestões)	Sim	Sim	Sim	Sim
Compilação e debug	Não	Sim	Sim	Não
Acesso a linha de comandos	Não	Não	Sim	Não
Suporte a diferentes navegadores	Sim	Sim	Sim	Sim
Tipo de serviço	*Freemium*	*Freemium*	*Freemium*	*Freemium*

3.2.3. Ideias Retiradas

Nesta subsecção será feito um apanhado das ideias que foram retiradas do estudo das aplicações concorrentes e que se encaixam na solução que se pretende implementar.

Primeiro que tudo, importa referir que as funcionalidades que foram tidas em conta visam as características que foram definidas para a solução a implementar. Estas características são: produção de *feedback* ao utilizador e elevado grau de interatividade.

Depois de concluído o estudo das aplicações concorrentes foram identificadas as funcionalidades que vão de encontro às características referidas anteriormente e que possam ser uma mais-valia para a solução:

- Possibilidade de editar código;
- Suporte à notação Kleio;
- Automatismos:
    - Completar código;
    - Indentação;
    - Realce da sintaxe;
- Teclas de atalho para executar operações básicas, tais como:
    - Copiar, colar e cortar;
    - Anular e refazer;
    - Entre outras;
- Suporte aos navegadores mais comuns.

3.3. Soluções Encontradas

Nesta secção são apresentadas as soluções que foram encontradas e adotadas para a execução deste projeto e que vão de encontro às necessidades encontradas na subsecção 3.1.3. e às ideias retiradas do estudo das aplicações concorrentes (subsecção 3.2.3.). Foi também tido em conta que se tratavam de soluções *open source*.

Tendo em conta que implementar uma solução com todas estas características seria um processo extenso e que, provavelmente, não seria exequível num ano letivo, procurou-se fazer um levantamento de *frameworks* para simplificar este processo. A *framework* que mais se aproximou do pretendido foi o Firepad [18], que está dependente de outras duas *frameworks* (Ace [19] e Firebase [20]) para que as soluções disponíveis funcionem na sua plenitude. De seguida são detalhadas as *frameworks* referidas.

A escolha destas ferramentas prendeu-se também com o facto de estas serem produtos com elevada maturação e com uma comunidade à sua volta pró-ativa. Só para se ter uma ideia, a *framework* Ace é um produto tão confiável que é usado em projetos bastante complexos como, por exemplo, o GitHub ou o ShareLaTeX [21].

Ace

A *framework* Ace foi desenvolvida em Setembro de 2010, é o sucessor do projeto Mozilla Skywriter, e o licenciamento desta assenta na licença *Berkeley Software Distribution* (BSD) [22]. As suas principais são:

- Realce da sintaxe para mais inúmeras linguagens;
- Indentação automática;
- Suporte a documentos quem contêm muita informação;
- Combinação de teclas;
- Agrupamento de informação;
- Entre outras.

Firebase

Esta ferramenta surgiu em Setembro de 2011, é detida pela Google e o licenciamento desta assenta sobre a licença Apache 2.0 [23]. Trata-se de uma plataforma de desenvolvimento de aplicações e tem como principais características:

- Armazenamento de dados;
- Sincronização de dados;
- Mecanismos de autenticação;
- Hospedagem de aplicações *web*;
- Suporte à consulta dos dados armazenados em modo *offline*;
- Entre outras.

Firepad

Esta ferramenta surgiu em Setembro de 2014, é detida pela Google e o licenciamento desta assenta sobre a licença Massachusetts Institute of Technology (MIT) [24]. Este é um projeto *open source* desenvolvido pelo Firebase. Como o Firepad está dependente das *frameworks* Ace e Firebase, este acaba por herdar as características destas. No entanto a combinação destas *frameworks* faz com que outras características surjam:

- Edição colaborativa;
- Deteção da presença de outro utilizador;
- Entre outras.

Capítulo 4
Especificação

Este capítulo detalha algumas das fases que foram realizadas ao longo do estágio. Numa primeira secção será detalhado todo o processo de levantamento e especificação de requisitos. Nas secções 2 e 3 serão apresentados, respetivamente, os diagramas de casos de uso e as *mockups*. Na quarta secção será ilustrada e descrita a arquitetura do sistema. Na quinta e última secção será representada a arquitetura de dados e especificadas as suas particularidades.

4.1. Requisitos

Nesta secção será detalhado o processo de levantamento de requisitos e especificados os requisitos que foram identificados e validados durante todo o processo.

4.1.1. Levantamento e Validação

O levantamento de requisitos é parte fundamental do projeto, pois é nesta fase que são identificadas todas as condições que o sistema deverá satisfazer. Para isso foram utlizadas várias técnicas, nomeadamente: entrevista e elaboração de reuniões participativas com os *stakeholders*.

Na entrevista com os *stakeholders* foram apresentadas, de uma maneira geral, as principais ideias para a aplicação. Nas reuniões participativas foram identificados, discutidos e validados todos os requisitos, desde os gerais aos mais específicos.

Apesar dos requisitos terem sido definidos e validados numa fase inicial do estágio, estes sofreram algumas alterações numa fase já adiantada. Tais alterações foram realizadas para irem ao encontro de algumas necessidades que surgiram ao longo da segunda metade do estágio.

4.1.2. Especificação

Os requisitos foram divididos em funcionais, relacionados com as funcionalidades da aplicação, e não-funcionais, que dominam a qualidade da aplicação. Foi-lhes atribuído um nível de prioridade, que pode ser baixa, média ou alta. Quando um requisito tem prioridade alta significa que a não realização do mesmo compromete o resultado final da aplicação. Já os requisitos de prioridade média quando falham não comprometem o resultado final da aplicação e, quando são concretizados, acrescentam significantes valias à aplicação. Os requisitos de prioridade mais baixa são aqueles que, caso falhem, não comprometem todo o trabalho realizado.

Requisitos Funcionais

Na tabela 6 são detalhados os requisitos que definem as ações fundamentais da aplicação.

Tabela 6: Requisitos Funcionais

ID	Nome	Descrição	Prioridade
RF.1	Editor de texto	A aplicação deve disponibilizar ao utilizador a possibilidade de escrever código Kleio que posteriormente poderá ser gravado para ficheiro, traduzido e adicionado à base de dados.	**Alta**
RF.2	Editar ficheiros	O utilizador poderá editar ficheiros que se tenham sido carregados por si.	**Média**
RF.3	Gravar ficheiros	Deve ser permitido ao utilizador gravar o conteúdo da caixa de texto, num ficheiro já existente ou num novo ficheiro.	**Média**
RF.4	*Autocomplete*	À medida que o utilizador vai escrevendo o código Kleio no editor deverá ser possível completar o código que está a ser escrito.	**Alta**
RF.5	Realce da sintaxe	Todas as *keywords* específicas da notação Kleio deverão ser realçadas quando o texto é inserido no editor.	**Alta**
RF.6	Sugestões	À medida que o utilizador vai escrevendo o código Kleio no editor deverá ser possível sugerir ao utilizador quais os parâmetros obrigatórios para um grupo que faça parte da notação Kleio (p.ex. caso o utilizador queira adicionar informação biográfica acerca de um batismo, deverá ser indicado que os parâmetros obrigatórios são: id, dia, mês, ano).	**Baixa**
RF.7	Edição colaborativa	A aplicação deve suportar a edição de código colaborativa, isto é, vários utilizadores em máquinas diferentes deverão poder editar o mesmo ficheiro.	**Baixa**

Requisitos Não-Funcionais

Na tabela 7 são detalhados os requisitos que são fatores determinantes para a qualidade da aplicação, isto é, que definem capacidades ou condições que o sistema deve cumprir.

