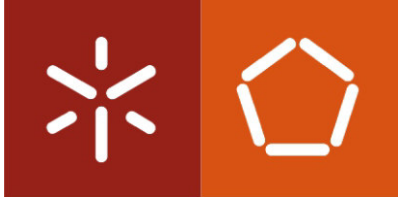
Universidade do Minho
Escola de Engenharia

Daniel Bruno Teixeira Teles Meneses

Geração de aplicações a partir de especificações UML como fator de sucesso para o desenvolvimento de sistemas de informação

Outubro de 2015

Geração de aplicações a partir de especificações UML como fator de sucesso para o desenvolvimento de sistemas de informação
Daniel Bruno Teixeira Teles Meneses
UMinho | 2015



Universidade do Minho
Escola de Engenharia

Daniel Bruno Teixeira Teles Meneses

Geração de aplicações a partir de especificações UML como fator de sucesso para o desenvolvimento de sistemas de informação

Dissertação de Mestrado
Mestrado integrado em Engenharia e
Gestão de Sistemas de Informação

Trabalho Efetuado sob a orientação do
Professor Doutor João Eduardo Quintela Alves de Sousa Varajão

Outubro de 2015

DECLARAÇÃO RELATIVA ÀS CONDIÇÕES DA REPRODUÇÃO DA DISSERTAÇÃO:

Nome: Daniel Bruno Teixeira Teles Meneses

Correio eletrónico: danielmeneses.pt@gmail.com

Tel./Tlm: 913022727

Número do Bilhete de Identidade: 12411315

Título da dissertação: Geração de aplicações a partir de especificações UML como fator de sucesso para o desenvolvimento de sistemas de informação

Ano de conclusão: 2015

Orientador: Prof. Doutor João Eduardo Quintela Alves de Sousa Varajão

Designação do Mestrado: Mestrado integrado em Engenharia e Gestão de Sistemas de Informação

Área de Especialização:

Universidade do Minho, __/__/____

Assinatura: _____

Agradecimentos

Esta dissertação surge como um objetivo pessoal e académico a que me propus. Representa muitas horas de trabalho e de reflexão que não seria possível alcançar sem a ajuda do Prof. Doutor João Eduardo Quintela Alves de Sousa Varajão, meu orientador. Expresso o meu apreço pela sua orientação científica, partilha de conhecimento, profissionalismo, revisão crítica deste trabalho de dissertação e pela sua disponibilidade.

À minha namorada, Eduarda, agradeço o apoio na realização da dissertação em todo o percurso académico em geral.

Ao José, agradeço o companheirismo em todo o percurso académico.

Por fim, agradeço a todas as pessoas que, de uma forma direta ou indireta, contribuíram para o sucesso do meu percurso académico às quais desejo todo o sucesso para as suas vidas profissionais, académicas e pessoais.

Resumo

Frequentemente as organizações não documentam os seus sistemas de informação (SI) ou sistemas informáticos. Este facto provoca graves problemas no que diz respeito à continuidade dos processos, aplicações de tecnologias da informação e SI. No que respeita ao sistema informático das organizações, sendo este um elemento importante do SI, a documentação é também muitas vezes inexistente, podendo traduzir-se no descontrolo do seu desenvolvimento e manutenção.

No presente trabalho estabeleceu-se como objetivo principal a criação de uma aplicação com capacidade de gerar aplicações de *software* e respetivo código-fonte, a partir de especificações UML (*Unified Modeling Language*), assim promovendo e incentivando a realização de documentação de especificação.

Adotou-se Scrum como abordagem metodológica para o processo de desenvolvimento de *software*, tendo sido definido um processo personalizado de acordo com as circunstâncias particulares do trabalho a desenvolver.

Foi possível concretizar a aplicação proposta, possuindo esta atualmente a capacidade de gerar aplicações *Web*. As aplicações geradas são completamente funcionais, no entanto, são suscetíveis de serem modificadas em aspetos como, por exemplo, apresentação e usabilidade.

Palavras-chave: Sistemas de Informação; especificações UML; Scrum; Aplicações de *software*.

Abstract

Often organizations do not document their information systems or computer systems. This causes serious problems regarding the maintainability of processes, activities and operations performed in the organizational context. Regarding computer systems, which are important components of information systems, documentation is in most cases lacking, resulting in uncontrolled development and maintenance.

In this dissertation we propose the development of an application that has the ability to generate source code from UML (*Unified Modeling Language*) specifications, promoting and encouraging the documentation of applications.

We have adopted the Scrum methodology approach for the development of the software application.

It was possible to implement the application and it has the ability to generate applications following the proposed guidelines. The generated applications are completely functional. Notwithstanding, they are susceptible to modifications as for instance, improvement of appearance or user experiences aspects.

Keywords: Information system; UML specifications; Scrum; Software applications.

Índice

Índice de figuras

Índice de tabelas

Siglas e acrónimos

ACL: *Access Control List*

API: *Application Programming Interface*

BPMN: *Business Process Modeling Notation*

CDG: *Custom Database Generation*

CSCG: *Custom Source Code Generation*

CRUD: *Create, Read, Update and Delete*

CSS: *Cascading Style Sheets*

DOM: *Document Object Model*

DRY: *Don't Repeat Yourself*

ERD: *Entity relationship diagrams*

EPC: *Event-driven Process Chain*

HTML: *HyperText Markup Language*

ICB: *IPMA Competence Baseline*

IPMA: *International Project Management Association*

ISDocPal: *Information System Documentation Pal*

MVC: *Model-View-Controller*

PHP: *Hypertext Preprocessor*

PMBoK: *Project Management Body of Knowledge*

PMI: *Project Management Institute*

POO: Programação Orientada a Objetos

Prince2: *Projects IN a Controlled Environment*

SI: Sistemas de Informação

UID/UXD: *User Interface Design/User Experience Design*

UML: *Unified Modeling Language*

Yii: *Yes it is* (*framework* PHP)

1. Introdução

Neste capítulo são abordados diversos assuntos relevantes para a compreensão do trabalho realizado, tais como: o enquadramento do tema; a descrição do problema e proposta de solução; a enumeração dos objetivos definidos para a presente dissertação; a identificação da abordagem metodológica adotada; e as tecnologias e ferramentas utilizadas no desenvolvimento do trabalho.

1.1. Enquadramento e relevância do tema

Tecnologias da Informação (TI) e Sistemas de Informação (SI) são conceitos sem um entendimento universal, de modo que é pertinente apresentar as suas definições (Amaral & Varajão, 2000, 2007). É, assim, importante esclarecer os termos "aplicações" e "sistemas de informação" no contexto deste trabalho. O termo "aplicações" refere-se a aplicações de *software* e poderá também ser entendido como aplicações informáticas. Relativamente a SI, existem diversas definições, nem sempre consensuais. SI é recorrentemente entendido como o domínio científico que se debruça sobre o estudo e desenvolvimento da adoção e utilização de aplicações de tecnologia da informação no suporte ao funcionamento das organizações (Carvalho, 1996). No entanto, outras definições de SI não estão unicamente associadas às tecnologias de suporte ao SI, entendendo que se trata de um sistema de atividade humana que poderá ser suportado por computadores (Buckingham, Hirschheim, Land, & Tully, 1986). No contexto do presente trabalho, SI deve ser entendido como uma abstração do sistema organizacional com foco nas atividades capazes de fornecer informação aos agentes organizacionais.

O aumento crescente do volume de dados e de informação é reflexo direto da cada vez maior complexidade das organizações e da sociedade em geral. Daqui radica a necessidade absoluta da existência de sistemas que assegurem a recolha, armazenamento, processamento, consulta e comunicação da informação necessária para o desenvolvimento desejado das suas atividades (Varajão, 2003a, 2005). A correta adoção de TI nas organizações é um fator chave para o alcance de resultados de negócio superiores e para a obtenção de vantagens competitivas (Gonzálvez-Gallego, Soto-Acosta, Trigo, Molina-Castillo, & Varajão, 2010), dado que as capacidades internas e externas de TI e SI aumentam o desempenho das organizações (Gonzálvez-Gallego, Molina-Castillo, Soto-Acosta, Varajão, & Trigo, 2015).

As TI desempenham um papel fundamental nas organizações contemporâneas (Carriço, Varajão, Fernandes, & Dominguez, 2014; Varajão, 2001), estando presentes em praticamente todas as vertentes dos negócios (Amaral, Fernandes, & Varajão, 2015) sendo várias as motivações para a

sua adoção (Varajão, Ribeiro, Figueiredo, & Barroso, 2007; Varajão, Trigo, & Barroso, 2009; Varajão, Trigo, Figueiredo, Barroso, & Bulas-Cruz, 2009). As motivações vão desde o aumento da produtividade e redução de custos operacionais (Varajão, Trigo, Figueiredo, & Barroso, 2007a, 2007b) até aspetos estratégicos (Trigo, Varajão, Soto-Acosta, González-Callego, & Castillo, 2015).

Com o passar o tempo, torna-se cada vez mais evidente que sem o uso eficiente e eficaz das TI/SI, as empresas não podem ser competitivas e, na grande maioria dos casos, a própria sobrevivência depende dessa capacidade (Trigo, Varajão, Figueiredo, & Barroso, 2007; Varajão, 1997, 2003a; Varajão, 2005). As fundações para um bom desempenho de uma organização dependem de um estreito alinhamento entre os objetivos do negócio e as capacidades dos SI/TI (Baptista, Varajão, & Moreira, 2013).

1.1.1. Descrição do problema

Vivemos numa Era em que impera a urgência da disponibilidade de ferramentas que economizem tempo às equipas que desenvolvem e mantêm aplicações de *software*, que promovam o aumento da produtividade das organizações e diminuam o custo de desenvolvimento e manutenção. A ausência de documentação e especificação dos sistemas informáticos pode comprometer o sucesso de desenvolvimento dos SI, pois pode colocar em causa a continuidade/manutenção das aplicações. Essa falta de documentação deve-se, tipicamente, à escassez de recursos e ao facto de a mesma não ser reconhecida como uma mais-valia direta para o próprio sistema informático. A documentação e especificação dos sistemas informáticos tem de passar a ser vista como um investimento e não como um desperdício de tempo.

Alguns dos problemas que podem ser observados como tendo origem direta na ausência de documentação são o descontrolo no desenvolvimento e na manutenção dos sistemas informáticos. A falta de documentação que claramente identifique os requisitos dos sistemas informáticos pode originar um produto final integralmente ou parcialmente diferente do pretendido. Um segundo problema está relacionado com o aumento do tempo de desenvolvimento devido à falta de conhecimento do negócio e à necessidade frequente de trabalho corretivo, o que consequentemente afeta negativamente o custo de desenvolvimento.

É um facto que, tanto a organização cliente como a organização que desenvolve o sistema informático são diretamente afetadas quando a documentação é escassa ou inexistente, pois os efeitos negativos assumem uma proporcionalidade direta, i.e., se o desenvolvimento produz um custo mais elevado para quem o desenvolve, então o cliente paga mais por este desenvolvimento. O mesmo

acontece relativamente ao tempo de desenvolvimento: quanto mais tempo este demora, mais tarde o cliente recebe o seu sistema informático, o que representa um incremento indireto nos custos para o cliente.

1.1.2. Proposta de solução

Perante o problema apresentado, a presente dissertação visa o desenvolvimento de uma aplicação capaz de gerar aplicações informáticas através de especificações UML, promovendo assim a criação de documentação *à priori* do processo de desenvolvimento facilitando o próprio processo de desenvolvimento de *software*. A proposta apresentada permite justificar a escolha do título escolhido para esta dissertação, e este foi: "Geração de aplicações a partir de especificações UML como fator de sucesso para o desenvolvimento de sistemas de informação".

1.2. Objetivos

O objetivo geral desta dissertação reside em desenvolver uma aplicação de *software* (ISDocPal) que possua a capacidade de gerar aplicações funcionais através de especificações de UML, possibilitando posteriores modificações ao seu código fonte.

Para alcançar o objetivo geral, foram delineados objetivos específicos que orientaram as diversas fases desta dissertação:

1- Identificar aplicações semelhantes à aplicação proposta na presente dissertação e exercer respetiva análise crítica;
2- Definir a arquitetura lógica da aplicação a desenvolver;
3- Estabelecer um caso exemplificativo que permita demonstrar a utilidade e mais-valia da aplicação desenvolvida;
4- Analisar criticamente a aplicação desenvolvida e sugerir melhorias futuras.

Tendo por referência os objetivos definidos a presente dissertação assume o carácter de ***projeto***, devendo ser encarado como tal.

1.3. Síntese da Metodologia e ferramentas tecnológicas utilizadas

Na presente dissertação adotou-se a Scrum como metodologia para o processo de desenvolvimento de *software*. Nos capítulos 2 e 4 esta metodologia é descrita em detalhe.

As ferramentas e tecnologias utilizadas para o desenvolvimento da aplicação proposta são: a *Unified Modeling Language* (UML); o *Hipertext Preprocessor* (PHP); o padrão *Model, View and Controller* (MVC); a Yii2 *framework*; a livraria *React.js*; a *framework Node.js*; a *framework Bootstrap*,

e o sistema de gestão de ficheiros Git. No decorrer do capítulo 2 são descritas em detalhe todas estas tecnologias e justifica-se a sua adoção.

1.4. Organização da dissertação

No presente capítulo, introdutório, é efetuado o enquadramento do presente trabalho.

No capítulo 2 apresentam-se as abordagens metodológicas, as diversas ferramentas adotadas para o desenvolvimento da aplicação proposta e, por fim, são apresentadas aplicações semelhantes à aplicação proposta.

No capítulo 3 descreve-se a conceptualização da ISDocPal, assim como se apresenta a sua arquitetura lógica.

No capítulo 4 é descrito em detalhe o processo Scrum definido para o desenvolvimento da aplicação proposta. Apresentam-se ainda os requisitos e a especificação da ISDocPal assim como um exemplo de sua implementação

No capítulo 5 apresentam-se as conclusões da dissertação e são tecidas algumas considerações relativamente a trabalhos futuros.

2. Revisão de literatura

Neste capítulo apresentam-se as abordagens metodológicas e as diversas ferramentas adotadas para o desenvolvimento da aplicação proposta, justificando a sua adoção. Identificam-se ainda algumas aplicações semelhantes à aplicação proposta para desenvolvimento e apresenta-se uma crítica individual relativamente à sua aplicabilidade no contexto do desenvolvimento de SI. Neste sentido, importa referir que após uma pesquisa exaustiva foi possível encontrar várias aplicações potencialmente semelhantes, no entanto optou-se por descrever apenas duas, pois são as que mais se assemelham quanto ao âmbito e finalidade do presente trabalho. O processo de investigação passou unicamente pela pesquisa de informação disponível na internet (*Web*).

2.1. Abordagens metodológicas

Nesta secção são abordadas a gestão de projetos e o processo de desenvolvimento de *software*.

2.1.1. Gestão de projetos

A gestão de projetos é usada de forma crescente pelas organizações para auxiliar a alcançar os seus objetivos de negócio (Catarino, Gonçalves, Pereira, & Varajão, 2009). São vários os fatores de sucesso da gestão de projetos como, por exemplo, a seleção de um gestor de projetos competente (Colomo-Palacios, González-Carrasco, López-Cuadrado, Trigo, & Varajão, 2014; Varajão & Cruz-Cunha, 2013). A gestão de projetos é essencial no contexto bem-sucedido dos projetos, sendo transversal e tendo aplicação em muitos setores (Santos & Varajão, 2015).

Nos últimos anos os projetos de desenvolvimento de TI (Varajão, Cardoso, Gonçalves, & Cruz, 2008), assim como os projetos de desenvolvimento de SI, tem revelado baixos níveis de sucesso (Varajão et al., 2008; Gonçalves, Cruz, & Varajão, 2008), apesar da medida do sucesso ser complexa devido às várias perspetivas que existem sobre o tema (Paiva, Varajão, Domínguez, & Ribeiro, 2011; Rijo, Varajão, & Gonçalves, 2012). Na realidade, existem diversas definições para o sucesso dos projetos e para o sucesso da gestão de projetos, e vários fatores podem influenciá-los (Ribeiro, Paiva, Varajão, & Dominguez, 2013; Varajão, Domingues, Ribeiro, & de Paiva, 2014). A investigação sobre os fatores relacionados com o sucesso da gestão de projetos tem evoluído significativamente nos últimos anos (C. Santos, Santos, Tavares, & Varajão, 2014).

O estabelecimento de práticas de gestão de projetos eficientes e eficazes continua a ser um desafio (Liberato, Varajão, & Martins, 2015). Para um projeto ser bem-sucedido é preciso gerir todas as atividades e aspetos do mesmo (Varajão, Dominguez, Ribeiro, & Paiva, 2014).

No contexto de um trabalho desta natureza, a gestão de projetos assume uma importância fundamental. O PMI (*Project Management Institute*) define Gestão de Projetos como sendo o "conjunto de processos aplicados a um projeto para produzir um produto ou serviço. É a arte de coordenar atividades com o objetivo de atingir as expectativas dos indivíduos e das organizações diretamente envolvidas no projeto ou aqueles cujos interesses podem ser afetados de forma positiva ou negativa no decorrer do projeto ou após a sua conclusão" (PMI, 2008).