Tabela 7: Requisitos não-funcionais

ID	Nome	Descrição	Prioridade
RNF.1	Portabilidade	A aplicação *web* não depende do sistema operativo e é compatível com os navegadores mais recentes, a partir das seguintes versões: Google Chrome 42 ou superior, Mozilla Firefox 38 ou superior e Safari 7 ou superior.	**Baixa**
RNF.2	Usabilidade	Será tido em conta algumas regras que tornem a aplicação mais *user-friendly*.	**Média**

4.2. Diagrama de Casos de Uso

Nesta secção é apresentado o diagrama de casos de uso. Serve este diagrama para descrever o comportamento do sistema na perspetiva do utilizador, isto é, serve para mostrar quais as funcionalidades do sistema.

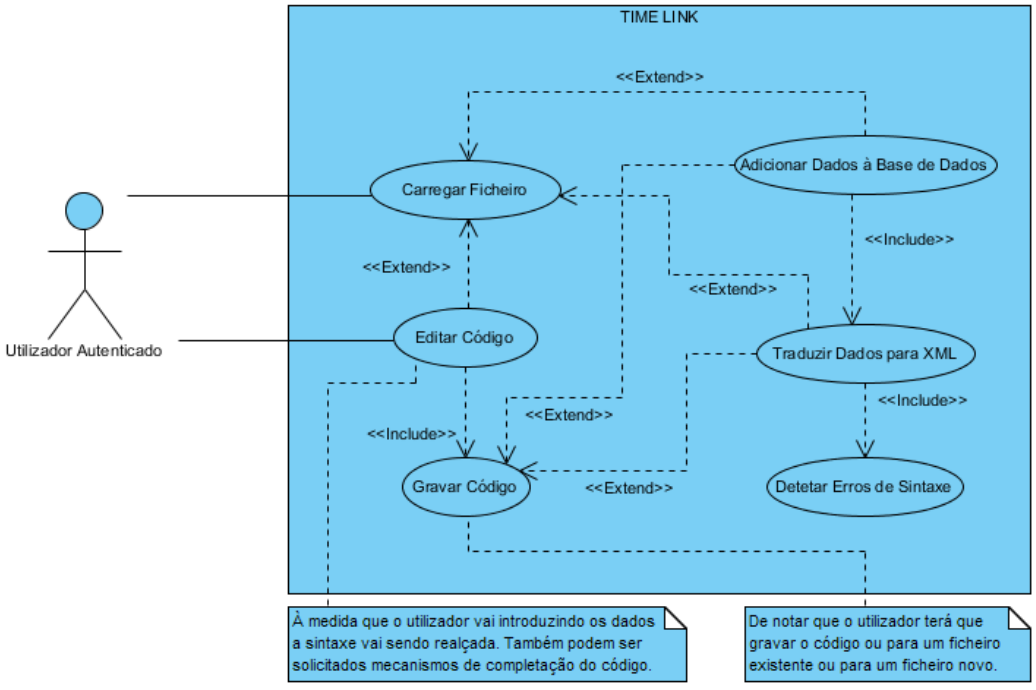

Figura 2: Diagrama de casos de uso

Depois de analisar a figura 2, identificamos, inequivocamente, duas ações principais: "Editar Código", "Carregar Ficheiro". Só a partir dessas ações é que são desencadeadas ações secundárias como: "Gravar Código", "Traduzir Dados para XML" e "Adicionar Dados à Base de Dados". Essas ações secundárias nunca poderão ser realizadas sem que as primeiras ações as desencadeiem, isto é, por exemplo, nunca podemos traduzir os dados para XML sem que o código tenha sido gravado para um ficheiro novo ou existente ou tenha sido carregado um ficheiro.

A ação "Detetar Erros de Sintaxe" é uma ação obrigatória, ou seja, é sempre desencadeada quando a ação "Traduzir Dados para XML" é executada, pois não será possível traduzir os dados Kleio para XML sem fazer a deteção de erros de sintaxe, para certificarmo-nos que os dados são adicionados sem erros.

A ação "Adicionar Dados à Base de Dados" também necessita que os dados Kleio tenham sido traduzidos para XML com sucesso, pois só assim é que poderão ser adicionados à base de dados.

4.3. *Mockups*

Nesta secção é apresentada uma das várias *mockups* desenhadas ao longo do estágio. As restantes *mockups* poderão ser consultadas no anexo A.

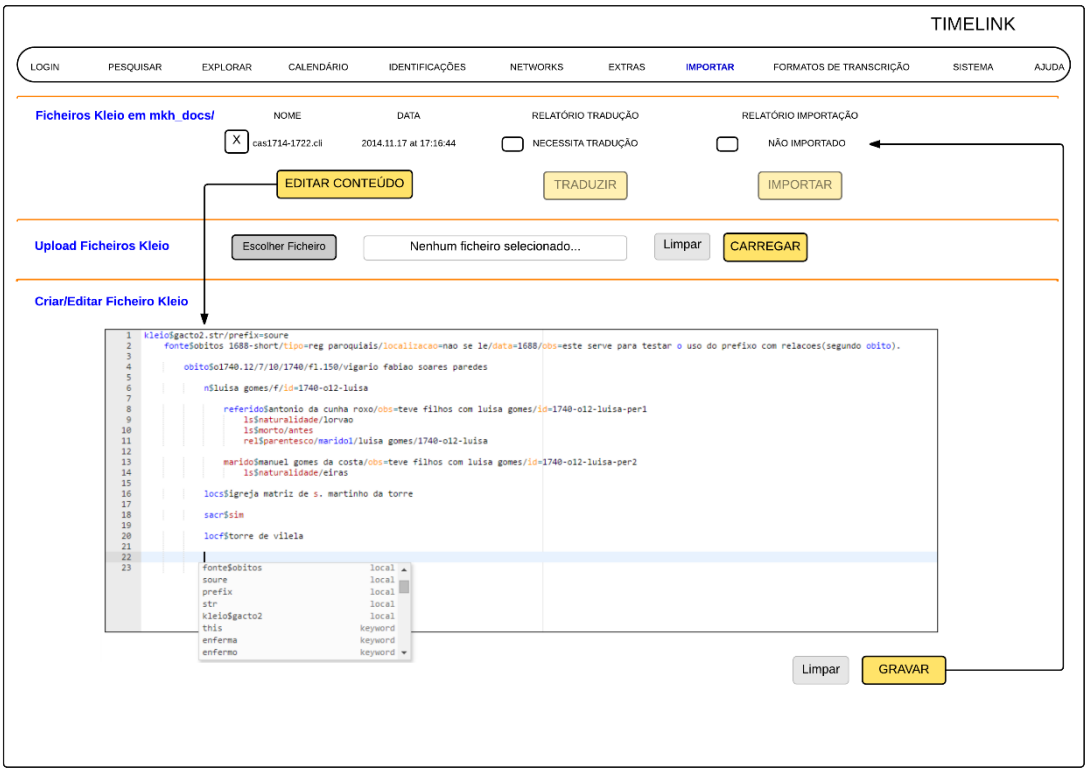

Figura 3: Mockup para editar um ficheiro

Na *mockup* da figura 3 o utilizador encontra-se na página de navegação "Importar". Neste caso o utilizador está a editar um documento que está guardado na pasta "mhk_docs" e à medida que vai editando os dados a sintaxe vai sendo realçada e caso pretenda poderá invocar mecanismos de completação de código. Quando vai gravar o ficheiro a data de modificação vai ser atualizada e caso queria adicionar os dados à base de dados terá que fazer primeiro a tradução e, caso seja bem-sucedido, a posterior importação.

4.4. Arquitectura do Sistema

Nesta secção é representada a proposta de resolução do problema, em concordância com os requisitos enunciados na secção 4.1.2.. Na primeira subsecção é especificado o modelo arquitetural do sistema. Na subsecção seguinte é detalhada a arquitetura do sistema caso este utilize a funcionalidade de edição colaborativa.

4.4.1. Modelo Arquitetural Final

Nesta secção é representada a arquitetura do sistema que será implementado e é descrita a interação entre todos os componentes que fazem parte desse sistema.

Tal como podemos constatar na figura 4, o cliente pode interagir com a *framework* Ace e o servidor do TIME LINK, realizando variadíssimas tarefas. A interação com estes componentes é feita mediante a(s) funcionalidade(s) que for(em) solicitada(s).

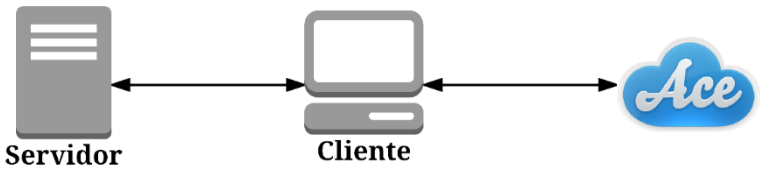

Figura 4: Arquitetura da aplicação

Caso o cliente solicite funcionalidades que necessitem de comunicar com o servidor serão usados os comandos GET ou POST, consoante a funcionalidade pretendida. Esta comunicação é feita através de pedidos AJAX (*Asynchronous* Javascript *and* XML) que permitem contactar o servidor assincronamente e receber respostas do mesmo. As funcionalidades que necessitam de comunicar com o servidor dizem respeito a operações de manipulações de ficheiros, tais como editar ou gravar um dado ficheiro.

Quando as funcionalidades executadas pelo cliente impulsionam o Ace quer dizer que estamos perante tarefas de edição de código, isto é, realce da sintaxe, indentação, entre outros. É o Ace que é responsável pela parte visível da aplicação, ou seja, é ele que disponibiliza a informação desejada ao cliente. A grande mais-valia do Ace é tratar-se de uma *framework client-side*, o que permitiu facilitar a integração do mesmo na aplicação e manter o foco na edição de código, isto é, no tratamento da informação que é mostrada ao cliente tendo em conta todas as especificidades da notação Kleio.

As funcionalidades de edição de código que são solicitadas, através de um navegador comum, instanciam o Ace. Na figura 5 é demonstrada a lógica interna de negócio do mesmo.

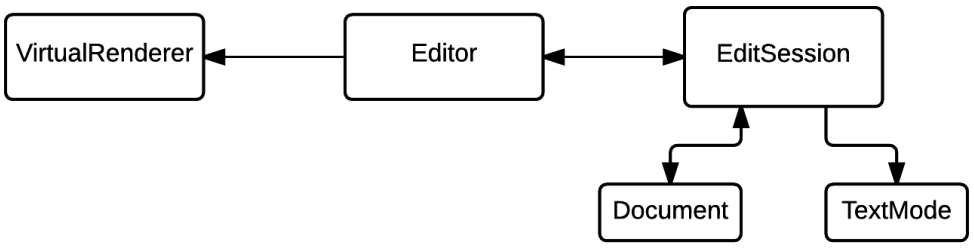

Figura 5: Lógica de negócio do Ace

Editor: Esta classe é o principal ponto de entrada para as funcionalidades inerentes ao Ace e é acedida quando o Ace é instanciado. É esta classe que administra a classe EditSession, bem como a classe VirtualRenderer. Todos os eventos do utilizador que utilizem o teclado ou rato são tratados nesta classe.

EditSession: Nesta classe são armazenados todos os estados relacionados com um certo documento (corpo do texto que se encontra no editor). É esta classe que mantém informada a classe Editor dos seguintes eventos:

- Seleção de texto – quando o texto selecionado passa a ser outro é acionado o evento "changeSelection";
- Posição do cursor – se a posição do cursor for alterada é emitido o evento "changeCursor";
- Posição do *scroll* – os eventos "changeScrollLeft" e "changeScrollTop" são, respetivamente, acionados quando o *scroll* esquerdo e o *scroll* de cima são alterados;
- Ações de reversão – caso sejam executadas tarefas de desfazer ou refazer são invocados os métodos "undo()" e "redo()", respetivamente;
- Modo como interpreta a linguagem – através da classe TextMode.

TextMode: Nesta classe são especificados os detalhes que caracterizam a linguagem que é utilizada. Esses detalhes são: regras para realce da sintaxe, regras de indentação e regras de agrupamento de instruções.

A abordagem que o Ace utiliza para realçar a sintaxe assemelha-se a uma máquina de estados, isto é, o *highlighter* começa sempre por um estado inicial e vai transitando de estado em estado, procurando a expressão regular correspondente. Os estados que fazem parte do *highlighter* são definidos através de: nome do estado, *tokens*, expressões regulares e, opcionalmente, por estados transitórios.

No que toca às regas de indentação o Ace aplica-as da mesma forma que os editores mais comuns (p.ex. Eclipse [25]), ou seja, verifica se se está a fechar um agrupamento de instruções e caso se confirme fecha a instrução e recua algumas colunas, caso contrário mantém a indentação da linha anterior.