Para a gestão de projetos existem diversos referenciais, sendo que os mais conhecidos e utilizados são: o PMBoK (*Project Management Body of Knowledge*) (P. M. I. PMI, Inc., 2015); o Prince2 (*Projects IN a Controlled Environment*) (ALEXOS, 2015); e o ICB (*IPMA Competence Baseline*) (IPMA, 2015). Cada um destes referenciais é sustentado por um conjunto de boas práticas, que surgem do sucesso da sua utilização em diversas organizações e que revelaram ser um contributo para o sucesso na gestão de projetos. De um modo geral, os seus objetivos principais são a identificação e descrição das diversas fases de um projeto e respetivas atividades, com a finalidade de atingir o sucesso na gestão de projetos. Contemplam linhas orientadoras que descrevem quais os processos que devem ser levados a cabo em determinada fase do projeto e quais os artefactos que devem ser recolhidos e obtidos no início e final de cada uma das fases. Outra característica importante é que assentam numa base de conhecimento evolutivo, i.e., são atualizados mediante o aparecimento de novos paradigmas, técnicas e/ou ferramentas que provem ser uma melhor abordagem ou simplesmente acrescentem melhorias ao processo de gestão de projetos. Esta abordagem permite reunir uma base de conhecimento mais consolidada, garantindo um conjunto de boas práticas para área de gestão de projetos.

O PMBoK é um dos mais respeitáveis referenciais para a gestão de projetos. Publicado pelo PMI (*Project Management Institute*) em 1996, sob a forma de um guia, descreve as melhores práticas para a gestão de projetos apresentando-se como uma das soluções mais completas, cobrindo diversas áreas de gestão de projetos, tais como: gestão da integração; gestão do âmbito; gestão do tempo; gestão do custo; gestão da qualidade; gestão de *stakeholders*; gestão dos recursos humanos; gestão da comunicação; gestão do risco; e gestão de aquisições/contratações.

O referencial apresenta 47 processos de gestão, organizado em cinco grupos de processos, também consideradas de fases do projeto: iniciação; planeamento; execução; monitorização e

controlo; e encerramento (P. M. I. PMI, Inc., 2015). Cada uma destas fases pressupõe um conjunto de atividades, sendo estas realizadas pelas respetivas áreas de gestão. Tal garante que todas as áreas de gestão ou de conhecimento intervêm e contribuem para o sucesso da gestão de projetos.

Adotou-se o referencial PMBoK, edição 5 como guia para a gestão do projeto realizado na presente dissertação e pelo facto de se tratar de um referencial com uma base de conhecimento sólido, fases bem definidas, atividades claras e adequadas ao trabalho em desenvolvimento.

2.1.2. Processo de desenvolvimento de *software*

O processo de desenvolvimento de software é um conjunto de atividades organizado de forma a obter produtos de software, definindo quem faz o quê, como, quando e onde (Liberato et al., 2015).

Conforme mencionado anteriormente, no contexto do processo de desenvolvimento de *software*, nesta dissertação foi adotada a metodologia Scrum. De seguida descreve-se a metodologia e justifica-se a sua escolha.

O Scrum é um modelo para o processo de desenvolvimento de produtos, criado por *Ken Schwaber* em 2009 (Scrum.org, 2015). Trata-se de uma metodologia que permite a aplicação de vários processos e técnicas de desenvolvimento. A metodologia do Scrum consiste em equipas Scrum, funções associadas, eventos, artefactos e regras. Cada componente pertencente à metodologia serve um propósito específico o que se torna essencial para o sucesso da sua utilização. Assenta na teoria de controlo empírico dos processos, i.e., o conhecimento surge da experiência e as decisões devem ser baseadas em experiências passadas.

O Scrum assenta sobre três pilares de implementação de controlo empírico de processos: transparência; inspeção; e adaptação. "Transparência" refere-se ao facto dos aspetos significativos de determinado processo deverem ser visíveis às pessoas responsáveis pelo *output* de cada processo. "Inspeção" traduz-se na necessidade de os utilizadores Scrum inspecionarem frequentemente os artefactos que permitem progredir e alcançar um objetivo de um *Sprint*, por forma a detetar variações indesejáveis. Esta inspeção deve ser frequente, no entanto, não deve colocar em causa a realização do trabalho. O último pilar, "Adaptação", determina que, se um inspetor perceber que um ou mais aspetos do processo se desviam do foco, fazendo com que o produto final não seja aceitável, o processo ou o objeto a ser processado tem de ser ajustado.

Quanto ao processo de desenvolvimento o Scrum passa, numa primeira fase, pela definição do *Product Backlog*. O *Product Backlog* é o conjunto de funcionalidades ou requisitos (*User Stories*)

identificados aquando da conceção do produto final. São funcionalidades a serem desenvolvidas ao longo do processo Scrum definido. A coleta destas *User Stories* é normalmente realizada pelo *Product Owner*. As *User Stories* são numa fase posterior organizadas em *Release Backlogs* e cada uma destas origina um ou mais *Sprints,* sendo estas atribuídas às equipas de desenvolvimento (Scrum *Team*).

O Scrum prescreve quatro eventos formais para a inspeção e adaptação: *Sprint Planning*; *Daily Scrum*; *Sprint Review*, e *Sprint Retrospective*. Estes eventos são utilizados para minimizar a necessidade de reuniões não definidas no processo Scrum concebido. De seguida descrevem-se cada um destes eventos (Sutherland, 2013):

- *Sprint Planning*: nesta fase é realizada uma reunião com a equipa de desenvolvimento com a finalidade de equacionar quais as *User Stories* a incluir no *Sprint Backlog*. O *Sprint Backlog* reúne um conjunto de tarefas que se pretendem realizar numa *Sprint*. A *Sprint* é o "coração" do Scrum e deve ter uma duração superior a uma semana e inferior a um mês. Após a sua conclusão, espera-se que uma pequena parte do produto seja concluída, funcional e potencialmente enviada para ambiente produtivo. O *output* de uma Sprint deve ser independente, i.e., não deverá depender do resultado de futuras *Sprints* para funcionar, no entanto pode depender de funcionalidades anteriormente produzidas. Uma nova *Sprint* começa após uma outra ter sido concluída, i.e., segundo o Scrum, não é aceitável que um elemento da equipa trabalhe em mais do que uma *Sprint* em simultâneo.

- *Daily Scrum*: é uma reunião diária de 15 minutos entre a equipa de desenvolvimento (Scrum *Team*) com a finalidade de acompanhar a execução das tarefas da *Sprint*, por forma a evitar eventuais desvios do foco ou do tempo para a execução. As questões fundamentais da reunião são: qual o trabalho realizado no dia anterior; qual o trabalho a realizar no próprio dia; e quais os impedimentos à realização do trabalho.

- *Sprint Review*: é realizado no fim de cada *Sprint* e tem por objetivo a análise do trabalho realizado e o respetivo *output* da *Sprint*. Tal *output* é analisado e sujeito a aceitação (ou recusa) por parte do *Product Owner*.

- *Sprint Retrospective*: é a oportunidade da equipa Scrum se questionar relativamente ao que pode ser feito para melhorar o processo de trabalho para uma próxima *Sprint*. Para tal deve ser criado um plano que permita definir o processo a seguir. Este evento deve ocorrer após a *Sprint Review* e antes do próximo *Sprint Planning*.

Optou-se pela adoção do Scrum para o processo de desenvolvimento de *software*, pois apresenta um perfil interativo, incremental, possibilitando uma incrementação evolutiva da aplicação ao longo do trabalho de forma ágil e controlada. Para o presente trabalho foram assumidos os três papéis, i.e., pelo autor, exercendo funções como: *Product Owner*, *Scrum Master* e como elemento da *Scrum Team*. Esta tarefa, embora mais complexa, foi possível e adequada ao desenvolvimento da aplicação por se tratar de um modelo ágil.

2.2. Técnicas e ferramentas de desenvolvimento

Nesta secção são apresentadas as várias e técnicas e ferramentas de desenvolvimento adotadas para o desenvolvimento da ISDocPal.

2.2.1. Representação de sistemas de informação (UML)

A representação e descrição das aplicações de *software* facilitam a sua leitura e permitem a previsibilidade das atividades e processos definidos. Neste sentido, torna-se desejável que estas aplicações sejam representadas e descritas *à priori* do seu processo de desenvolvimento. É fundamental que o desenho resultante da representação seja a base para a criação das aplicações, garantido o cumprimento dos processos tecnológicos concebidos.

Existem diversas linguagens para a representação de SI e/ou aplicações de *software* e de desenho de processos de negócio, como: UML (OMG, 2015b); BPMN (*Business Process Modeling Notation*) (OMG, 2015a); EPC (*Event-driven process chain*) (Instagram, 2015); e várias outras.

A UML foi criada por *Grady Booch, Ivar Jacobson* e *James Rumbaugh* na *Rational Software* entre 1994-95, e adotada como uma linguagem *standard* em 1997 pela OMG (*Object Management Group*), um consórcio sem fins lucrativos que até aos dias de hoje a gere. É uma linguagem para "visualização, especificação, construção e documentação" (Booch, Rumbaugh, & Jacobson, 1999) de um SI. É um dos modelos mais utilizados para a representação de sistemas informáticos em todas as suas vertentes, estrutura; comportamento; arquitetura; possibilitando ainda a modelação de processos de negócio; e estruturação de dados (OMG, 2015b).

Optou-se pela adoção da UML por vários motivos, nomeadamente por ser um requisito do trabalho proposto, também por ser extensivamente adotada pelos especialistas da área científica (OMG, 2015b), ser versátil, disponibilizar uma grande diversidade de técnicas para especificação estrutural ou comportamental de um sistema. A especificação pode ser textual (ex.: casos de uso) ou diagramática (diagrama de casos de uso, de classes, de atividades, etc.). Neste trabalho,

sobretudo, serão utilizados os diagramas de casos de uso e os diagramas de base de dados (ou entidades-relacionamentos), não pertencendo este último ao *standard* UML. A opção da inclusão de apenas estes dois diagramas justifica-se devido a limitações de tempo para a execução do trabalho e por se considerar que estas duas técnicas são as que melhor permitem representar um sistema informático numa primeira fase, visando a implementação futura de técnicas de representação que permitam detalhar os processos e as regras negócio como por exemplo: o diagrama de atividades; diagramas de estados; diagramas de sequência; etc.

2.2.2. Tecnologias da informação

Nos últimos anos, a Internet e, muito particularmente, a World Wide Web (WWW), tem-se expandido continuamente, em termos da tecnologia utilizada e em termos de dimensão, tornando-se um meio essencial para a realização de negócios (Gouveia, Oliveira, & Varajão, 2007). Na verdade a Internet mudou a forma como as organizações realizam os seus negócios (Varajão, 2003b), estimulando a colaboração e a cooperação entre pessoas e organizações (Silva, Moreira, & Varajão, 2010).

Numa área em constante evolução torna-se difícil prever quais as tecnologias de amanhã e mantermo-nos a par das tecnologias existentes no presente. É igualmente difícil a escolha das tecnologias mais adequadas aos projetos em que nos envolvemos. Tal obriga à realização de um estudo meticuloso na tentativa de perceber as vantagens e desvantagens de cada uma e tomar decisões informadas e ponderadas. Com base no estudo realizado para esta dissertação apresentam-se, de seguida, as tecnologias principais que serão utilizadas, justificando-se a sua escolha.

Linguagem de programação *server-side*

PHP é uma linguagem de programação de *server-side scripting* massivamente utilizada no desenvolvimento de aplicações em ambiente *Web* pelos grandes *players* na área de desenvolvimento. A linguagem foi criada por *Rasmus Lerdorf* em 1992 (Group, 2014) e inicialmente era apenas capaz de executar um pequeno conjunto de operações que permitiam registar o número de visitas à sua página pessoal, por esse motivo, a nomeou de *Personal Home Page Tools*. Rapidamente *Rasmus* percebeu que havia iniciado uma linguagem de programação promissora e não tardaram os pedidos de mais funcionalidades. Por isso reescreveu todo código implementando mais funcionalidades, como o acesso a bases de dados e uma interface de acesso a diversos recursos dos sistemas operativos, o que permitiu o desenvolvimento de aplicações *Web* dinâmicas.

Com estes novos incrementos, e muitos outros que foram surgindo ao longo das versões disponibilizadas, também com a colaboração de *Andi Gutmans* e *Zeev Suraski of Tel Aviv*, de Israel, a linguagem tornou-se bastante apelativa para a comunidade de desenvolvimento *Web*, sendo disponibilizada de forma gratuita e com o nome redefinido para *Hypertext Preprocessor* (PHP), nome este que se mantém até à presente data.

Atualmente a equipa que desenvolve e mantém a linguagem de programação PHP conta com a colaboração de dezenas de programadores e continua em pleno desenvolvimento (à data encontra-se na versão 5.6.5, estando a versão 7 em desenvolvimento) (Group, 2014).

Nos últimos anos tornou-se na linguagem mais utilizada (82%) para a construção de *websites* (Q-Success, 2015).

A linguagem de programação *server-side* adotada para a realização da presente dissertação foi PHP, pois trata-se de uma linguagem bastante versátil, eficiente e adequada ao desenvolvimento *Web* no que diz respeito aos recursos disponibilizados, facilidade de integração com sistemas e serviços externos. É também um requisito indireto do trabalho, pois o requisito da utilização da Yii2 *framework* obriga à utilização da linguagem de programação PHP, uma vez que esta *framework* funciona apenas sobre esta mesma linguagem.

Padrão e estruturação de código fonte

Ainda no contexto do desenvolvimento da aplicação proposta neste trabalho, sentiu-se a necessidade de adotar uma técnica que permitiu a troca de mensagens entre os seus componentes de forma organizada e previsível, tendo para tal sido adotado o padrão MVC. Este é um padrão primeiramente descrito por *Trygve Reenskaug* em 1979, que trabalhava na *Smalltalk-80 v2.0*, na *Xerox* PARC. Posteriormente revisto e reformulado em 1992 pela *Smalltalk-80 v2.5*. *Steve Burbeck* descreve a versão final do padrão MVC (Steve Burbeck, 1992) explicando que este assenta na separação de três camadas lógicas observáveis: *Model-View-Controller*. Os dados fornecidos pelo utilizador (*input*), a modelação de dados informacionais e o *feedback* visual ao utilizador são explicitamente separados e geridos por três objetos distintos, i.e., residem nos três componentes *View*, *Model* e *Controller* (ver Figura 1).

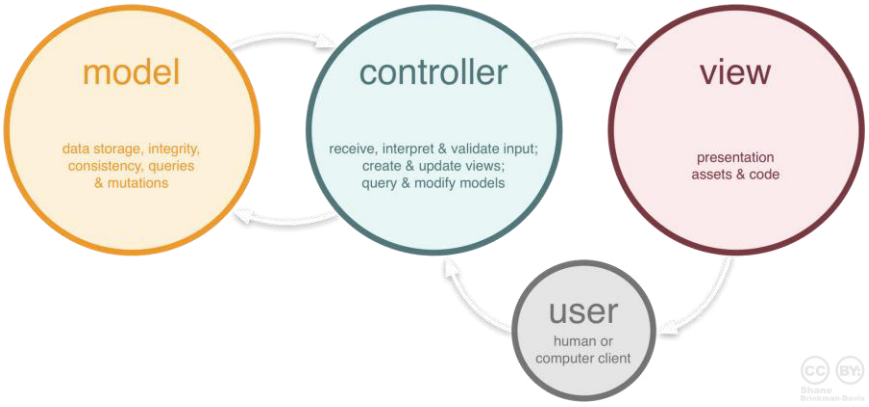


Figura 1: relacionamento entre os componentes MVC (Delamore, 2012).

A *View* trata de aspetos gráficos ou visuais, ocupando-se do modo como a informação de retorno ou resposta é apresentada ao seu destinatário no contexto da aplicação.

O *Controller* interpreta o *input* ou parâmetros de entrada direcionados ao sistema, normalmente despoletados por uma ação do utilizador do sistema.

Por último, o *Model* gere o comportamento e os dados informacionais no domínio do sistema, i.e., responde a pedidos de consulta sobre o estado da informação e responde a instruções de alteração do estado dessa mesma informação.

Framework de suporte ao padrão MVC

Nesta dissertação foi utilizada a Yii2 *framework*. Trata-se de uma das *frameworks* mais conhecidas e utilizadas para o desenvolvimento de aplicações em ambiente *Web* para a linguagem de programação PHP. Assenta no padrão MVC, o que permite conhecer e tratar os diferentes estados da aplicação de forma proativa. O nome Yii é pronunciado como "i" e é o acrónimo de "Yes it is". Os criadores da *framework* dizem ser a resposta mais acertada às várias questões que os novos utilizadores colocam quando pensam utilizar a Yii *framework*: "Is it fast? ... Is it secure? ... Is it professional? ... Is it right for my next project?" ... "Yes it is!" (LLC, 2015a).

Esta *framework* é *open-source* e encontra-se disponível para a linguagem de programação PHP (versão 5 ou superior), promove o conceito DRY (*Don't Repeat Yourself*) e o rápido desenvolvimento de aplicações *Web*. A *framework* implementa diversas otimizações de desempenho, eficiência e extensibilidade. Possibilita ainda a integração e criação de extensões/*plugins*, disponibiliza uma documentação abrangente e dispõe de uma comunidade em constante atividade e que permite a partilha de conhecimento. A Yii2 permite a manutenção do código fonte de forma organizada e adequada a projetos de média e grande dimensão. Na Figura 2 apresenta-se a estrutura de uma aplicação Yii2.