Quanto às regras de agrupamento de instruções (ou *code folding*) a abordagem é praticamente a que se verifica nas regras de realce da sintaxe, os pontos inicias e finais de agrupamento são definidos através de expressões regulares.

Document: Esta classe contém todo o texto que faz parte do documento. Basicamente, toda a informação é guardada num *array* de *strings*, correspondendo cada linha do documento ao índice no *array*. Cada vez que o documento sofre modificações é acionado o evento "change" e a classe EditSession é notificada do ocorrente.

Cada *array* de *strings* contém o texto presente no documento e é constituído por vários *tokens*. Para que cada *token* seja identificado é utilizado um *tokenizer*. Caso uma linha seja modificada é necessário reindexar toda a informação abaixo da mesma, para isso é executado em segundo plano outro *tokenizer* (*background tokenizer*).

VirtualRenderer: Esta classe é responsável por desenhar tudo o que é apresentado no editor do Ace, tais como: *layers*, texto, cursores, entre outros. Importante referir que sempre que há alterações no corpo de texto presente no editor esta classe é chamada para redesenhar tudo.

4.4.2. Modelo Arquitetural com Edição Colaborativa

Sentiu-se a necessidade de dividir o modelo arquitetural em duas secções, pois numa fase inicial a funcionalidade de edição colaborativa foi considerada e adicionada à solução. No entanto, esta não foi integrada na aplicação final porque adicionava algumas limitações ao TIME LINK, nomeadamente gestão de ficheiros, necessidade de conexão à Internet e necessidade de criar uma conta Firebase. No entanto, é especificada nesta subsecção a arquitetura do sistema caso a edição colaborativa fizesse parte deste.

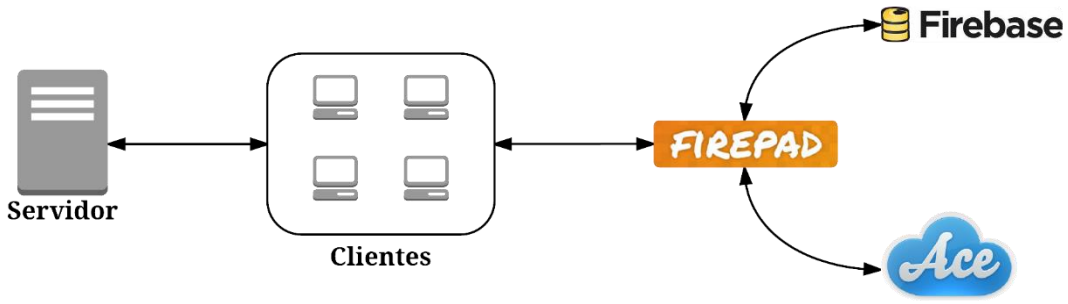

Figura 6: Arquitetura da solução com edição colaborativa

Como podemos constatar na figura 6, existe um conjunto de clientes (um ou mais) que pode interagir com a aplicação e realizar variadíssimas tarefas, desde que acedam via Internet e utilizem um navegador comum. A solicitação dos componentes (servidor e/ou Firepad) é feita mediante a(s) funcionalidade(s) que é/são solicitada(s).

A comunicação entre o(s) cliente(s) e o servidor segue o mesmo modelo arquitetural especificado na secção 4.4.1.. Caso este(s) solicite(m) funcionalidades que necessitem de comunicar com o servidor serão usados os comandos GET ou POST, consoante a funcionalidade executada.

A principal diferença entre este modelo arquitetural e o descrito na secção 4.4.1. é que quando as funcionalidades solicitadas pelo(s) cliente(s) dizem respeito a tarefas de edição de código e/ou de edição colaborativa estas impulsionam a *framework* Firepad. Contrariamente ao que acontece no modelo presente na secção 4.4.1., em que a *framework* impulsionada é a Ace.

O Firepad é o responsável pela parte visível da aplicação, ou seja, é este que disponibiliza ao(s) clientes(s) a informação desejada. Para que essa informação vá de encontro ao que é pretendido o Firepad faz uso das *frameworks* Ace e Firebase por exemplo, para realçar a sintaxe e/ou carregar a última sessão, respetivamente. Isto é, ele comporta-se como um mediador entre as *frameworks* referidas anteriormente, utilizando funcionalidades de cada uma para oferecer um editor de código altamente eficaz e que suporte a edição colaborativa. Estas valias que foram determinantes na escolha do Firepad para integrar a aplicação pois é possível ter um editor de código altamente configurável, ou seja, usufruir de todas as funcionalidades da *framework* Ace, e que vários clientes possam editar o mesmo código e detetar a presença de outros clientes, funcionalidade ligadas à integração com a *framework* Firebase, tudo isto em tempo real. Como, por exemplo, se verifica no Google Docs [26].

Tomando como exemplo uma aplicação real do sistema TIME LINK, em que uma equipa de investigadores precisa de introduzir diversos dados biográficos num só ficheiro e esse trabalho não é de uma só pessoa, com as características que o Firepad oferece é possível que diferentes pessoas do mesmo grupo de trabalho editem o mesmo ficheiro simultaneamente, o que aumenta a produtividade, reduz o custo e acaba com a limitação de um dado ficheiro apenas ser acessível a uma única pessoa.

É a *framework* Firebase que monitoriza o Firepad, o que permitiu adicionar à aplicação um *back-end* com bastante credibilidade e robustez. Sendo assim, a aplicação também herdou todas as funcionalidades que a *framework* Firebase por norma contém: controlos de segurança de excelência, fácil acessibilidade de dados e atualizações de baixa latência. O Firepad também armazena toda a informação acerca dos utilizadores (ID, cursor, cor do cursor) e histórico de revisões (ID revisão, ID utilizador, lista de operações efetuadas) no *Uniform Resource Locator* (URL) do Firebase que é definido. Estes dados estão disponíveis para consulta e manipulação, para isso basta introduzir num navegador o URL que foi referenciado anteriormente.

Quanto às funcionalidades de edição de código, estas são geridas pela *framework* Ace e seguem a mesma lógica de negócio que a descrita na secção 4.4.1.

Por fim, é importante referir novamente que esta funcionalidade não está presente no produto final, pois seriam adicionadas algumas limitações à solução e que comprometiam o resultado final da mesma, a saber:

- Necessidade de ligação à Internet, pois o Firepad quando é instanciado é necessário especificar o URL Firebase para guardar os dados.
- Deficiente gestão na edição de diferentes ficheiros por parte de utilizadores distintos, quando estes se encontrassem a utilizar o TIME LINK na mesma altura;

Capítulo 5
Desenvolvimento

Neste capítulo são dadas a conhecer as decisões que foram tomadas para desenvolver os diferentes requisitos e na última secção são descritos os testes que foram realizados.

5.1. Pedidos ao Servidor

Nesta secção é apresentada a abordagem aplicada na gestão dos pedidos que são feitos ao servidor. Como foi referenciado na subsecção 3.1.2., todos os pedidos efetuados são geridos pelo servidor.

Os principais objetivos foram: enviar/receber dados sem que seja visível ao utilizador, efetuar pedidos sem ter que recarregar a página por inteiro (atualizando apenas o que realmente interessa). Para ir de encontro a estas características foi utilizada a técnica de programação AJAX que utiliza o JavaScript para fazer pedidos assíncronos entre o navegador e as componentes (pedidos HTTP). A classe em JavaScript adotada foi a XMLHttpRequest [27] que cria um objeto (instância) da mesma e possibilita trocar dados com o servidor. Esta classe segue os objetivos que foram definidos anteriormente e ainda tem a mais-valia de suportar dados no formato de texto, JSON, XML, entre outros.

Para executar as operações solicitadas pelos pedidos HTTP existem dois métodos, que fazem parte do protocolo HTTP, que são utilizados:

- GET – requisita informação ao servidor, como, por exemplo, para procurar dados acerca de uma dada pessoa;
- POST – submete informação que vai ser processada pelo servidor, como, por exemplo, para gravar dados num ficheiro.

Quando são processadas as respostas vindas do servidor acerca dos pedidos efetuados é necessário verificar duas situações: qual o estado do pedido efetuado e qual o estado da resposta HTTP do servidor. Abaixo são listados os valores a ter em conta quando é verificado o estado do pedido efetuado.

- 0 – não inicializado;
- 1 – a carregar;
- 2 – recebido;
- 3 – em processamento;
- 4 – concluído e resposta enviada.

A resposta que é retornada pelo servidor tem um código associado. De seguida é apresentada a listagem dos vários tipos de resposta possível:

- 200 OK – o pedido foi efetuado com sucesso. A informação que é retornada (na resposta) depende do método que é solicitado: se for GET, é devolvida uma entidade que corresponde aos dados que foram requisitados; se for POST, é devolvida uma entidade que contém o resultado da solicitação;
- 404 Not Found – não foi encontra nada que corresponda ao Uniform Resource Identifier (URI) do pedido;

- 500 Internal Server Error – o servidor encontrou uma condição inesperada que o impediu de cumprir o pedido.

Como se pode constatar, existem vários valores associados ao estado de um pedido e diversos tipos de resposta possíveis de serem devolvidos consoante esse mesmo pedido. No entanto, para o estado do pedido verificou-se apenas se é devolvido o valor 4 e para o tipo de resposta retornada simplesmente foi verificado se o código associado a esta é 200 OK.

5.2. Funcionalidades

Na presente secção é explicado todo o trabalho desenvolvido para implementar as funcionalidades que estão integradas na aplicação.

5.2.1. Editor de Texto

Para que o utilizador tenha a possibilidade de editar código e usufruir de todas as funcionalidades, foi necessário definir na página *web* uma secção, através da *tag* HTML " ". É nesta secção que vai ser incorporado o editor Ace e este é instanciado sempre que a página *web* é acedida, para isso é necessário chamar o método edit() da classe Ace, especificar a secção anteriormente citada e associá-lo a uma variável JavaScript (p.ex. *var editor*). Esta associação permite manipulações ao editor como, por exemplo, definir o texto que é apresentado.

Para que o editor dê suporte à notação Kleio é preciso definir o modo no qual este vai trabalhar, através do método *editor.getSession()setMode("ace/mode/kleio")*. O editor vai carregar o ficheiro "mode-kleio.js" e associá-lo à sessão atual. É este ficheiro que contém as especificidades da notação Kleio tais como: palavras-chave, regras de realce da sintaxe e regras de indentação.

5.2.2. Editar Ficheiros

Quando o utilizador necessita de editar um ficheiro tem que seleccionar o mesmo e carregar no botão "Edit". Este vai ser redireccionado para a página onde está o editor implementado e só depois é que poderá editar o conteúdo. Para isso é feito um pedido ao servidor para que este envie o conteúdo do ficheiro que se quer editar. A resposta a este pedido é enviada e é convertida em texto e esse texto colocado no editor para ser manipulado pelo utilizador.

5.2.3. Gravar Ficheiros

Quando o utilizador quer gravar o código que editou tem que pressionar o botão "Gravar". É feito então um pedido ao servidor em que são enviados os campos nome e caminho do ficheiro, bem como todo o texto que foi editado.

5.2.4. *Autocomplete*

Este requisito oferece ao utilizador a possibilidade de encontrar de forma rápida e eficaz uma palavra-chave e de completar a palavra que está a escrever sem ser necessário escrevê-la por inteiro. Esta funcionalidade pode ser visualizada na figura 7.

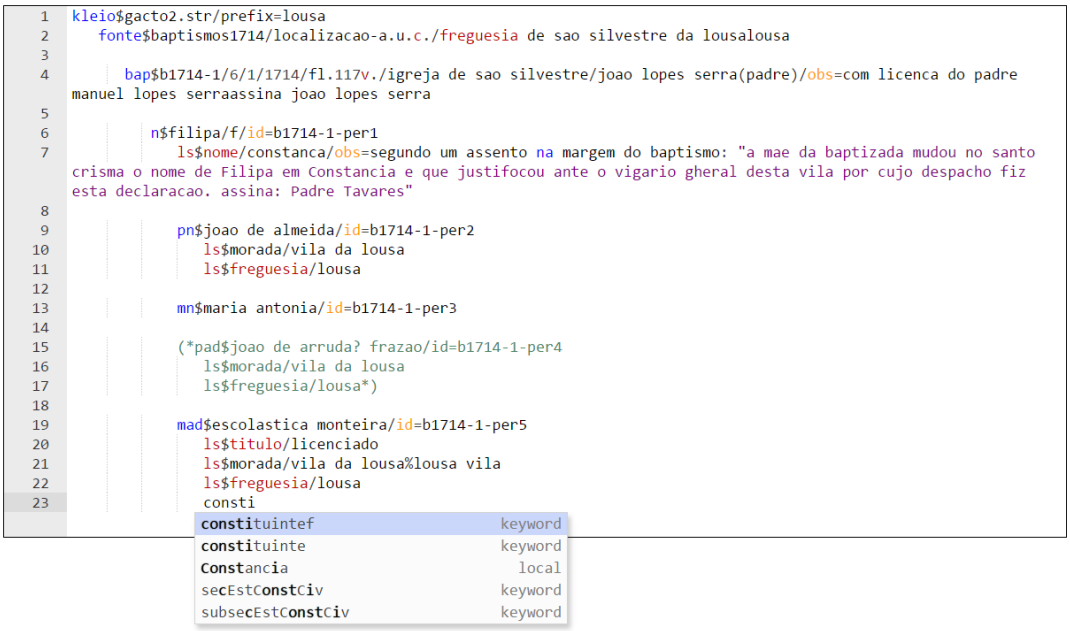

Figura 7: Exemplo de utilização do editor

Como é percetível na figura 7, o utilizador introduz um conjunto de caracteres e o editor abre uma janela (*pop-up*) com a listagem das *keywords* que são sugeridas para completar a palavra que o utilizador quer escrever. Estas *keywords* podem ser locais, isto é, que se encontrem definidas no texto, ou podem ser palavras-chave da notação Kleio, estas foram previamente definidas no ficheiro "mode-kleio.js". Para que as *keywords* sejam listadas estas têm que ter pelo menos todos os caracteres que estão a ser escritos e são ordenadas descendentemente consoante a sua pontuação. Esta pontuação é calculada através da comparação com o conjunto de caracteres escritos pelo utilizador, tendo em conta os índices de cada caracter correspondente e a distância dos mesmos ao conjunto de caracteres introduzido pelo utilizador.