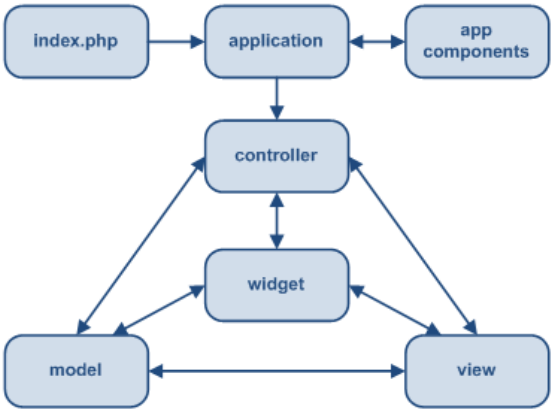


Figura 2: estrutura estática de uma aplicação Yii2 (LLC, 2015b).

A Yii2 *framework* foi adotada para o desenvolvimento da aplicação a criar neste trabalho. Esta opção deve-se à fiabilidade, diversidade de funcionalidades disponibilizadas pela *framework*, ao suporte disponibilizado pela comunidade e à existência de extensões e *plugins* disponibilizados gratuitamente. Foi também um requisito base do projeto.

Desenvolvimento do aspeto e da usabilidade da aplicação proposta

Foi ainda adotada uma biblioteca para a construção de *user interfaces* (UI). Esta biblioteca é a *React.js* e é baseada na linguagem de programação *Javascript*. Desenvolvida pelo *Facebook* em cooperação com o *Instagram*, é descrita como sendo o V (*view*) no padrão MVC. A biblioteca foi criada para resolver um problema com que o *Facebook* se deparou na construção e manutenção de grandes aplicações, em que a informação muda ao longo do tempo (collaboration, 2015). *Tom Occhino, React.js Team Leader*, relata na *React.js Conference 2015 Keynote – Introducing React Native*, que a biblioteca surge do projeto *Facebook Ads Org*, quando se depararam com problemas de manutenção do código que ironicamente piorava à medida que a esquipa de desenvolvimento aumentava, i.e., a equipa de desenvolvimento crescia, mas apenas alguns programadores tinham conhecimento profundo do código-fonte produzido até então, o que dificultava bastante a sua manutenção. *Tom Occhino* refere que a *framework* de *Javascript* assente no padrão MVC utilizada anteriormente, se mostrou ineficaz, pois toda a equipa tinha receio de fazer alterações às aplicações dado que poderia haver implicações noutras partes das aplicações, o que levava à desaceleração do desenvolvimento em geral (Developers, 2015).

Por experiência própria, de referir que as *frameworks* de *Javascript* assentes no padrão MVC podem ser contraproducentes. Esta adversidade surge da natureza da linguagem de programação *Javascript*, e da "imensidão" de eventos a serem "despoletados" que por sua vez "despoletam"

outros eventos tornando a aplicação imprevisível, o que consequentemente gera receio quanto a alterações ao código. Deve-se também ao facto da linguagem possuir uma abordagem POO (Programação Orientada a Objetos) com uma implementação própria orientada a prototipagem (*prototype*) que diverge da mais conhecida abordagem e utilizada POO com recurso a classes. *Javascript* não implementa classes nativamente, no entanto aquando da produção deste relatório já se encontram disponível a versão 6 e esta versão já permite a criação de classes, no entanto, não é extensivamente suportada pelos *browsers*.

Acrescenta-se ainda que a *React.js* funciona com base na criação de componentes. Estes componentes são autónomos e cada um destes pode incorporar outros componentes. Esta caraterística é aparentemente vulgar, no entanto, na *React.js* os componentes são efetivamente autónomos permitindo que, alterações realizadas a determinado componente não afetem outros componentes. Tal permite uma redução significativa do receio de alterações em projetos de grande escala, o que promove a aceleração do processo de desenvolvimento.

Um dos conceitos mais inovadores da *React.js* é a implementação da DOM (*Document Object Model*) virtual. Essencialmente, todo o HTML (*HyperText Markup Language*) inserido na função *render()* dos componentes *React.js*, são mapeados numa DOM virtual e sempre que ocorrem alterações de estados em algum dos componentes a livraria determina as diferenças entre a DOM virtual e atualiza a DOM real (do *browser*) (W3C, 2015) apenas com o que realmente foi modificado. Esta implementação é importante, pois as alterações à DOM (do *browser*) atrasam o processamento e *render* do HTML e, por esse motivo, torna-se importante atualizar apenas o que realmente muda.

Na Figura 3 encontra-se um exemplo de um componente *React.js* em que se pode verificar, por exemplo, a função *render()* mencionada anteriormente.

```
Code
// tutorial1.js
var CommentBox = React.createClass({
  render: function() {
    return (
      <div className="commentBox">
        Hello, world! I am a CommentBox.
      </div>
    );
  }
});
React.render(
  <CommentBox />,
  document.getElementById('content')
);
```

Figura 3: exemplo de componente *React.js* (Instagram, 2015).

Para o presente trabalho optou-se pela utilização da livraria *React.js* (versão 0.12.2) para a criação e manutenção da *user interface* devido à sua flexibilidade quanto à capacidade de gerir componentes de forma autónoma e devido à eficiência manifestada pela livraria na interação com a DOM (do *browser*) e o acréscimo de desempenho que, em consequência, esta proporciona.

No seguimento da biblioteca React.js surge a biblioteca *React native,* também desenvolvida pelo *Facebook* em cooperação com o *Instagram*. Apresentada na *React.js Conference 2015 Keynote – Introducing React Native,* como sendo a mais poderosa tecnologia para a criação de aplicações nativas para ambientes *mobile* nas plataformas *android* e IOS, utilizando a linguagem de programação *Javascript*. O lema adotado para esta nova tecnologia diverge de outras bibliotecas e *frameworks* existentes, como *phonegap* e *appcelerator* que seguem o lema "Develope once, deploy anywhere". A *React Native* introduz um novo lema "Learn once, deploy anywhere". A principal reivindicação dos criadores, é que a livraria é mesmo capaz de originar aplicações com a mesma experiência de utilização de uma aplicação nativa nos ambientes mobile (*android* e IOS), i.e., o nível de fluidez das transições, animações e interações com as aplicações será igual às aplicações obtidas com recurso a programação através das API (*Application Programming Interface*) nativas das respetivas plataformas. *Tom Occhino, React.js Team Leader*, explica na *React.js Conference 2015 Keynote*, que o segredo está na forma como a biblioteca acede às funções nativas. Segundo *Occhino*, o que possibilita a fluidez das aplicações é a implementação de uma *thread* que "corre" concorrentemente à *thread* principal e que funciona como o meio de comunicação entre a linguagem de *Javascript* e a linguagem nativa da respetiva plataforma. Esta *thread* possui a capacidade de trocar mensagens com a *thread* principal e por isso não ocorrem atrasos na renderização dos componentes, pois a *thread* principal mantém-se liberta para o processamento normal, i.e., o mesmo processamento realizado numa aplicação nativamente construída (Developers, 2015). Tal está ilustrado na Figura 4.

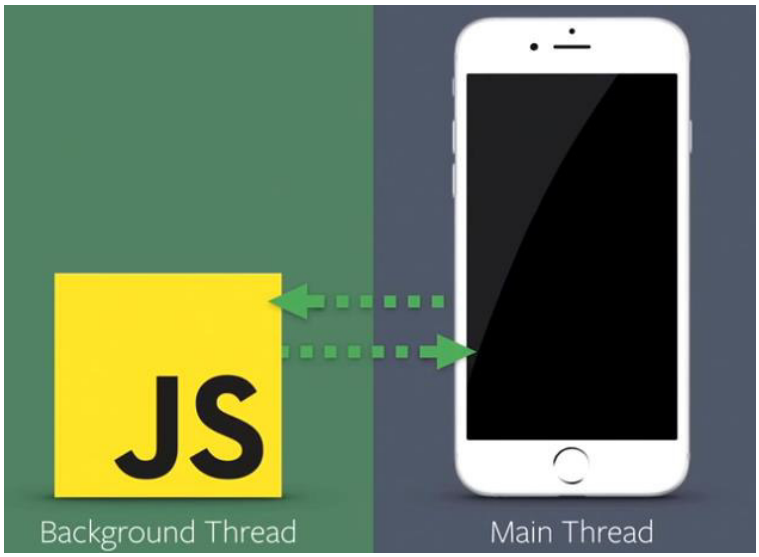


Figura 4: *Javascript thread* e *thread* principal (*React Native*) (Developers, 2015).

Optou-se pela adoção da *React Native* para a geração de aplicações *mobile* devido à semelhança com *React.js* no que diz respeito à manipulação de componentes e objetos. No entanto, o aspeto mais importante para a decisão diz respeito ao produto final que é possível alcançar, i.e., aplicações nativas. Serve de ressalva que, devido a limitações do tempo disponível para o desenvolvimento da aplicação capaz de gerar aplicativos, a geração de aplicações nativas é identificada como trabalho futuro.

Javascript no servidor

Adotou-se também a *framework Node.js*. Trata-se de uma *framework* orientada a eventos assíncronos (*event-driven*) (Joyent, 2015) cujo funcionamento assenta numa *event-loop*, em que todos os eventos registados (no código, explicitamente) são testados, i.e., é verificada a sua ocorrência e, após essa captação, a sua execução é enviada para uma fila de execução (*execution stack*). Esta abordagem permite uma melhoria substancial no desempenho das aplicações, pois embora todo o processo ocorra em apenas uma *thread* por servidor *Node.js*, dificilmente a aplicação experiencia travagens ou bloqueios. Para além do referido, a replicação de servidores é extremamente simples de implementar, pois apenas é necessário iniciar uma nova instância, atribuir numa nova porta na qual o servidor irá receber os "pedidos" e, toda esta lógica pode ser efetuada em poucas linhas de código. Outro aspeto interessante é que funciona sobre a linguagem de programação *Javascript*, o que permite que a maior parte do código que é executado no servidor *Node.js*, funcione também em ambiente cliente (o *browser*). Permite ainda a criação de instâncias de servidores, possibilitando o pré-processamento de tarefas diretamente no servidor. Um exemplo

prático é que, podemos executar *Javascript* no servidor (através do *Node.js*), esta execução poderá gerar HTML que é enviado para o *browser*, mas a grande vantagem é que este HTML (enviado para o *browser*) poderá conter código de *Javascript*, e este pode ser também utilizado para executar no *browser*, i.e., este processo permite que todo este código de *Javascript* possa ser mantido apenas no servidor, mas utilizado em ambos (servidor e *browser*),sem a necessidade de utilizar outra linguagem. Esta abordagem possibilita a utilização de apenas uma linguagem de programação no servidor e no cliente (*browser*), o que permite centralizar e otimizar a gestão do código-fonte.

No presente trabalho o *Node.js* v0.12.0 irá ser utilizado para gerar os componentes *React.js* no servidor. Desta forma, será possível pré-processar parte do HTML no servidor (HTML dos componentes *React.js*), libertando o *browser* da execução de uma grande parte do *Javascript* e respetivas atualizações de DOM, o que permite obter aplicações mais fluídas e com melhores índices de desempenho.

Na Figura 5 apresenta-se um exemplo da implementação de um servidor em *Node.js*.

```
var http = require('http');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, "127.0.0.1");

console.log('Server running at http://127.0.0.1:1337/');
```

Figura 5: Exemplo de criação de servidor (*Node.js*) (Joyent, 2015).

Tratamento do aspeto da aplicação proposta e aplicações geradas

A *framework Bootstrap* (@fat) foi adotada para solucionar questões relativas à facilidade de implementação e qualidade da apresentação e usabilidade da aplicação proposta. A *Bootstrap* é uma *framework* de HTML, CSS (*Cascading Style Sheets*) e *Javascript*, destinada ao desenvolvimento de aplicações *Web* com aspeto responsivo. Originalmente criado por *Mark Otto* e *Jacob Thornton* que trabalhavam no *Twitter*, a *framework* seria disponibilizada como *open-source* em agosto de 2011, no *GitHub* (W3Schools.com). Esta *framework* permite um elevado nível de abstração relativamente à criação de elementos comuns a qualquer aplicação e a aspetos de UID/UXD (*User interface design/User experience design*), pois torna todo o *layout* responsivo, i.e., ajusta os componentes das páginas *Web* às medidas dos ecrãs dos vários dispositivos. Disponibiliza um

conjunto de componentes funcionais e com aspeto amigável, como: *dropdowns*; *quicksearchs*; *menu bars*; e vários outros. Assegura ainda a compatibilidade com os *browsers* mais recentes, como: *Chrome*; *Firefox*; *Internet Explorer*; *Safari;* e *Opera*.

A *Bootstrap framework* foi utilizada no presente trabalho, pois permite agilizar e acelerar o desenvolvimento de UID/UXD. Permite ainda a automatização e redimensionamento de *layouts* às diversas plataformas: telemóveis; *tablets*; e *desktops*.

Sistema de controlo de versões de ficheiros

No que concerne ao sistema de controlo de versões de ficheiros optou-se pela utilização do *Git*. Este é um sistema distribuído de controlo de versões de ficheiros que permite trabalho colaborativo, facilita a partilha de projetos entre dispositivos e, quando utilizado com recurso a um servidor *Git*, assume um papel importante para assegurar a replicação/redundância dos dados. Foi originalmente projetado e desenvolvido por *Linus Torvalds* em 2005, para ajudar na gestão de versões de ficheiros aquando do desenvolvimento do *Kernel Linux*. Posteriormente, foi adotado por muitos outros *players* da indústria de desenvolvimento de *software*, entre eles o *Facebook, Google, Microsoft, Twitter, Linkedin* e muitos outros (Team, 2015). Este sistema, assenta numa arquitetura de estágios de dados independentes, sendo estes a: *Working Copy*; *Staging Area;* e o *Local Repository*.

A *Working Copy* é o espaço de trabalho individual, quer isto dizer que todos os ficheiros são primeiramente adicionados e modificados neste estágio. O estágio de *Staging* serve de recurso intermédio, i.e., quando os ficheiros são modificados ou adicionados só passam a estar no sistema de gestão de versões se se encontrarem neste estágio. Este estágio intermédio permite reverter alterações indesejadas ou simplesmente assegurar o versionamento de ficheiros. O *Local Repository* é o estágio que recebe o *commit* dos ficheiros alterados, e que normalmente representa o estado mais atual do trabalho desenvolvido e finalizado. É ainda possível enviar os ficheiros para um servidor remoto *Git*. Esta operação pode ser alcançada através do comando *push*. Por outro lado o comando *pull* (ou *fetch* + *merge*), permite obter alterações de um repositório remoto. Neste sentido o servidor remoto pode ser visto como um quarto estágio em que o trabalho colaborativo realmente acontece.

O *Git*, possuí diversas funcionalidades úteis, nomeadamente a *stash*. A *stash* no *Git* é um "contentor" de trabalhos inacabados que possibilita que estes possam lá ser colocados e mais tarde obtidos. Esta é bastante útil em situações em que é urgente fazer algum *commit* sem enviar ficheiros

pendentes ou sob modificação, evitando assim a colocação de trabalho em progresso em servidores de produção.

Na figura 6 apresentam-se os estágios e respetivo processo de versionamento do *Git*.

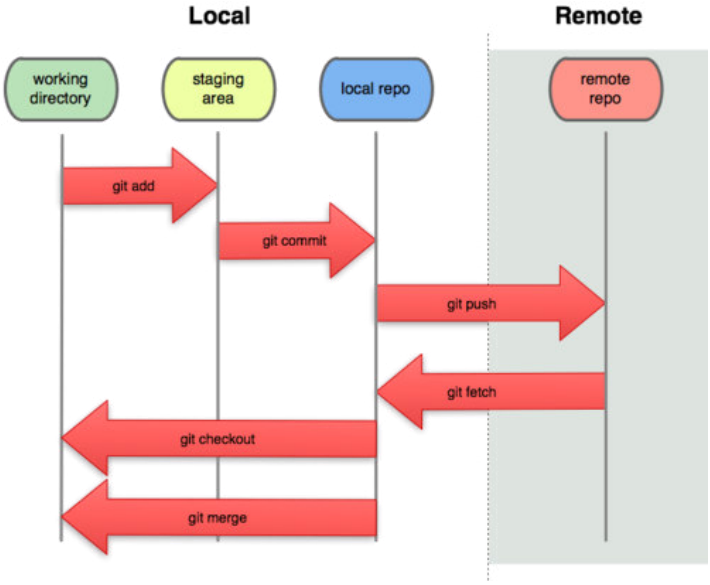


Figura 6: fluxo de interação *Git* (Franco, 2015).

Optou-se pela utilização do *Git* para o presente trabalho, pois considerou-se ser a tecnologia mais adequada ao controlo de versão de ficheiros, devido à simplicidade de interação, disponibilização de *stash*, robustez dos repositórios criados, por se tratar de um dos sistemas de controlo de versão de ficheiros mais utilizado, devido à vasta documentação disponível e por possibilitar futura colaboração pública no projeto, por exemplo na plataforma *Github*. Durante o processo de desenvolvimento e até ao final do trabalho de dissertação o acesso ao servidor *Git,* que contém o código fonte da aplicação em desenvolvimento, foi privado, no entanto, este será aberto ao público após a apresentação da dissertação.

2.3. Aplicações semelhantes à proposta

Nesta secção são apresentadas as aplicações *OutSytems Plataform* e *Phreeze*, tendo sido a informação obtida nos respetivos *websites*.