Importante referir que a definição das *keywords* no ficheiro "mode-kleio.js" obedeceu ao módulo de sintaxe "mhk.xml", que já se encontrava definido antes do início deste estágio. Neste módulo encontram-se todas as *keywords* da notação Kleio e são divididas em 3 conjuntos distintos (grupos, elementos e relações).

5.2.5. Realce da Sintaxe

Um dos requisitos mais importantes do estágio é o realce da sintaxe porque permite ao utilizador perceber visualmente quais as *keywords* que fazem parte da linguagem e assim familiarizar-se com a mesma, agilizando e acelerando todo o processo de introdução de dados.

Como se pode constatar na figura 7, pertencente à subsecção 5.2.4., existem várias partes da informação presente no editor que se encontram realçadas. Para isso foram definidas regras de realce da sintaxe no ficheiro "mode-kelio.js", obedecendo ao módulo de sintaxe "mhk.xml", referido na subsecção anterior. Na tabela seguinte são listadas, descritas e exemplificadas todas as regras que definem o realce da sintaxe da linguagem:

Tabela 8: Regras de realce da sintaxe

Regra	Descrição	Exemplo
Comentários de uma linha	É desencadeada pela sequência de caracteres "--" e serve para comentar uma única linha.	--linha comentada
Comentários de múltiplas linhas	Quando o utilizador insere a sequência de caracteres "(*" toda a informação que se segue é comentada. O fim deste bloco é precedido da sequência "*)".	(* bloco de linhas comentadas *)
Citações da fonte original	Toda a informação que é delimitada por aspas.	"citação original"
Keywords da notação Kleio	Dizem respeito a grupos.	fonte
	São elementos.	nome
	Definem relações.	casado

5.2.6. Sugestões

Esta funcionalidade visa complementar a funcionalidade da secção anterior, isto é, é possível ao utilizador completar palavras-chave que dizem respeito a grupos, no entanto estas têm parâmetros obrigatórios (como, por exemplo, se verifica nos ciclos *for* da linguagem Java). Com a implementação destas sugestões é possível inserir no editor a estrutura genérica da *keyword* de grupo que o utilizador escolheu. Para isso foram definidas as estruturas das *keywords* de grupo num ficheiro JavaScript, o que permitiu ao Ace interpretá-las para depois serem sugeridas ao utilizador, tal como acontece nos mecanismos de *autocomplete*. Na figura 8 é apresentado um exemplo da aplicação desta funcionalidade.

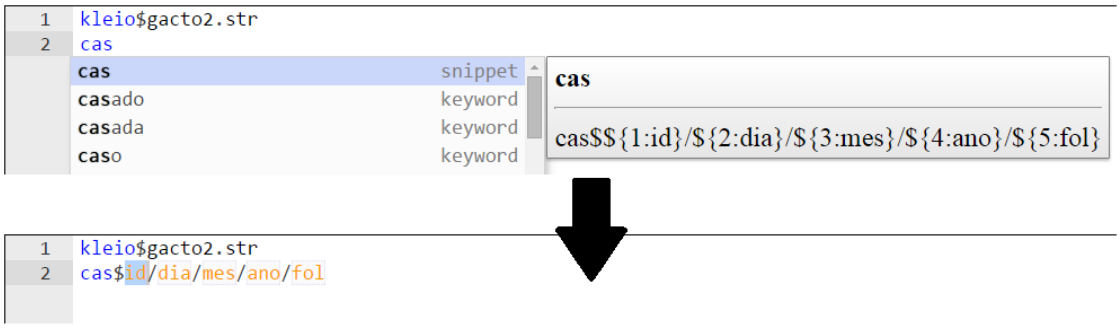

Figura 8: Exemplo de sugestões

Importante referir que as estruturas genéricas definidas no ficheiro JavaScript obedeceram ao módulo de sintaxe "mhk.xml", que já se encontrava definido antes do início deste estágio. Neste módulo encontram-se todas as estruturas genéricas da notação Kleio.

5.2.7. Edição Colaborativa

Este requisito não consta no produto final pois em vez de acrescentar melhorias ao produto final, adicionou algumas limitações. No entanto, este requisito foi implementado e testado.

Para este requisito fazer parte da aplicação é necessário guardar numa variável a instânica do Ace e noutra variável a URL do Firebase. Estas variáveis depois são passadas como parâmetros de entrada no construtor do Firepad e assim é adicionada esta funcionalidade à aplicação.

Basicamente, o Firepad adiciona a funcionalidade de código colaborativo à aplicação através da comunicação com o Firebase (onde vai buscar informação sobre os utilizadores) e o Ace (onde disponibiliza toda a informação). Quaisquer alterações em relação ao utilizadores são comunicadas imediatamente ao Firepad e este aplica-as de seguida no editor Ace.

5.3. Testes e Validação

Esta secção consiste na realização de um leque de testes à aplicação desenvolvida, o que permite validar o cumprimento dos objetivos delineados para o âmbito no qual se insere este estágio.

5.3.1. Testes Unitários e de Integração

Durante o processo de desenvolvimento foram efetuados testes unitários e de integração. Os testes unitários consistiram em testar o comportamento individual da funcionalidade antes da mesma ser integrada na aplicação. Só depois de passar nestes testes é que a funcionalidade foi integrada na aplicação e seguidamente foram feitos os testes de integração, com o propósito de perceber se a aplicação continuava a funcionar corretamente. Este processo foi repetido para todas as funcionalidades implementadas.

5.3.2. Testes de Sistema

Estes testes têm como objetivo validar todas as funcionalidades implementadas. Estes são apresentados segundo o modelo que pode ser visto na tabela 9. Nessa tabela para cada teste é: definido um ID único, associado um ID do requisito que está a ser testado (ID esse que foi atribuído aquando do levantamento de requisitos na subsecção 4.1.2.), adicionada uma descrição, especificada a ação executada pelo utilizador e reportados os resultados esperados e óbitos. Caso o resultado obtido for de encontro ao resultado esperado, o teste é bem-sucedido (Passou), caso contrário, o resultado é negativo (Falhou). Os testes de sistema encontram-se tabelados no Anexo B.

Tabela 9: Tabela de modelo dos testes de sistema

ID-Teste	T.X
ID-Requisito	RF.X
Descrição	Descrição da funcionalidade testada.
Ação	Ação executada.
Resultado Esperado	O que se espera.
Resultado Obtido	O que se obtém.
Resultado do Teste	Passou / Falhou

5.3.3. Testes de Aceitação dos Requisitos Funcionais

Estes testes tiveram como finalidade validar e verificar que todos os requisitos se encontravam implementados e o seu comportamento era o esperado. Para isso foi realizada uma reunião com o orientador externo, depois da fase de desenvolvimento das funcionalidades ser dada como concluída e de todos os testes de sistema terem passado com êxito.

Tabela 10: Resultados dos testes de aceitação dos requisitos funcionais

ID	**Requisito**	**Funcionalidade implementada**	**Comportamento esperado**
RF.1	Editor de texto	Sim	Sim
RF.2	Editar ficheiros	Sim	Sim
RF.3	Gravar ficheiros	Sim	Sim
RF.4	*Autocomplete*	Sim	Sim
RF.5	Realce da sintaxe	Sim	Sim
RF.6	Sugestões	Sim	Sim
RF.7	Edição colaborativa	Sim	Não

Na tabela anterior verifica-se que todos so requisitos fundamentais ao correto comportamento da aplicação foram concluídos com sucesso, ficando a faltar apenas um requisito (RF.7) que tinha prioridade baixa. Como já foi referido em secções anteriores, este requisito foi implementado mas deixado de parte visto não ter o comportamento esperado, pois este adicionava algumas limitações ao sistema TIME LINK.

5.3.4. Testes de Aceitação dos Requisitos Não-Funcionais

Para além dos testes de aceitação acerca das funcionalidades da aplicação, é importante também validar e verificar se os atributos de qualidade (enumerados e descritos na secção 4.1.2.) foram corretamente desenvolvidos.

Tendo em conta que todas as alterações aos atributos de qualidade seriam processos simples, pois os requisitos funcionais que foram delineados nunca seriam postos em causa, estes testes foram os últimos a serem feitos.

Portabilidade

O acesso à aplicação é feito por diferentes utilizadores e esta premissa implica, necessariamente, que esse acesso possa ser feito em diferentes sistemas operativos e diferentes navegadores. Para isso a aplicação foi testada nos navegadores mais utilizados nos dias de hoje.

Tabela 11: Resultado dos testes do requisito não funcional - Portabilidade

Navegador	Versão	Compatível
Safari	7.0.1	Sim
Google Chrome	42.0.2311.152	Sim
Mozilla Firefox	38.0.5	Sim

Usabilidade

Dado que a aplicação pretende agilizar a introdução dos dados por parte do utilizador, interessa que toda a interface da mesma vá de encontro às suas necessidades e objetivos, apresentando-se simples e intuitiva.

Decidiu-se que todas as alterações ao interface da aplicação seriam discutidas entre a equipa, onde fazem parte o orientador externo e a Doutora Ana Isabel Ribeiro, e validadas na reunião seguinte.

Algumas das alterações mais pertinentes foram:

- Quando o utilizador tinha que apagar todo o conteúdo escrito no editor este precisava, em primeiro lugar, de selecionar todo o texto e só depois é que poderia apagá-lo. Para facilitar este processo foi adicionado um botão "Limpar";
- O esquema de cores do tema definido no Ace não estava em concordância com o do TIME LINK. Sendo assim, esse esquema de cores foi alterado para ir de encontro ao pretendido;
- Para guardar o texto que está no editor para um ficheiro é pressionado o botão "Guardar". No entanto, caso o utilizador carregasse neste sem querer, o texto era gravado na mesma. Para evitar estes casos é pedida uma confirmação, em que o utilizador pode continuar a ação ou cancelar a mesma;
- Depois de adicionado o botão "Limpar" foi necessário adicionar uma confirmação, devido às mesmas razões faladas no ponto anterior. Esta confirmação segue o mesmo mecanismo que a confirmação mencionada no ponto anterior.

Capítulo 6
Conclusões

Neste capítulo é apresentada uma reflexão sobre o trabalho desenvolvido ao longo deste estágio, as contribuições do estagiário e o trabalho que poderá ser feito no futuro. Neste documento é retratada toda a dedicação e esforço ao longo deste estágio.

6.1. Reflexões Finais

Durante todo o estágio existiu contato com diferentes pessoas. Numa primeira fase foi realizada uma reunião formal com o CEO da iClio, Alexandre Pinto, que foi importante para clarificar o âmbito e objetivos do estágio. Nos meses seguintes, fui trabalhando com o orientador externo (Doutor Joaquim Ramos de Carvalho), fundador do TIME LINK e, inerente a isso, com uma vasta experiência de utilização, e que me apoiou em todas as decisões de especificação e desenvolvimento da aplicação e com a Doutora Ana Isabel Ribeiro, que por ser uma utilizadora ativa e também com uma vasta experiência de utilização, foi um apoio importante para ir de encontro às necessidades do utilizador comum.

Neste documento é visível, de forma clara e objetiva, todo o processo que levou à conceção da aplicação. Este iniciou-se com o planeamento; de seguida, foi feita a pesquisa por aplicações concorrentes; depois foi especificada a solução a implementar; por fim, foram descritas todas as decisões em termos de desenvolvimento da aplicação.

Metodologia e Planeamento

A escolha da metodologia de trabalho foi importante pois permitiu fazer uma divisão de tarefas coerente com os requisitos necessários à plataforma e tornou possível a implementação dos mesmos. O planeamento feito no início do 1º semestre (meados de Setembro) permitiu ter uma visão global do esforço pretendido para o estágio e para cada semestre. Este planeamento foi importante para servir de referência na metodologia de trabalho. Apesar de ser uma metodologia ágil posso dizer que o planeamento foi cumprido quase na sua totalidade. De referir que apenas não foram cumpridas as últimas metas de Janeiro e Junho de 2015 pois a época de exames não correu como esperado.

Estado da Arte

A análise e estudo do estado da arte foi importante para perceber como as melhores aplicações de edição de código do mercado funcionam e quais as funcionalidades que oferecem aos seus clientes. O conhecimento obtido foi importante para aplicar algumas ideias e assim melhorar a aplicação de forma a poder oferecer aos utilizadores funcionalidades mais intuitivas e que vão de encontro às suas necessidades.

Especificação

A especificação do sistema engloba o levantamento de requisitos, o diagrama de casos de uso, as *mockups* da aplicação e a arquitetura da aplicação.

Uma parte importante deste capítulo foi o levantamento de requisitos pois permitiu identifica-los, especificá-los, alterá-los, validá-los e esquematizá-los. Depois de todos estes processos, os requisitos estão definidos e serão importantes para perceber de que forma o projeto está a correr.

Toda a modelação permitiu que no 2º semestre a maior parte da atenção recaísse no desenvolvimento e testes do sistema a desenvolver.

A arquitetura da aplicação foi importante para se perceber como funciona a interação entre as diferentes componentes mesma.