OutSystems Plataform

A *OutSystems Platform* é uma plataforma que foi desenvolvida pela *OutSystems* e que se encontra presente em 25 países prestando os seus serviços em 22 indústrias, contando já com cerca de 90.000 aplicações em produção. A plataforma é bastante automatizada e trata de

praticamente todos os aspetos relativos ao desenvolvimento de sistemas informáticos. Utiliza um motor de *Business Process Management* (BPM) que permite desenhar todo o processo de negócio a implementar na aplicação, suporta integração com *Oracle* e SQL *Server*, disponibiliza serviços de agendamento de rotinas, integra vários sistemas de versão de ficheiros, promovendo o trabalho colaborativo, possui um sistema de *rollback* que permite retomar estados anteriores da aplicação, disponibiliza tecnologia de alta escalabilidade o que permite escalar as aplicações, possibilitando, por exemplo, a adição de mais servidores a um dado balanceador, é extensível, pois permite a adição de código-fonte .NET ou Java que poderá ser conjugado com os vários elementos da plataforma, i.e., o código adicionado possibilita a exposição das suas funções que podem ser utilizadas para manipular ou interagir com os elementos da plataforma. É auto documental e todo o código é gerado em Java ou .NET e, ainda oferece uma funcionalidade designada por *detach*, que permite a extração de todo o código-fonte necessário para a execução das aplicação criadas através da plataforma. Esta última funcionalidade só é disponibilizada quando o cliente *OutSystems*, por algum motivo, pretende cessar o contrato com a *OutSystems* e pretende que as aplicações desenvolvidas possam ser utilizadas externamente.

O objetivo da *OutSystems Platform* é a geração de aplicações de *software* para as várias plataformas (*Mobile*, *Web* e *Desktop*) através da especificação do sistema informático. No entanto, podem-se observar algumas questões inerentes a toda esta automatização. O primeiro problema que se coloca é que, dificilmente uma organização com um elevado volume de código-fonte já produzido e, com diversas aplicações em ambiente produtivo, possa integrar todo este trabalho numa plataforma como esta, pois a complexidade desta integração é com certeza superior à geração de todo código partindo do zero, utilizando a plataforma *OutSystems*. Para além disto, a plataforma apenas permite a geração de código-fonte para as linguagens de programação .NET e Java e, no que diz respeito a aplicativos desenvolvidos para o ambiente *Web*, linguagens como PHP, *Node.js* (*Javascript*), *Ruby* e *Ruby on Rails*, entre outras, representam uma grande maioria das aplicações presentes na *Web* e estas não usufruem da possibilidade de integração com a plataforma *OutSystems*. Outro aspeto a ponderar é que a *OutSystems Plataform* é bastante dispendiosa e por isso torna-a menos acessível a pequenas e médias organizações (Hinkle, 2013). No *website* da plataforma verifica-se que, uma subscrição básica para empresas, começa nos 1.650$/mês, para além de que o valor final é determinado pela capacidade contratada da plataforma, do espaço em disco, da capacidade de processamento contratada e do número de utilizadores a incluir por conta (OutSystems, 2015).

Aplicação *Phreeze*

A segunda aplicação que se apresenta de seguida é a aplicação *Phreeze* (Hinkle, 2013). Desenvolvida por *Jason Hinkle* e publicada em 2013, é uma aplicação *open-source* e que permite a criação de aplicações através da descrição das mesmas, no entanto esta descrição é realizada através do preenchimento de campos que, de uma forma mais ou menos explícita, representam o modelo de dados que suportará a aplicação a ser gerada. Faz uso de ORM e disponibiliza de imediato uma *RESTful* API gerada dinamicamente por cada aplicação desenvolvida. Tanto a aplicação *Phreeze* como as aplicação geradas possuem um aspeto bastante atualizado e amigável, graças à utilização da *Bootstrap framework*.

Relativamente a aspetos menos positivos da aplicação *Phreeze*, de referir: o facto de não ser possível descrever visualmente a especificação das aplicações a gerar, não recorrendo, por exemplo, a diagramas da UML ou BPMN. Por este motivo observa-se que as aplicações geradas tendem a ser mais complexas, menos previsíveis e essencialmente mais difíceis de manter. Por outro lado, a opção da criação de uma pequena *framework* para suportar o padrão MVC pode ser considerada um risco, pois normalmente as organizações que produzem *software* adotam *frameworks* que garantam um elevado nível de maturidade do código, pois necessitam que haja uma constante atualização de documentação da *framework*, desejam garantia de suporte técnico e anseiam uma elevada comunidade envolvente e ativa que permita ajudar na resolução de problemas relativos à *framework* e que contribua para a mesma através do desenvolvimento de novas funcionalidades, *plugins*, etc.

3. Modelo ISDocPal

No presente capítulo descreve-se o modelo da ISDocPal, bem como a sua arquitetura lógica.

3.1. Aplicação proposta

Após a pesquisa de soluções disponíveis para o desenvolvimento de *software* no contexto de SI, identificaram-se várias aspetos importantes, os quais devem estar presentes na aplicação resultante do presente trabalho, no entanto, também se identificaram alguns aspetos menos positivos que se pretendem colmatar. Neste sentido pretende-se que a aplicação proposta permita, numa primeira fase, descrever parcialmente o SI, contudo, para o presente trabalho e, por uma questão de limitação de tempo para a sua execução, a aplicação deve estar focada na descrição do modelo de dados e na definição dos casos de uso da aplicação a gerar. Para tal, são utilizados o diagrama de base de dados e o diagrama de casos-de-uso. Deve ainda gerar código estruturado segundo o padrão MVC e esta estruturação deve assentar em *frameworks* amplamente utilizadas. Para o presente trabalho optou-se pela geração de código-fonte para a Yii2 *framework*. No entanto, a aplicação assenta numa arquitetura que permite a geração de código-fonte para qualquer linguagem de programação, tornando-a mais abrangente e facilitando a integração com projetos existentes. Por fim, pretende-se, no final desta dissertação, disponibilizar todo o código fonte através de um projeto *open-source*.

À aplicação desenvolvida no presente trabalho de dissertação foi atribuído o nome ISDocPal. O nome ISDocPal é a abreviatura de *Information System Documentation Pal* e surge por se considerar que a aplicação pretende efetivamente ser uma "amiga" da documentação, no contexto dos SI, tendo como o objetivo principal a documentação dos sistemas informáticos *à priori* do seu processo de desenvolvimento.

Esta aplicação deve ter a capacidade de gerar aplicações e respetivo código-fonte, de forma automatizada e a partir de especificações de UML. As aplicações geradas devem ser funcionais, no entanto, deve ser possível realizar modificações ao código-fonte após a sua geração.

A ISDocPal introduz uma nova abordagem para o desenvolvimento de aplicações que se definiu como "Describe once, run anywhere, modify". Esta abordagem é a sugestão de uma nova perspetiva orientada para o desenvolvimento de sistemas informáticos e que surge no decorrer deste desenvolvimento do trabalho de dissertação. Não se pretende com a abordagem mencionada

contrapor nenhuma outra abordagem existente, como: “Learn once, deploy anywhere” (Developers, 2015).

A abordagem “Describe once, run anywhere, modify” assenta em três momentos distintos de processo de desenvolvimento de *software*.

- “Describe once”: fase de descrição ou representação do sistema de informático. No presente trabalho, a aplicação ISDocPal utiliza especificações UML para a representação dos sistemas informáticos. No entanto, a abordagem assume que a representação pode ser realizada com recurso a qualquer uma das técnicas que permita representar ou descrever sistemas informáticos. A abordagem conceptual inerente a esta fase é que seja possível descrever o sistema (ou parte deste) uma única vez e a camada tecnológica tratará de gerar todo o código fonte da aplicação;
- “Run anywhere”: nesta fase, é expectável que a aplicação anteriormente representada/descrita seja suportada em qualquer plataforma, i.e., em telemóvel; *tablet*; *desktop*; entre outras. Obviamente as diferentes plataformas exigem incrementos na aplicação ISDocPal. No entanto, e no contexto do presente trabalho, primeiramente foi tratada a componente *Web desktop* e *Web mobile* devido a limitações de tempo para a execução do mesmo;
- “Modify”: fase em que o código gerado é modificado ou é introduzido novo código na aplicação. Este código deve focar aspetos de otimização de UID/UXD, portanto não deverá consistir em alterações que anulem a representação do sistema anteriormente realizada sob pena de quebrar a lógica do processo até então realizado. Pode incidir também na implementação (em código-fonte) de regras de negócio que a aplicação ISDocPal não permitiu realizar.

Importa também referir o que a ISDocPal não é uma aplicação com a capacidade de gerar aplicações otimizadas para ambientes produtivos, i.e., as aplicações necessitarão de posteriores ajustes e desenvolvimentos, no entanto, esses ajustes estarão maioritariamente relacionados com apresentação e regras de negócio.

3.2. Arquitetura da aplicação ISDocPal

A aplicação ISDocPal encontra-se desenvolvida para ser utilizada em ambiente *Web*. A arquitetura encontra-se descrita na Figura 7 pelo que de seguida se detalha os seus componentes.

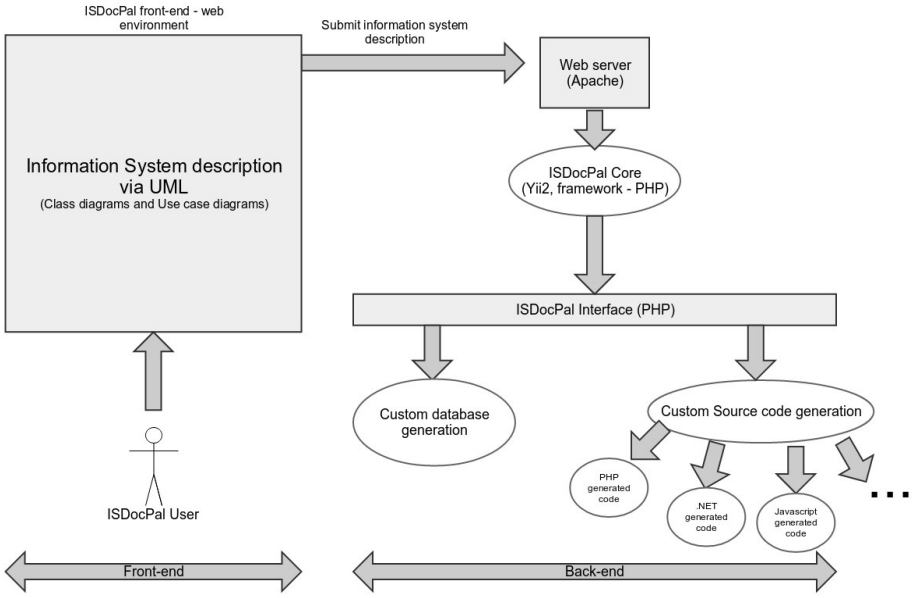


Figura 7: arquitetura da aplicação ISDocPal

O componente *Front-end* foi desenvolvido com recurso às tecnologias *HTML5*, *CSS3*, *Javascript*, *Bootstrap*, *ReactJS* e *JQuery*.

O objetivo deste componente é disponibilizar uma ferramenta que permita que o utilizador da ISDocPal descreva e especifique a aplicação ou sistema informático a ser gerado de forma amigável, abstraindo-se da complexidade dos detalhes de implementação, permitindo que o foco incida unicamente nessa mesma descrição de forma simples. Neste componente podem ser encontradas três áreas importantes para a construção da aplicação a gerar, nomeadamente a: *Entity relationship diagrams* (ERD) ou diagrama de base de dados; *Use cases*; e *Users*.

- A área ERD permite descrever a estrutura de dados que suportará a aplicação a gerar. Essencialmente permite identificar e relacionar as entidades intervenientes no sistema, sendo que estas serão criadas como tabelas na base de dados e automaticamente relacionadas no momento da criação. Esta área permite ainda listar, editar e eliminar todas as ERD previamente criadas.

- *Use cases* é área que diz respeito à criação/edição/consulta de casos de uso da aplicação a gerar. Neste sentido os casos de usos são descritos com recurso ao digrama de casos de uso segundo a UML.

- Na área *Users* podem ser criados utilizadores, definindo o seu perfil e respetivas permissões de acesso. Esta funcionalidade permite que o desenvolvedor da aplicação possa rapidamente definir os utilizadores com acesso à aplicação.

No componente *Back-end*, podem-se observar os subcomponentes: ISDocPal *Core*; ISDocPal *Interface*; *Custom Database Generation* (CDG); e *Custom Source Code Generation* (CSCG).

- ISDocPal *Core* é o "coração" da aplicação. Todos os estímulos exteriores são primeiramente absorvidos por este componente, de seguida interpretados e executados.

Em seguida descrevem-se os passos de execução que caracterizam o componente ISDocPal *Core*:

1. Receção e validação dos dados estruturados provenientes do componente *Front-end*;
2. Manipulação e simplificação dos dados recebidos;
3. "Canalização" da informação obtida no passo anterior para os objetos que implementem o ISDocPal *Interface* (processo imediatamente seguinte).

Importa referir que o ISDocPal *Core* assenta na Yii2 *framework*, neste sentido, o componente utiliza diversas funcionalidades que esta *framework* oferece, entre elas: o *Object-relational Mapping* (ORM); *Widgets support*; suporte a programação segundo o padrão MVC; entre outras funcionalidades comuns ao desenvolvimento de *software*.

O ISDocPal Core disponibiliza um *interface* que deve ser implementada quando se pretende gerar código-fonte para uma linguagem de programação que não seja ainda suportada pela aplicação. Neste trabalho a única implementação (implementação por omissão) foi a de geração de aplicações com código-fonte em PHP, estruturada para um projeto Yii2 *Framework*.

O componente ISDocPal *Interface* permite a geração de código-fonte para qualquer linguagem de programação e para tal fornece um conjunto de métodos. Alguns destes são obrigatórios e outros facultativos. Os métodos obrigatórios, como a própria designação indica, são de implementação obrigatória para que a geração de código ocorra corretamente e a aplicação ISDocPal atinja o seu objetivo elementar. Os métodos facultativos permitem melhorar alguns aspetos das aplicações geradas: o controlo de acesso dos utilizadores; validações de campos e atributos em formulários; entre outras funcionalidades.

Os métodos de implementação obrigatória presentes no componente ISDocPal *Interface* são:

- buildControllerSection($controllerName, $action, $data=[]): void

Este método permite gerar o código para o controlador (C do MVC) em questão, normalmente e na grande maioria das *frameworks* que implementam o padrão MVC, trata-se da geração de uma classe. Este método recebe o nome a atribuir ao controlador, o nome a atribuir à ação em questão e alguns parâmetros opcionais, essencialmente estes parâmetros opcionais dizem respeito a algumas indicações relativas ao aspeto e à usabilidade da página em questão. Importa referir que

os parâmetros formais presentes neste método ou outro qualquer que faça parte do ISDocPal *interface* são "injetados" pelo ISDocPal *Core*.

- buildModelsSection($modelsData, $extra=[]): array

Este método será responsável por gerar todo o código fonte relativo aos modelos (M do MVC) e respetivas tabelas da base de dados, fazendo uma associação direta entre modelos e tabelas de base de dados. Conforme mencionado anteriormente a aplicação ISDocPal faz uso do ORM e por esse motivo os modelos encontram-se, na sua maioria, relacionados com uma tabela de base de dados. Este método recebe como parâmetros formais a informação dos modelos, i.e., o seu nome; relação com outros modelos; campos de base de dados e tipos de valores; etc. Após a sua execução retorna uma lista (*Array*) com informação relativa à ocorrência ou não de erros e respetivas mensagens.

- buildViewSection($controllerName, $actionName, $data=[]): void

Finalmente, o método *buildViewSection* permite a geração da *View* (V do MVC) associada à *Action* anteriormente gerada dentro da classe representativa do *Controller*. Este método deve ser invocado no corpo do método *buildControllerSection,* pois é este método que possui os valores do nome do controlador e nome de ação, essenciais para a geração da pasta e ficheiro da *View*.

Os métodos facultativos são os seguintes:

- buildRoleAndPermission($roleData, $activityData): \yii\rbac\Role

Este método é responsável pela criação e ativação dos "papéis" e permissões dos atores descritos nos casos de uso. Retorna um objeto do tipo *\yii\rbac\Role*. Importa referir que este método apenas cria os "papéis" e respetivas permissões, ou seja, ele não cria as instâncias de utilizadores.

- buildAccessControll($accessControllData): mixed

Permite gerar o controlo de acesso ao nível dos controladores. Basicamente gera código que permite verificar se determinado utilizador possui nível de acesso para aceder a determinado recurso da aplicação, e fá-lo ao nível do controlador. Este método pode retornar o valor booleano *true* se não houver qualquer erro ou retorna um *Array* com os erros ocorridos no momento da geração do código. Novamente se refere que, este método não cria instâncias de utilizadores, apenas cria código que permite aferir se determinado utilizador possui nível de acesso suficiente para poder aceder a determinado recurso do sistema.

- buildComponentsForActivity($data): array

Este método é responsável pela geração do código-fonte de um determinado componente, essencialmente gera HTML. Estes componentes são tipicamente tabelas, formulários e *widgets* de qualquer tipo que permitam uma melhor interação do utilizador com a aplicação gerada. Este método recebe um parâmetro $data obtido do método *buildControllerSection* e por esse motivo deve ser invocado no corpo deste mesmo. Tem como retorno uma lista de possíveis erros ocorridos, sua descrição e código.

- Métodos performCommonTasksBeforeBuild() e performCommonTasksAfterBuild():

Estes métodos permitem que o utilizador possa executar um conjunto de ações antes e depois, respetivamente, da geração da aplicação. O objetivo da criação destes métodos deve-se ao reconhecimento de que existem diversas especificidades relativas às diversas linguagens de programação e respetivo código-fonte a gerar e pretende-se com a sua implementação possibilitar a implementação dessas especificidades tornando a aplicação ISDocPal mais abrangente.

Na Figura 8 apresenta-se a organização dos diversos métodos da componente ISDocPal *Interface*.

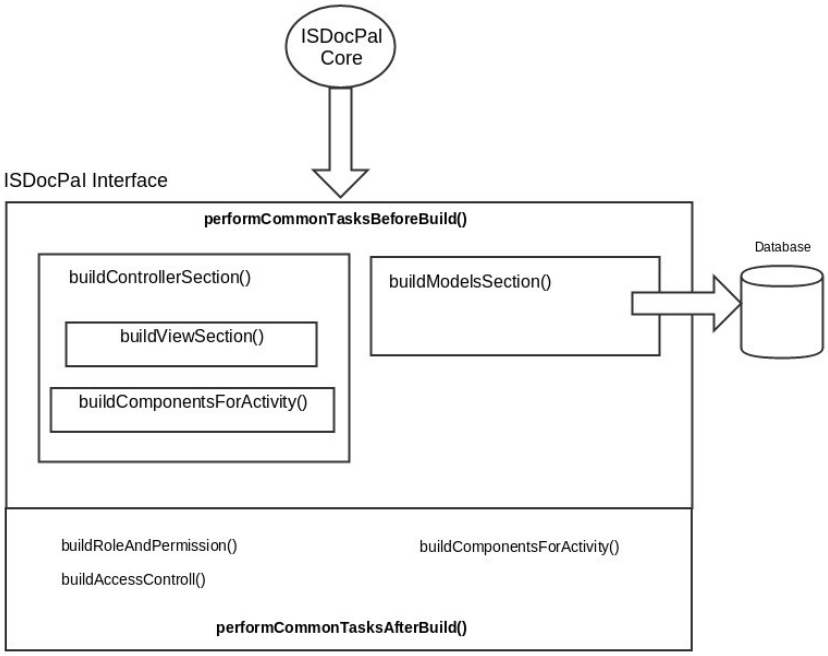


Figura 8: organização lógica do ISDocPal *Interface*.

Por fim, o CDG e CSCG são componentes dinâmicos, i.e., estes componentes são tipicamente criados pelo programador por forma a permitir a geração de código-fonte segundo as especificidades de cada uma das linguagens de programação. O ISDocPal *Core* disponibiliza diversas ferramentas para a geração de código fonte para PHP como a geração de classes, métodos, funções, variáveis

e operações de controlo. Para além destes aspetos, e após a apresentação do ISDocPal *Interface*, fica claro que a aplicação ISDocPal assume que a geração do código-fonte irá ser orientada à estruturação do código segundo o padrão MVC. A ISDocPal *Core* possui também a capacidade de gerar tabelas de base de dados para *MySQL*, *SQL Server*, *SQLite* e *PostgreSQL*.

4. Aplicação ISDocPal

No presente capítulo é descrito em detalhe o processo Scrum definido para o desenvolvimento da aplicação proposta.

4.1. Processo de desenvolvimento

No presente trabalho optou-se pela metodologia Scrum (Sutherland, 2013) para o desenvolvimento de *software*. Para tal definiu-se o processo que se descreve de seguida e que se apresenta Figura 9.

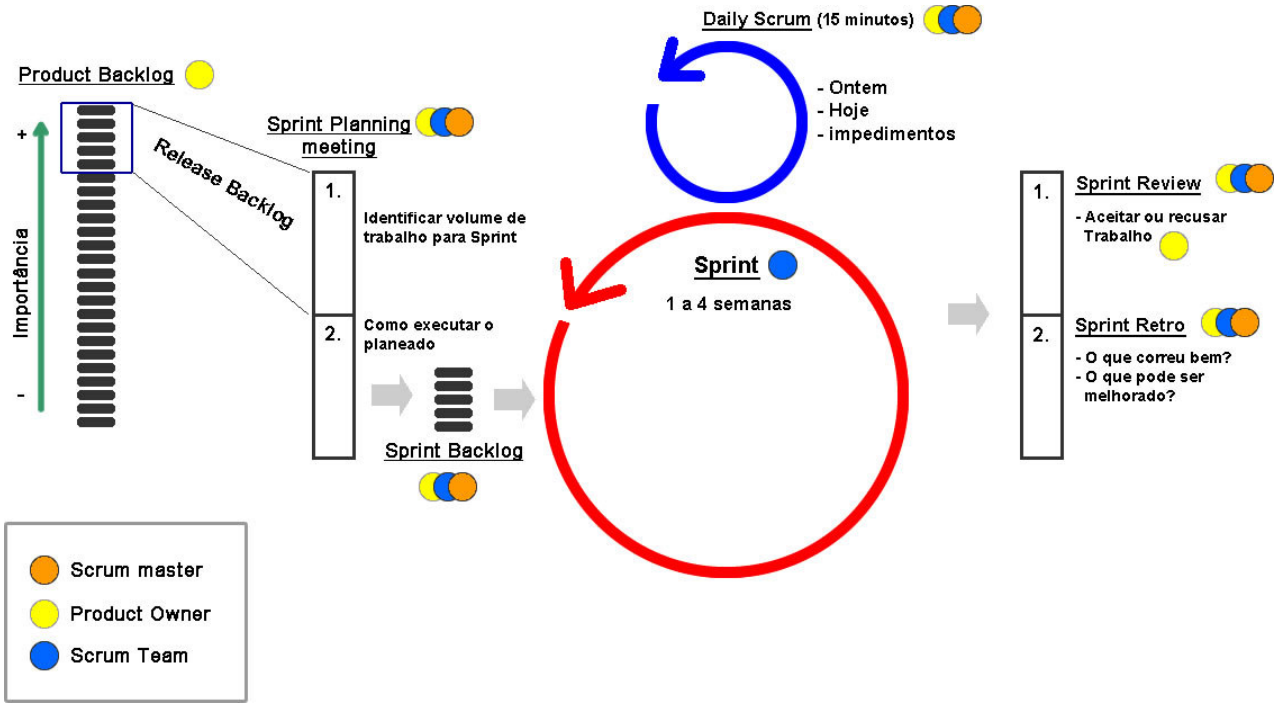


Figura 9: processo de desenvolvimento Scrum definido.

O processo adotado para o desenvolvimento da aplicação ISDocPal é o que se encontra presente na Figura 9. De seguida, descreve-se passo-a-passo como este foi implementado na prática.

O Scrum *Master* definiu o *Product Backlog*, organizando as *User stories* por ordem de prioridade decrescente. De seguida foram identificados os *Release Backlogs* e cada um destes foi particionado em um ou mais *Sprints*. Após a definição dos *Sprints Backlogs*, cada *Sprint* foi iniciado, tendo uma duração entre uma e quatro semanas. No decorrer de cada *Sprint* definiu-se a realização de um *Daily* Scrum (15 minutos) que permitiu identificar o trabalho realizado no dia anterior, a

realizar no próprio dia e possíveis impedimentos (tais impedimentos foram tratados pelo Scrum *Master*). Após a conclusão de cada um dos *Sprints* definiu-se a realização de um *Sprint Review* e de um *Sprint Retrospective*. O primeiro, com a finalidade de aceitar (ou recusar) o trabalho realizado (decisão do *Product Owner*) e, o segundo, com a finalidade de analisar o que correu bem e o que correu menos bem, de modo a que estes erros não se repetissem em *Sprints* subsequentes.

O autor assumiu todos os papéis para o processo descrito. Na prática funcionou bastante bem. Foi possível diferenciar cada um dos papéis e foi assumida uma postura responsável para concretizar cada uma das atividades do processo. Relativamente à *Daily Meeting*, numa primeira perspetiva parece ser algo estranho, pelo facto de se tratar de uma reunião com apenas um participante, mas na verdade as questões que se colocaram nessa reunião foram extremamente importantes para refletir sobre o trabalho diário. Portanto passou essencialmente por um exercício de reflexão diário que permitiu alcançar objetivos diários.

4.2. Requisitos e especificações da aplicação ISDocPal

Nesta secção apresentam-se os requisitos e a especificação da aplicação proposta. Para tal, elaborou-se uma tabela de cruzamento de requisitos e um *workflow* do processo da utilização da aplicação ISDocPal.

Tabela 1: cruzamento de requisitos específicos *vs.* requisitos gerais da aplicação proposta.

Requisitos específicos *vs.* Requisitos gerais	Gerir Utilizadores	Gerir Entidades-Relacionamentos		Gerir Casos de Uso		
		Gerir diagramas	Gerir entidades	Gerir diagramas	Gerir atividades	Gerir atores
Criar utilizador da aplicação ISDocPal	X					
Criar utilizador aplicação gerada	X					
Criar diagrama ER		X				
Editar diagrama ER		X				
Submeter diagrama ER		X				
Eliminar diagrama ER		X				
Ler diagrama ER		X				
Listar diagramas ER		X				

Requisitos específicos *vs.* Requisitos gerais	Gerir Utilizadores	Gerir Entidades-Relacionamentos		Gerir Casos de Uso		
		Gerir diagramas	Gerir entidades	Gerir diagramas	Gerir atividades	Gerir atores
Gravar Diagrama ER	X					
Criar entidade			X			
Editar entidade			X			
Ler entidade			X			
Eliminar entidade			X			
Relacionar entidades			X			
Definir propriedades entidade			X			
Criar diagrama de casos de uso				X		
Editar diagrama de casos de uso				X		
Ler diagrama de casos de uso				X		
Eliminar diagrama de casos de uso				X		
Criar atividade					X	
Editar atividade					X	
Ler atividade					X	
Eliminar atividade					X	
Definir propriedades de atividade					X	
Criar ator						X
Editar ator						X
Ler ator						X
Eliminar ator						X
Definir propriedades de ator						X
Relacionar ator a atividade				X	X	X
Associar entidade a atividade			X	X	X	

Na Tabela 1 apresentam-se os requisitos da aplicação ISDocPal. Optou-se pela divisão dos requisitos em áreas gerais e respetivo cruzamento com requisitos mais específicos. É de assinalar que todos os requisitos definidos encontram-se implementados na aplicação final.

Para facilitar a compreensão do funcionamento da aplicação e seu processo de execução, foi realizado um diagrama / *workflow*. Tal, pode ser observado na Figura 10 que se apresenta de seguida.

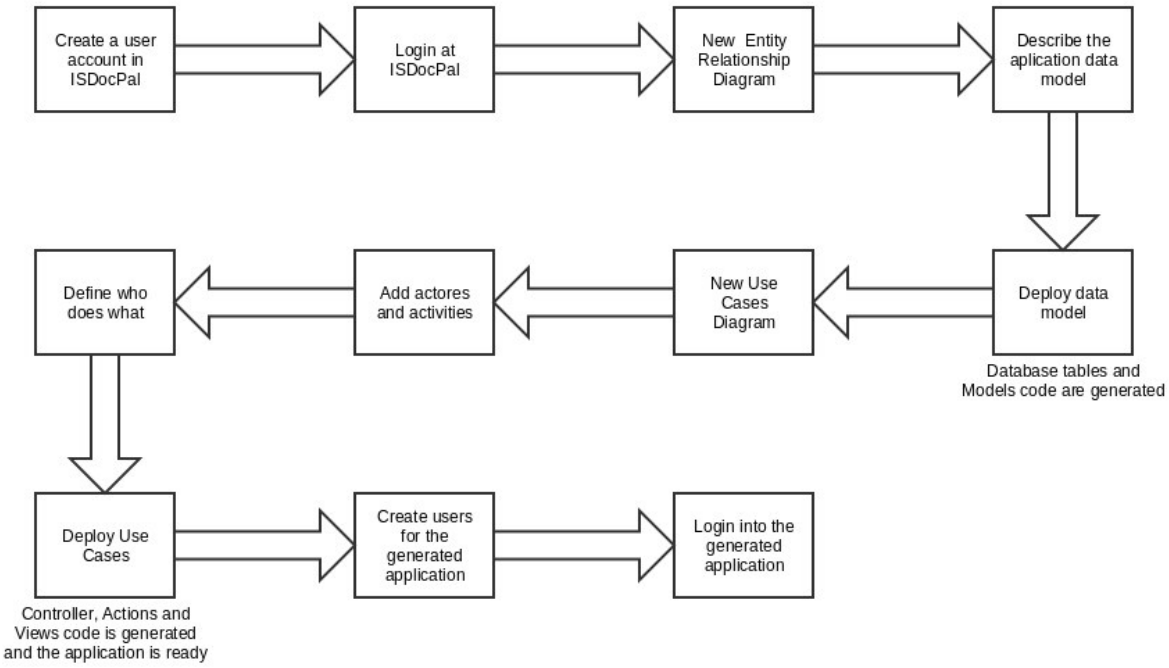


Figura 10: *workflow* processo desenvolvimento ISDocPal.

Através da Figura 10 pode-se verificar que o processo de desenvolvimento de uma aplicação com a ISDocPal se inicia criando uma conta de utilizador ISDocPal. Seguidamente é realizado o login na aplicação e avança-se para a criação de um diagrama de entidades-relacionamentos. Neste diagrama, o objetivo é descrever o modelo de dados que suportará a aplicação a gerar. Neste sentido, devem ser adicionadas todas as entidades, inserir todas as propriedades de cada entidade e realizar o respetivo relacionamento entre entidades. Para completar o processo de criação do modelo de dados é apenas necessário executar o *deploy* do modelo. Na prática, o *deploy*, permite gerar todo o código-fonte associado à gestão das respetivas entidades e permite ainda a geração das tabelas de base de dados que representam tais entidades. Após a finalização da definição do modelo de dados, em seguida, deve ser criado o diagrama de casos-de-uso. Este diagrama permite definir quais as atividades que a aplicação suportará e quem (atores) as poderá realizar. É ainda possível configurar quais os campos visíveis por omissão na aplicação gerada. Após a finalização do diagrama é necessário realizar o *deploy* do mesmo. E, neste caso, o *deploy*, permite gerar todo o código necessário para que a aplicação funcione, i.e., são gerados os controladores, ações, *Views*, código relativo a regras de acessos, etc. Por fim é apenas necessário criar o(s) utilizador(es) com os tipos de permissões anteriormente definidos nos casos-de-uso. Esta é uma

funcionalidade extra da aplicação ISDocPal que oferece um sistema de *Acess Control List* (ACL) por omissão, no entanto, o utilizador pode implementar o seu próprio ACL se assim o desejar. Após a criação dos utilizador(es) pode-se aceder à aplicação gerada e realizar o login com os dados do(s) utilizador(es) criado(s) no passo anterior, i.e., a aplicação encontra-se finalizada no que diz respeito à intervenção da ISDocPal.

4.3. Prova de conceito e discussão dos resultados obtidos

Nesta secção apresenta-se um exemplo de implementação da aplicação desenvolvida. Realizou-se também a discussão dos resultados obtidos.

4.3.1. Exemplo de uma implementação prática da aplicação ISDocPal

No trabalho de dissertação foi criado um caso de simulação prática da utilização da aplicação ISDocPal. Foi definido como objetivo a criação de uma aplicação para suporte a algumas atividades de um banco fictício. Nesse sentido, a aplicação deve suportar algumas operações básicas relativas à gestão de transações e empréstimos. Deverá ser possível que o cliente possa pedir empréstimos para diferentes finalidades e realizar transações diversas. Também é necessário que um responsável do banco tenha acesso ao sistema informático e possa gerir contas de cliente, contas bancárias, tipos de empréstimos e tipos de transações.

Importa referir que na simulação não será realizada qualquer alteração ao código-fonte gerado, essencialmente para que se possa apurar a eficácia da aplicação ISDocPal e também para que se possa refletir e ponderar melhorias futuras.

De seguida demonstra-se passo-a-passo todo o processo de desenvolvimento da aplicação bancária.

Conforme se verifica na figura 11 procedeu-se inicialmente ao registo do utilizador na aplicação ISDocPal.

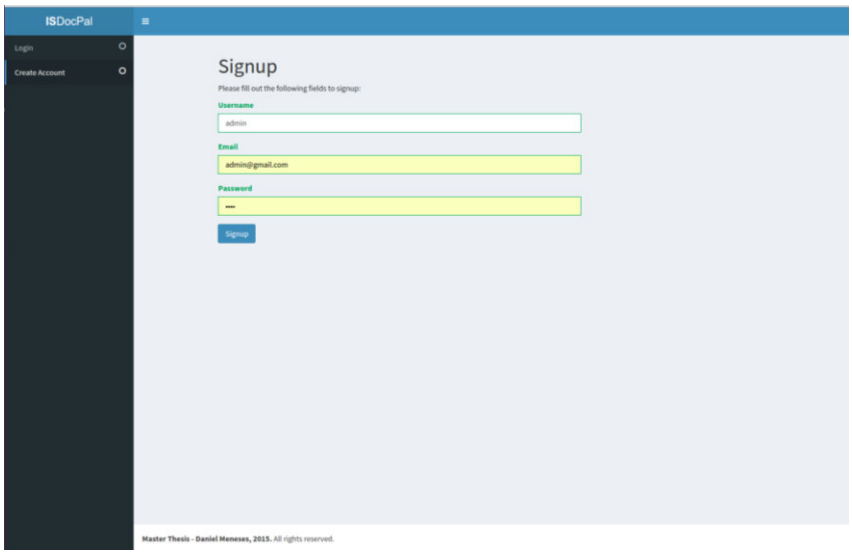


Figura 11: registo na aplicação ISDocPal

Após o registo procedeu-se ao respetivo *log-in*, sendo apresentada a informação presente na Figura 12.