Desenvolvimento

Aqui foi seguida uma metodologia ágil o que permitiu fazer uma divisão de tarefas coerente com os requisitos necessários à plataforma e tornou possível a implementação dos mesmos. Ainda foram efetuados um conjunto de testes com o intuito de garantir que a aplicação se comportava como era pretendido.

Conclusão Final

Para concluir, o resultado do trabalho desenvolvido está de acordo com o esperado, conseguindo mostrar autonomia e perseverança por parte do autor.

Esta foi a primeira vez que o autor contribuiu para o desenvolvimento de um sistema das dimensões do TIME LINK, sendo que todas as abordagens, conhecimentos e técnicas adquiridas deveram-se ao acompanhamento de ambos os orientadores, Doutor Joaquim Ramos de Carvalho e Doutor Mário Zenha Rela. Sem estes, o autor teria tido uma maior dificuldade para alcançar os objetivos a que se propôs. Esta experiência permitiu perceber a diferença entre o desenvolvimento de projetos académicos e profissionais.

Com a implementação destas novas funcionalidades pode dizer-se que algumas limitações do sistema foram dissipadas, nomeadamente no que toca à compreensão da notação Kleio e à manipulação de ficheiros, acelerando e agilizando o processo de transcrição de informação biográfica. Foram alcançados com sucesso os objetivos definidos, tendo em conta os resultados provenientes das validações dos requisitos.

6.2. Contribuições

A base existente para este estágio foi o sistema TIME LINK, com o qual a aplicação desenvolvida interage. Coube ao autor desenvolver a aplicação em si e tomar todas as decisões arquiteturais para os requisitos especificados.

As principais contribuições do autor foram:

- Levantamento de requisitos;
- Especificação do modelo arquitetural;
- Implementação de um editor de código altamente eficaz, que interage com o utilizador, produz *feedback* e dá suporte à notação Kleio;
- Integração com o sistema TIME LINK;

A aplicação desenvolvida pelo estagiário está atualmente integrada no TIME LINK e é uma das (várias) funcionalidades que estão disponíveis para os utilizadores.

6.3. Trabalho Futuro

Como ficou provado a solução é funcional e apresenta um leque de funcionalidades que interessam aos utilizadores que pretendem transcrever dados biográficos. Contudo, algum trabalho futuro pode aproximar ainda mais a aplicação da solução ideal. Este passa por:

- Integrar o Firepad na aplicação desenvolvida. Isto daria outra dimensão a todo o processo de transcrição dos dados biográficos, pois seria possível a equipas com um número significativo de elementos poderem adicionar os dados transcritos ao mesmo ficheiro de forma colaborativa, controlarem versões do código e controlarem o esforço de cada elemento. Esta melhoria iria melhorar significativamente a produtividade.

- Melhorar a forma como as palavras-chave e estruturas genéricas da notação Kleio são interpretadas. No produto final estas são guardadas de modo estático, o que constitui uma limitação pois o TIME LINK está em constante mudança e as palavras-chave e estruturas genéricas da notação Kleio são atributos que podem ser modificados. Uma solução é implementar um método que aceda, dinamicamente, ao módulo de sintaxe que está presente no ficheiro "mhk.xml" e que retire deste toda a informação necessária.

Referências

[1] "History for the New Media," iClio, [Online]. Available: http://www.iclio.net/. [Acedido em 15 Janeiro 2015].

[2] "Histoy for the New Media," iClio, [Online]. Available: http://www.iclio.net/company.html. [Acedido em 15 Janeiro 2015].

[3] "Time Link - Home," Time Link, [Online]. Available: http://timelink.fl.uc.pt/. [Acedido em 01 Setembro 2015].

[4] K. Schwaber e M. Beedle, Agile Software Development with Scrum, 2002.

[5] "Git," Git, [Online]. Available: https://git-scm.com/. [Acedido em 27 Março 2015].

[6] "GitHub," GitHub, [Online]. Available: https://github.com/. [Acedido em 23 Agosto 2015].

[7] T. B. Ceia, "Desenvolvimento de software em aplicações Web em Java," 2012/2013.

[8] J. M. S. A. N. d. Carvalho, "Time link : a evolução de uma base de dados prosopográfica," 2010.

[9] "Learn HTML5 From Scratch," Udemy, [Online]. Available: https://www.udemy.com/learn-html5-programming-from-scratch/. [Acedido em 20 Outuro 2014].

[10] "Free JQuery and JavaScript Training," Udemy. [Online]. [Acedido em 20 Outubro 2015].

[11] "Free Time Tracking Software & App," Toggl, [Online]. Available: https://www.toggl.com/. [Acedido em 01 Setembro 2015].

[12] "The world's most popular open source database," MySQL, [Online]. Available: https://www.mysql.com/. [Acedido em 23 Fevereiro 2015].

[13] "Programmer's Text Editor," jEdit, [Online]. Available: http://www.jedit.org/. [Acedido em 08 Novembro 2014].

[14] "ShiftEdit - Online IDE," ShiftEdit, [Online]. Available: https://shiftedit.net/. [Acedido em 8 Novembro 2014].

[15] "Developer Workspaces Controlled By Any IDE," Codenvy, [Online]. Available: https://codenvy.com/. [Acedido em 08 Novembro 2014].

[16] "Your development environment, in the cloud," Cloud9, [Online]. Available: https://c9.io/. [Acedido em 08 Novembro 2014].

[17] "Cross Plataform Cloud IDE," codeanywhere, [Online]. Available: https://codeanywhere.com/. [Acedido em 08 Novembro 2014].

[18] "An open source collaborative code and text editor," Firepad, [Online]. Available: https://firepad.io/. [Acedido em 01 Setembro 2015].

[19] "The High Performance Code Editor for the Web," Ace, [Online]. Available: http://ace.c9.io/. [Acedido em 10 Dezembro 2014].

[20] "Build Extraordinary Apps," Firebase, [Online]. Available: https://www.firebase.com/. [Acedido em 01 Setembro 2015].

[21] "ShareLaTeX, the Online LaTex Editor," ShareLaTeX, [Online]. Available: https://www.sharelatex.com/. [Acedido em 01 Setembro 2015].

[22] "Licença BSD," [Online]. Available: http://opensource.org/licenses/BSD-3-Clause.

[23] "Apache License 2.0," [Online]. Available: http://www.apache.org/licenses/LICENSE-2.0.

[24] "The MIT License," [Online]. Available: http://opensource.org/licenses/MIT.

[25] "Mars Eclipse," [Online]. Available: https://eclipse.org/. [Acedido em 12 Março 2015].

[26] "Documentos do Google - Crie e edite documentos online, gratuitamente," Google, [Online]. Available: https://www.google.com/docs/. [Acedido em 5 Março 2015].

[27] "XMLHttpRequest - Web API Interface," MDN, [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest. [Acedido em 01 Setembro 2015].