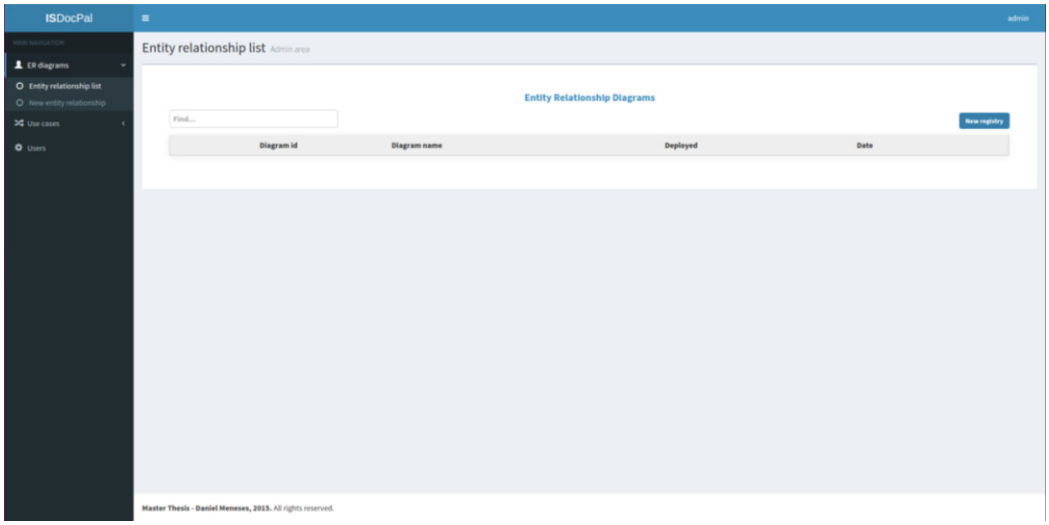


Figura 12: listagem de diagramas de base de dados

Após o *log-in*, a aplicação apresenta a área *Entity relationship list*. A partir desta área o utilizador poderá consultar todos os seus diagramas de entidades-relacionamentos, visualizá-los ou editá-los, ou poderá aceder a qualquer uma das restantes áreas através da barra de navegação lateral.

De seguida procedeu-se à criação do diagrama de entidades-relacionamentos que irá suportar toda a aplicação bancária. Na figura 13 é mostrado o ambiente de trabalho para realizar esta tarefa, assim como o editor de diagrama de base de dados.

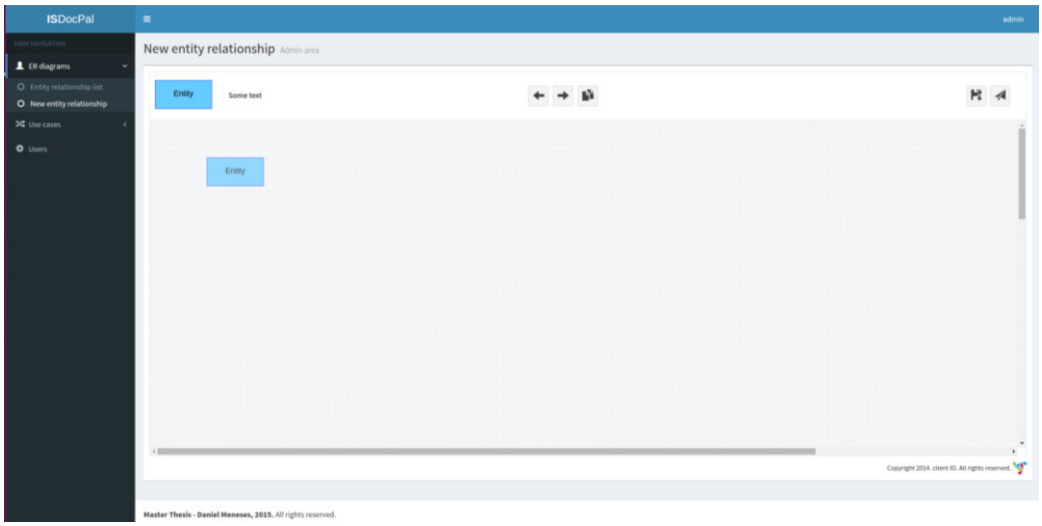


Figura 13: criação de diagrama de base de dados

A figura 14 mostra as propriedades de uma das entidades presentes no diagrama.

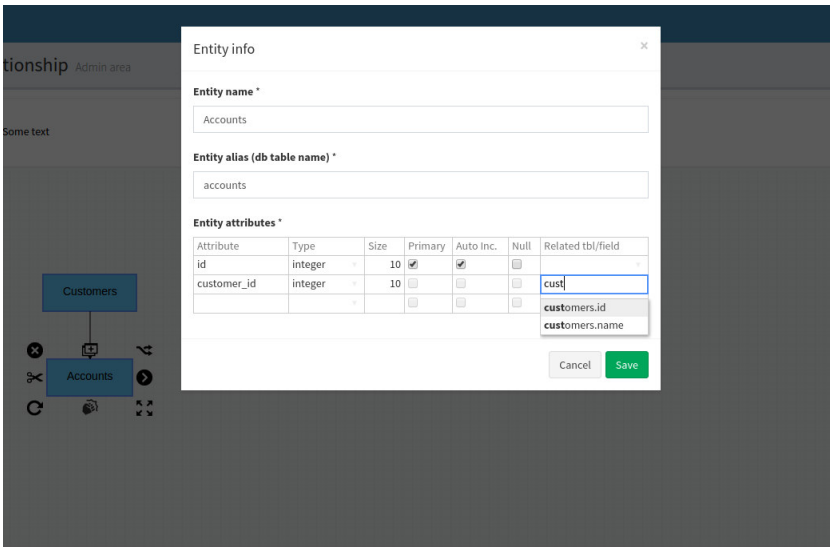


Figura 14: propriedades da entidade de base de dados.

No campo *Entity name* deve-se colocar o nome que se pretende atribuir à entidade, no campo *Entity alias* deve ser colocado o nome que será atribuído à tabela da base de dados que será gerada no momento do *deploy*. Na tabela *Entity attributes* devem ser incluídos todos os atributos ou campos que deverão ser associados à respetiva entidade que mais tarde serão associados à tabela de base de dados representativa desta mesma entidade. Conforme é visível na Figura 14, o campo *Related* tbl/ *field*, da propriedade *Entity attributes* permite relacionar a presente entidade com uma outra qualquer existente no diagrama.

Procedeu-se à criação das entidades necessárias e obteve-se o seguinte diagrama presente na Figura 15.

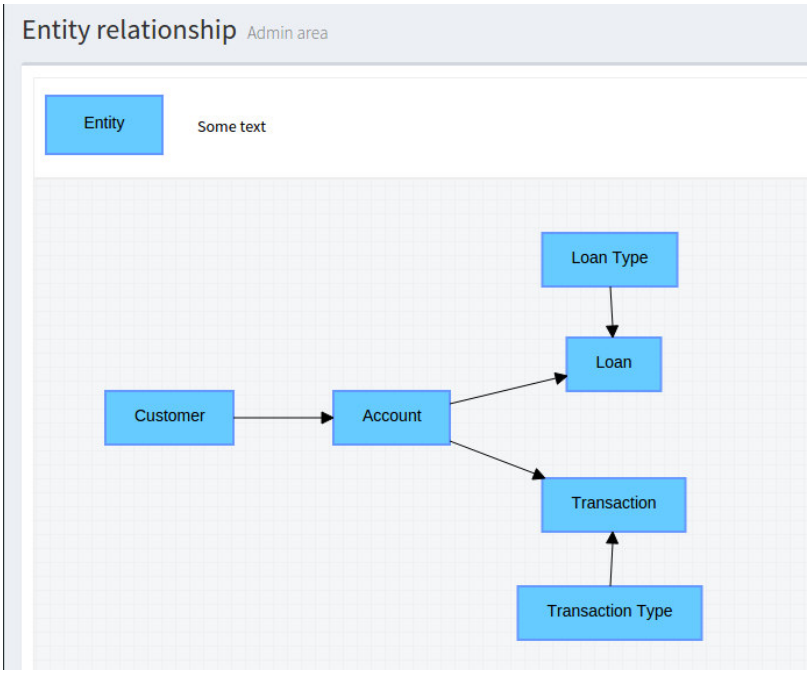


Figura 15: diagrama de base de dados final da aplicação bancária.

Importa referir que, na Figura 15, o sentido das setas estabelece o tipo de relacionamento, i.e., no exemplo, a entidade *Account* possui a chave-estrangeira que a relaciona com a entidade *Customer*. Neste sentido, pode-se observar que a aplicação irá permitir que cada cliente possa ter várias contas bancárias, mas cada conta corresponde apenas a um cliente. Por outro lado, cada conta pode realizar vários empréstimos e transações, mas cada empréstimo ou transação está relacionada com apenas uma conta bancária. É ainda possível a definição de tipos de empréstimos.

De seguida faz-se *deploy* do diagrama (Figura 16).

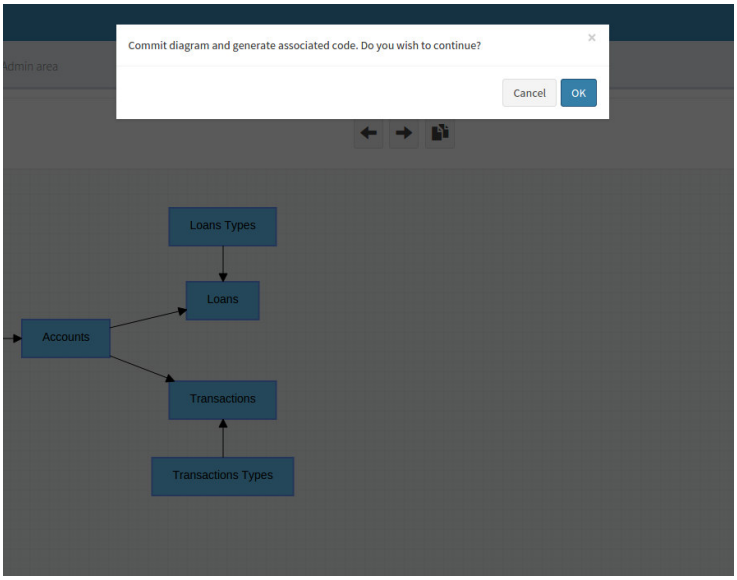


Figura 16: *deploy* do diagrama de base de dados.

Este *deploy* essencialmente permitirá gerar todo o código relativo aos modelos ou *Models* (M do MVC) e gerará todas as tabelas de base de dados. Importa referir que a base de dados para a qual o presente diagrama será gerado deverá já existir e deverá ser configurada no código fonte na pasta *common/config/main.php*.

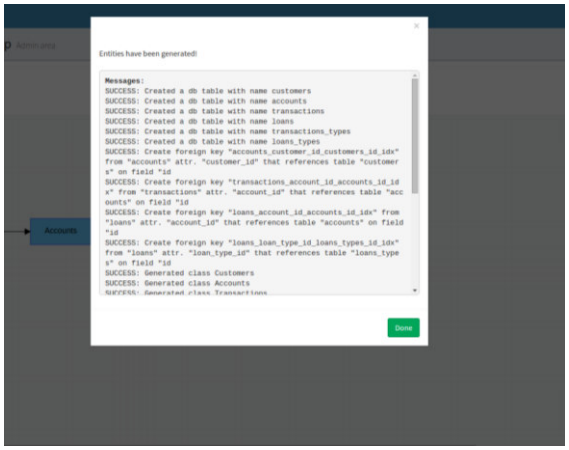


Figura 17: diagrama de base de dados gerado.

Após o *deploy* ser finalizado, a aplicação apresenta um resumo de todas as suas ações e resultados. Na Figura 17 verifica-se que foram criadas as tabelas *customers*, *accounts*, *transactions*, *loans*, *transactions_types* e *loans_types*, assim como todas as chaves estrangeiras dos relacionamentos definidos no diagrama. Pode-se ainda verificar a geração das classes associadas às mencionadas tabelas. Neste ponto a aplicação realizou cerca de metade do trabalho a que se propõe. Para algumas aplicações este pode ser o final da utilização da aplicação ISDocPal, pois neste momento já se possui toda lógica de relacionamento de entre as entidades na base de dados e um *ORM* consistente implementado na Yii2 *framework*.

Na Figura 18 pode-se verificar a existência das tabelas geradas e na Figura 19 pode-se confirmar a geração das classes *Models* associadas às respetivas tabelas.

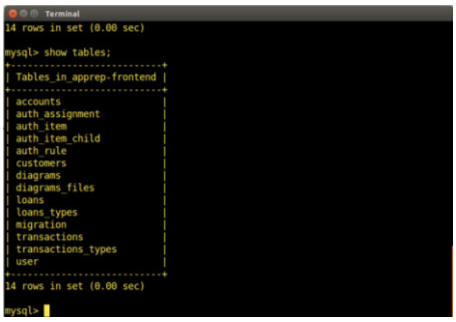


Figura 18: tabelas de base de dados geradas.

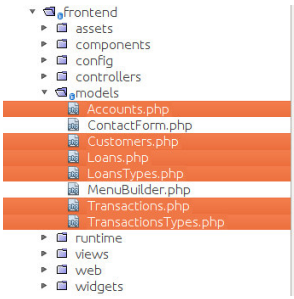


Figura 19: *Models* gerados (diagrama de base de dados).

De seguida, na Figura 20, apresenta-se o ambiente/editor de diagramas de casos-de-uso.

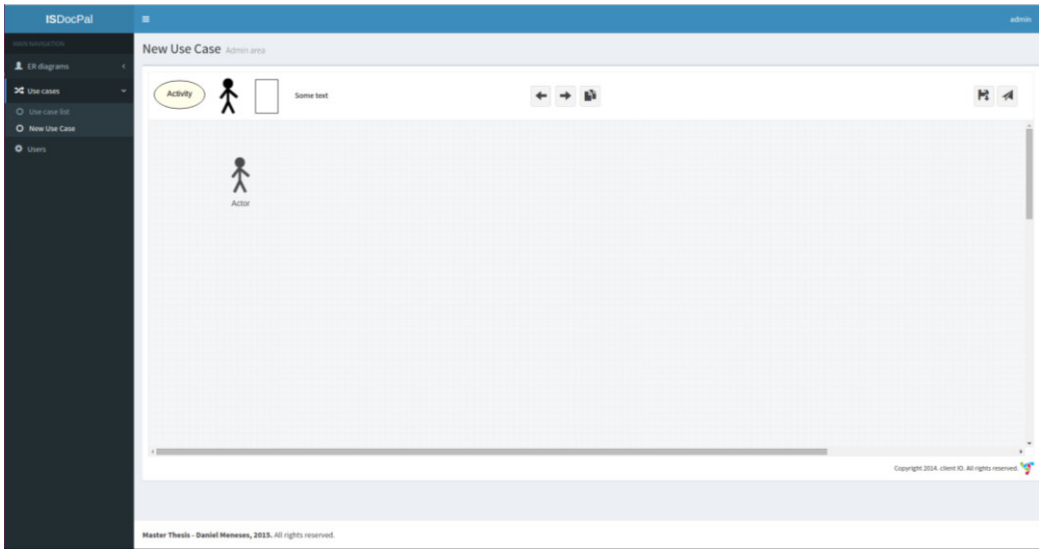


Figura 20: editor de casos de uso ISDocPal.

Conforme é possível observar na Figura 20 iniciou-se a criação do diagrama começando por criar um dos atores.

Na Figura 21 apresentam-se as propriedades de um ator dos casos-de-uso.

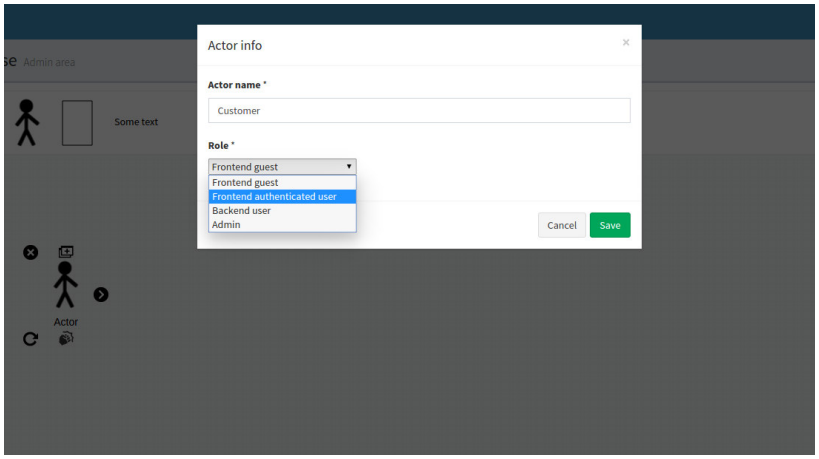


Figura 21: propriedades de Atores do diagrama de casos de uso.

Apenas é necessário atribuir um nome, no presente caso atribui-se o nome *Customer* e o *role* de *Frontend authenticated user*. A aplicação ISDocPal permite quatro tipos de *roles* ou papéis. O *Frontend guest* permite que qualquer utilizador possa interagir na atividade em questão; o *Frontend authenticated user* é um utilizador que pode aceder ao *Front-end*, mas apenas autenticado; o *Backend user* é um utilizador com autenticação para aceder ao *Back-end*; e, por fim, o *Admin* tem permissão para aceder a qualquer área da aplicação e realizar qualquer tipo de alteração.

Na figura 22 exemplifica-se a criação de uma atividade com o nome de *List transactions*, com o tipo de operação *List R*.

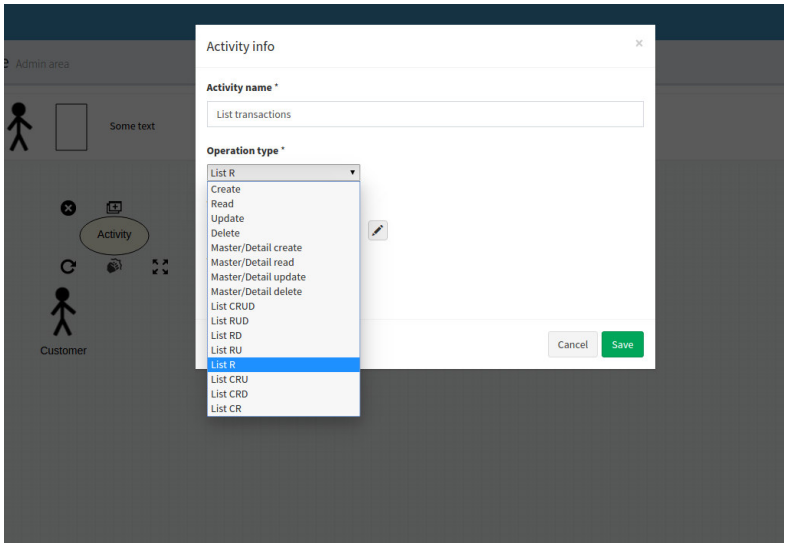


Figura 22: diagrama de casos de uso, tipos de operações.

Os tipos de operações foram divididos em conjuntos de *Create, Read, Update and Delete* (CRUD), i.e., CRUD por registo, CRUD por *master/detail* ou cabeçalho e linhas e CRUD de listagem de itens, ou seja, edição de vários registos diretamente numa listagem. No presente exemplo o objetivo é listar transações, por isso seleciona-se a opção *List R*, que basicamente quer dizer, listar e apenas visualizar (R = *Read*).

Na Figura 23 exemplifica-se como as entidades, previamente criadas, são associadas às atividades.

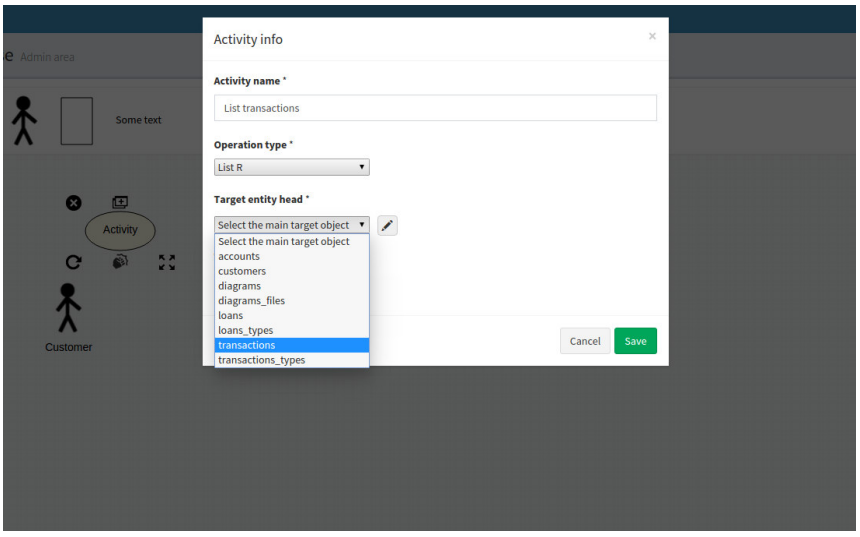


Figura 23: diagrama de casos de uso, *target entities*.

Existem dois campos para o fazer: *Target entity head* e *Target entity detail*. O primeiro permite identificar a entidade principal a manipular, i.e., seleciona-se a entidade ou tabela de base de dados *transactions*, pois o que se pretende é precisamente listar todas as transações (lista de *transactions*). Do lado direito do campo *Target entity head*, pode ser observado um botão com ícone de um lápis. Ao clicar neste botão surge uma outra janela (Figura 24) que permite configurar os campos a apresentar na aplicação a ser gerada.

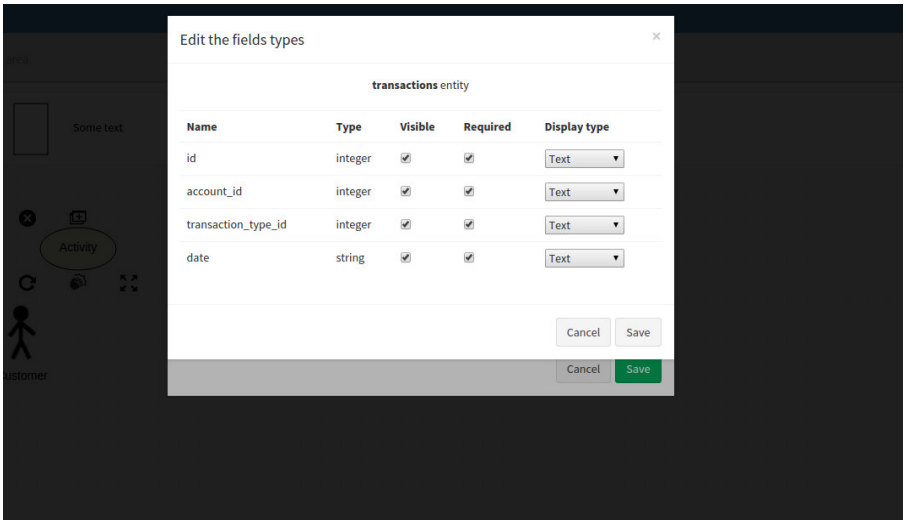


Figura 24: diagrama de casos de uso, configuração de campos.

Conforme se verifica na Figura 24 é possível alterar a visibilidade dos campos na aplicação a ser gerada. É ainda possível definir se o campo é de preenchimento obrigatório (se aplicável) e o tipo de dados a mostrar: *text*, *number*, *choice*, etc.

Após a criação de todas as atividades, atores e lógica de acessibilidade, obtivemos o seguinte diagrama de casos de uso:

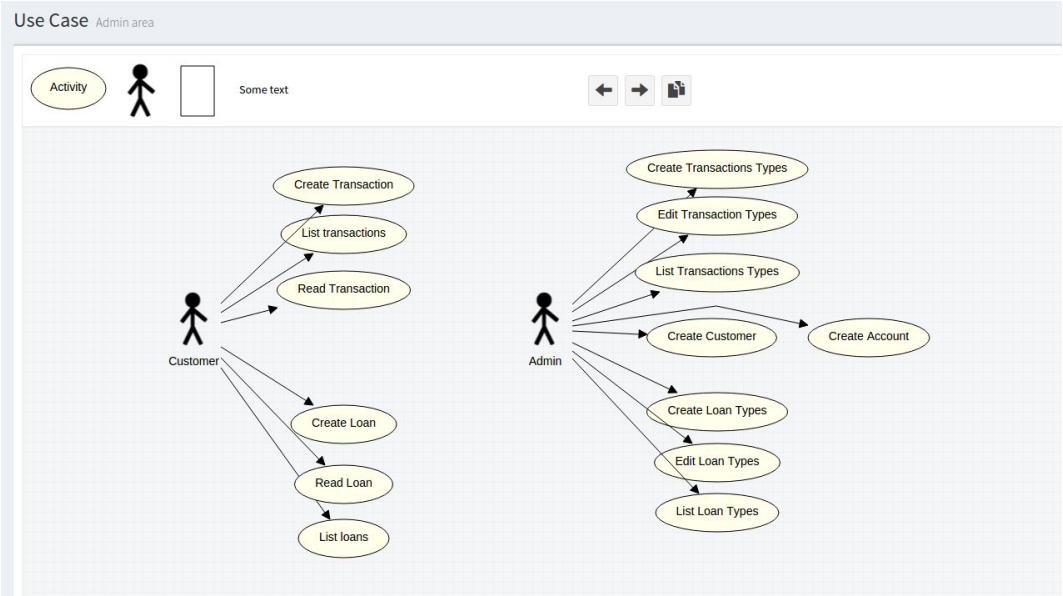


Figura 25: diagrama de casos de uso, modelo final.

Através da Figura 25 é possível perceber que o ator *Customer* terá acesso a criar transações, listar transações, ler transações, criar empréstimos (pedir), visualizar informação de empréstimos e listar empréstimos. Por outro lado, o ator *Admin* poderá criar, editar e listar tipos transações, criar clientes, criar contas bancárias e criar, editar e listar tipos de empréstimos.

Seguidamente procedeu-se ao *deploy* do diagrama de casos-de-uso elaborado, conforme se verifica na Figura 26.

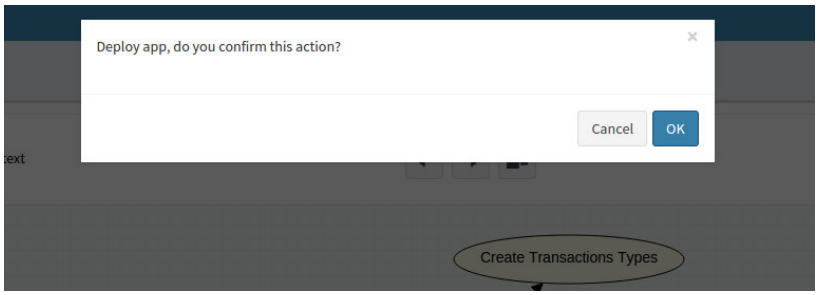


Figura 26: diagrama de caso de uso, *deploy*.

Na Figura 27 verificam-se as mensagens retornadas ao utilizador após a geração de código relativo ao diagrama de casos-de-uso.

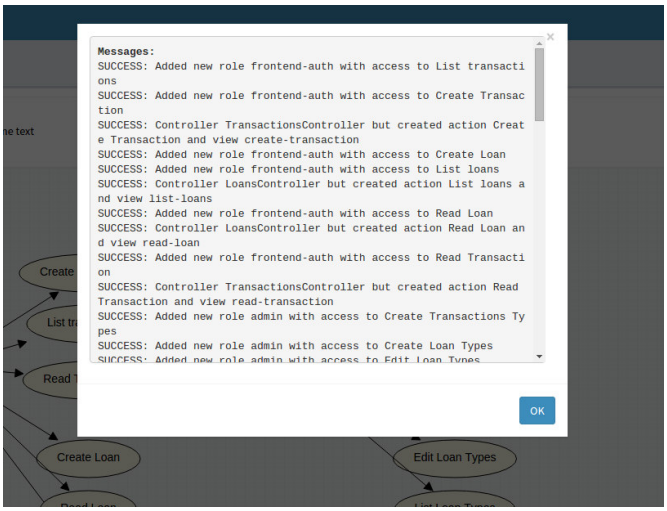


Figura 27: diagrama de caso de uso, finalizado.

Nesta fase a aplicação já se encontra completamente gerada e funcional, no entanto, falta ainda a criação de utilizadores com os tipos de permissões definidas nos atores do caso-de-uso. Para tal criou-se um utilizador com o perfil de *Admin* e outro utilizador com o perfil de *Frontend authenticated user*.

Na Figura 28 exemplifica-se a criação do utilizador *user* e na Figura 29 do utilizador *admin*.

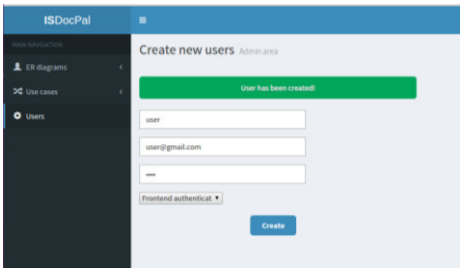


Figura 28: criação de utilizador *user*.

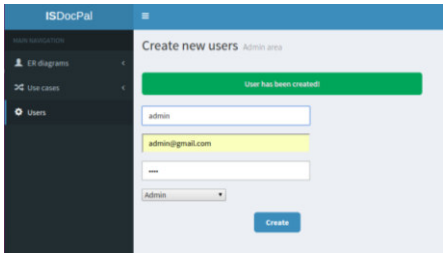


Figura 29: criação de utilizador *Admin*.

De seguida iniciou-se a aplicação gerada e é realizado o *log-in* com o perfil o utilizador *admin*, conforme visível na Figura 30.

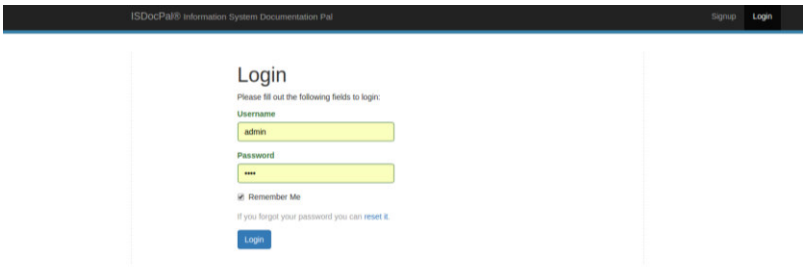


Figura 30: aplicação gerada, login.

Após o login, a aplicação apresenta a *home page*. Esta *home page* por omissão não tem qualquer conteúdo, por isso apenas irá mostrar a barra de navegação lateral com todas as funcionalidades a que o utilizador em questão tem acesso. Tal pode ser observado na Figura 31.

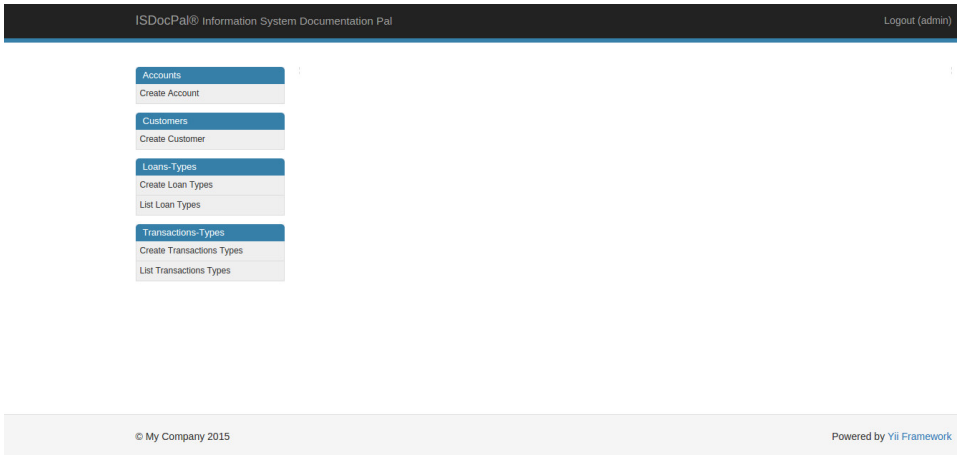


Figura 31: aplicação gerada, *home page*.

Na Figura 31 pode-se verificar ainda que a barra de navegação lateral possui os *links* para todas as atividades definidas no diagrama de casos-de-uso para o ator *Admin*. Apenas não estão visíveis os *links* de edição, pois a edição apenas se aplica a registos específicos.

De seguida procedeu-se à criação de clientes do banco fictício. Na Figura 32 esta ação pode ser verificada.

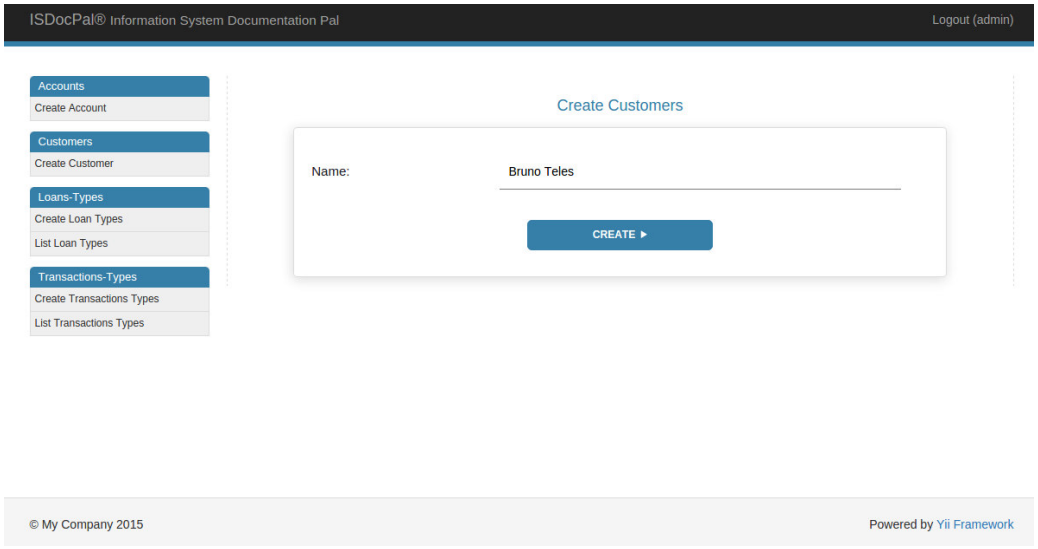


Figura 32: aplicação gerada, criação de cliente bancário.

Seguidamente criaram-se alguns registos como a criação de contas bancárias associando-as a um dado cliente bancário (Figura 33). Criaram-se ainda alguns tipos de transações e alguns tipos de empréstimos. A criação destes registos permitem, mais à frente, obter alguma informação básica para a criação de empréstimos e transações com a conta *user*.

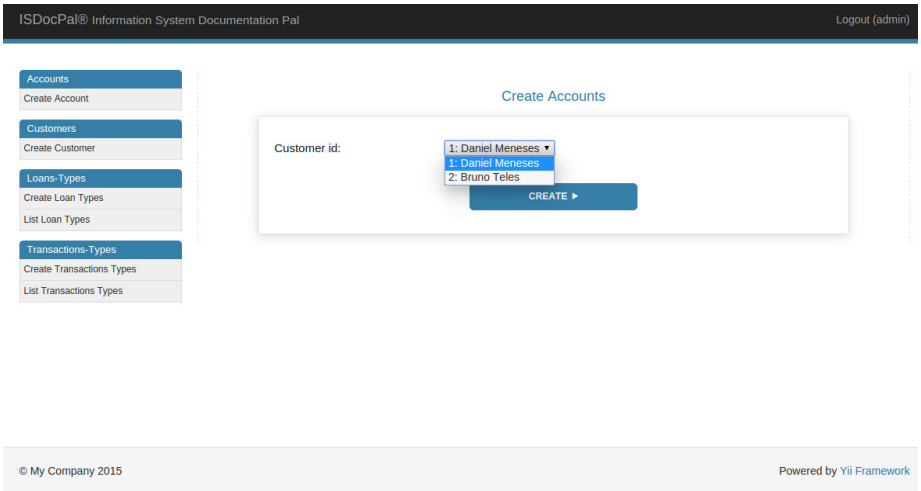


Figura 33: aplicação gerada, criação de conta bancária.

Na Figura 34 mostra-se o login do utilizador *user*.

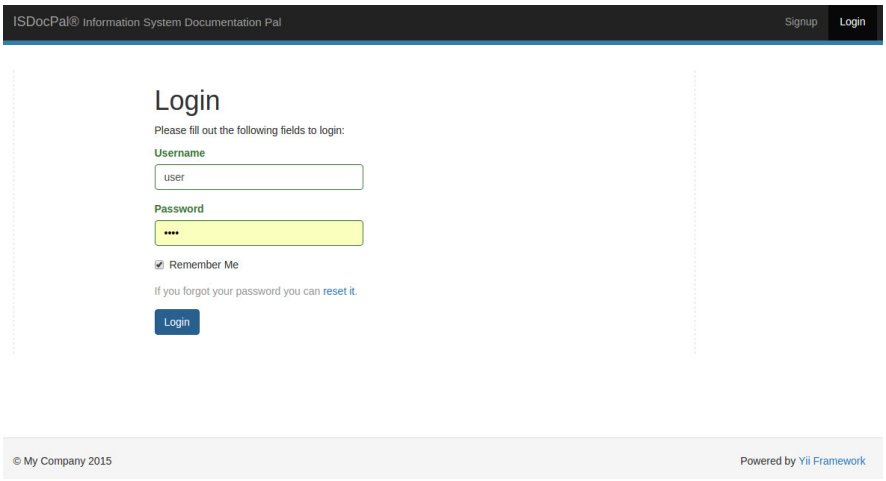


Figura 34: aplicação gerada, login *user*.

Após o login a aplicação apresenta a *home page* e procedeu-se ao pedido de empréstimo (Figura 35).

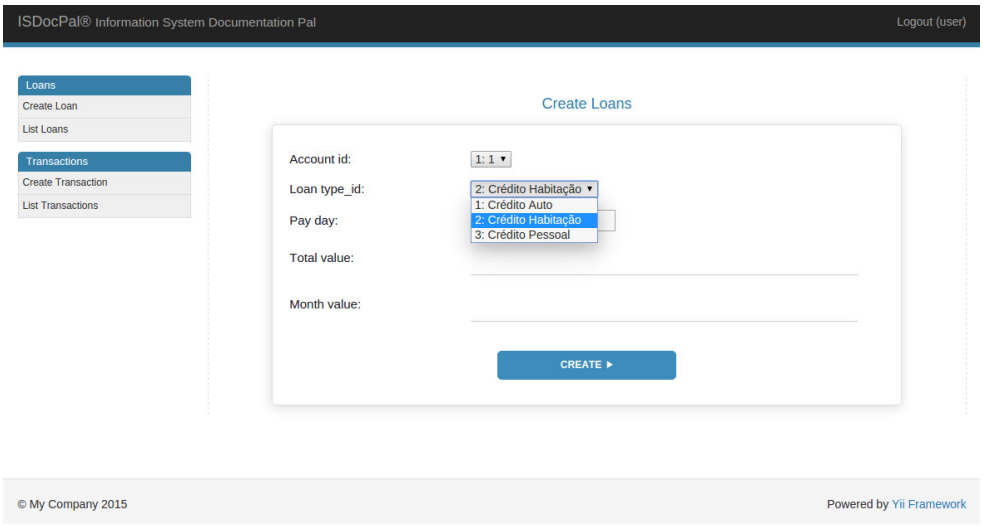


Figura 35: aplicação gerada, pedido de empréstimo.

Conforme se verifica na Figura 35 o utilizador *user* realiza o pedido de empréstimo e pode-se verificar que o campo *loan type_id* possui os valores criados pelo *admin*. Após o preenchimento do formulário e sua submissão o utilizador recebe o feedback presente na Figura 36.

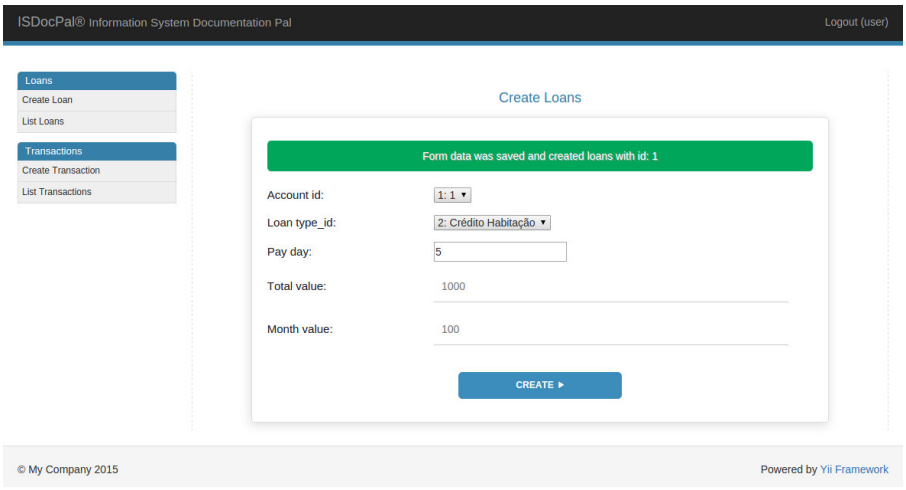


Figura 36: aplicação gerada, pedido de empréstimo, sucesso.

Após a gravação do pedido de empréstimo o utilizador poderá consultar todos os empréstimos clicando em *list loans*. A Figura 37 mostra a listagem de empréstimos requisitados.

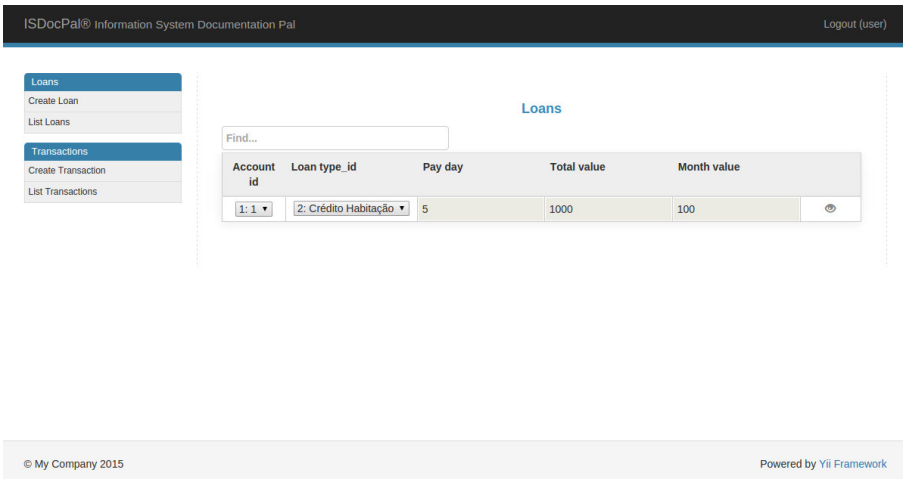


Figura 37: aplicação gerada, listagem de empréstimos.

O exemplo prático da aplicação bancária apresentado permite concluir que o processo de desenvolvimento de aplicações através da ISDocPal é bastante simples. Importa referir que todo o processo de desenvolvimento demorou apenas trinta minutos. É um dado muito importante e bom indicador da eficácia da aplicação ISDocPal.

4.3.2. Discussão dos resultados

O exemplo de implementação prática da aplicação ISDocPal, apresentado neste trabalho para uma aplicação bancária simplificada, demonstra um impacto bastante positivo quanto ao desenvolvimento de SI e quanto ao processo de desenvolvimento de *software*. Relativamente ao primeiro, o impacto positivo deve-se à facilidade e previsibilidade da aplicação, pois com recurso ao diagrama de base de dados foi possível descrever todas as entidades intervenientes no sistema informático reduzindo o receio de futuras alterações, abstraindo o programador da complexidade da sua implementação e mantendo o foco no SI. A descrição dos casos de uso permite estabelecer regras básicas da aplicação como: quais são as atividades disponíveis; como estas devem ser apresentadas ao utilizador; e quem tem permissões para as exercer. Embora seja uma descrição bastante simples do SI, pode-se observar que possui um grande potencial, pois permite a geração de aplicações bastante complexas e facilmente se percebe que se estas tivessem de ser programadas de raiz demoraria bastante mais, a complexidade seria muito mais elevada e alocaria bastantes mais recursos para a sua execução.

No que diz respeito ao processo de desenvolvimento de *software*, pode-se concluir que se observa uma aceleração bastante acentuada no que se refere ao tempo de execução. A aplicação bancária exemplo demorou cerca de trinta minutos a ser realizada, algo impensável se esta mesma fosse totalmente programada. Obviamente que, a aplicação exemplo não se encontra no seu estado final, i.e., seria necessário criar mais código-fonte ou até mesmo realizar modificações ao gerado para que a aplicação pudesse ser enviada para ambiente produtivo com todas as regras de negócio implementadas, no entanto, muito do trabalho foi efetivamente suportado pela ISDocPal. Com base na experiência, estima-se que cerca de 70% do código-fonte possa ser gerado pela ISDocPal e o restante tenha de ser realmente programado, mas no essencial tratar-se-á de código relativo a regras de negócio e navegação da aplicação.

5. Conclusão

No que respeita ao objetivo geral definido, que residiu no desenvolvimento de uma aplicação de *software* que possua a capacidade de gerar aplicações funcionais através de especificações UML, possibilitando posteriores modificações ao seu código fonte, pode-se verificar no capítulo 4 que a aplicação cumpre o proposto. Foi também possível identificar aplicações semelhantes à aplicação proposta e realizou-se a respetiva análise crítica que visou identificar alguns aspetos a melhorar, tendo estes sido ponderados aquando da conceção da aplicação ISDocPal. Foi ainda definida a arquitetura lógica da aplicação ISDocPal que permitiu obter uma visão de alto nível de toda a aplicação e possibilitou organiza-la em subsistemas independentes facilitando a sua compreensão e manutenção. No presente trabalho foi ainda possível demonstrar o funcionamento da aplicação ISDocPal através de um caso exemplificativo (aplicação bancária), o que permitiu demonstrar as mais-valias desta nova ferramenta.

Considera-se que o presente trabalho é uma mais-valia, pelo facto de conseguir resolver, parcialmente, alguns problemas relativos à documentação dos sistemas informáticos, pois a aplicação desenvolvida "obriga" a que a referida seja explicitamente criada no momento da sua utilização. Para o presente trabalho esta documentação assenta essencialmente na especificação do modelo de dados (diagrama de entidades-relacionamentos) e diagrama de casos de uso. No entanto, pode-se verificar que, com apenas estas duas técnicas da UML, as aplicações geradas tornam-se bastante mais previsíveis e fáceis de manter. O impacto no próprio processo de desenvolvimento de *software* é bastante positivo, devido à aceleração verificada no desenvolvimento de aplicações. A aplicação bancária desenvolvida com a ISDocPal demorou apenas trinta minutos, algo assinalável.

Por fim, apresenta-se algumas recomendações e sugestões para trabalho futuro e, neste sentido, considera-se importante que a aplicação seja utilizada em casos reais por forma a reforçar alguns pontos fulcrais para novos desenvolvimentos. Primeiramente, após algumas utilizações em situações reais, devem-se identificar aspetos a melhorar no que diz respeito à experiência de utilização e processos de execução. Por outro lado, é fundamental perceber qual o nível de aceitação por parte dos utilizadores. Para tal, deve ser realizado um estudo que permita recolher depoimentos dos referidos utilizadores em que sejam relatadas a experiência de utilização.

Relativamente a trabalhos futuros é essencial a implementação de uma outra funcionalidade que permita a descrição dos processos inerentes a cada caso de uso descrito, sugerindo-se que seja utilizado o diagrama de atividades da UML para o efeito. Esta funcionalidade permitiria elevar a aplicação no que diz respeito à descrição do sistema informático e resolveria algumas dificuldades sentidas quanto à especificação de processos e regras de negócio a reproduzir nas aplicações geradas, pois na condição que a aplicação se apresenta atualmente, muitos destes processos e regras teriam de ser implementados manualmente. Deve ainda ser avaliada a necessidade, após testes em situações reais, de implementar outro tipo de diagramas da UML como: diagramas de estados e diagramas de sequência. Podem ainda ser realizadas algumas melhorias às aplicações geradas relativamente a questões de apresentação e usabilidade. Sugere-se a implementação de um sistema de *templating* que permita alterar o aspeto das aplicações geradas através de configurações realizadas aquando da geração das mesmas. Estes *templates* poderiam ser simplesmente importados ou desenvolvidos de raiz num formato específico para ser utilizado na ISDocPal.

Com base na experiência profissional do autor, considera-se que a aplicação ISDocPal é uma mais-valia, tanto para o processo de desenvolvimento de *software* como para o próprio desenvolvimento de SI. O processo desenvolvimento de *software* é efetivamente acelerado, o que permite poupança de recursos a todos os níveis e consequentemente redução de custos associados ao desenvolvimento. No que se refere à componente técnica, a aplicação gera código segundo o padrão MVC e para *frameworks* conhecidas e, este é um aspeto bastante importante na perspetiva técnica, i.e., para o programador. No que se refere ao desenvolvimento de SI o impacto é igualmente positivo, pois todo o sistema informático se torna mais previsível e fácil de manter.

6. Referências bibliográficas

@fat, m. Built with Bootstrap. Retrieved 2015-02, 2015, from http://getbootstrap.com/

ALEXOS. (2015). What is PRINCE2? Retrieved 2015-02, 2015, from https://www.axelos.com/best-practice-solutions/prince2/what-is-prince2

Amaral, A., Fernandes, G., & Varajão, J. (2015). Identifying Useful Actions to Improve Team Resilience in Information Systems Projects. Paper presented at the ProjMAN - International Conference on Project MANagement. Procedia Computer Science, 64.

Amaral, L., & Varajão, J. (2000). Planeamento de Sistemas de Informação (2d ed.): FCA - Editora de Informática.

Amaral, L., & Varajão, J. (2007). Planeamento de Sistemas de Informação (4th ed.): FCA - Editora de Informática.

Baptista, J. P., Varajão, J., & Moreira, F. (2013). Função Sistemas de Informação nas organizações: realidade, desafios e oportunidades do uso de arquiteturas empresariais Novas tendências e marketing intelligence (pp. 155-159): Almedina.

Booch, G., Rumbaugh, J., & Jacobson, I. (1999). *The Unified Modeling Language user guide*: Addison Wesley Longman Publishing Co., Inc.

Buckingham, R. A., Hirschheim, R. A., Land, F. F., & Tully, C. J. (1986). *Information systems curriculum: a basis for course design.* Paper presented at the Information systems education: recommendations and implementation.

Carriço, N., Varajão, J., Fernandes, V. B., & Dominguez, C. (2014). Information architecture for is function: A case study. International Journal of Human Capital and Information Technology Professionals, 5(2), 28-37. doi: 10.4018/ijhcitp.2014040103

Catarino, M., Gonçalves, D., Pereira, A., & Varajão, J. (2009). Software projects' most important activities of quality management: A Delphi study. Communications of the IBIMA, 11, 60-66.

Carvalho, J. o. A. I. (1996). Desenvolvimento de Sistemas de Informaçao̧: Da Construçao̧ de Sistemas Informáticos à Reengenharia Organizacional. *Universidade do Minho, Departamento de Sistemas de Informação*.

Collaboration, F. I. (2015). Why React? Retrieved 2015-02, from http://facebook.github.io/react/docs/why-react.html

Colomo-Palacios, R., González-Carrasco, I., López-Cuadrado, J. L., Trigo, A., & Varajao, J. E. (2014). I-Competere: Using applied intelligence in search of competency gaps in software project managers. Information Systems Frontiers, 16(4), 607-625. doi: 10.1007/s10796-012-9369-6

Delamore, S. (2012). MVC Roles and Relationships Diagram. Retrieved 2015-02, from http://www.essenceandartifact.com/2012/12/the-essence-of-mvc.html

Developers, F. (2015). React.js Conf 2015 Keynote - Introducing React Native.

Franco, C. (2015). Git Architecture & 3 types of diff. Retrieved 2015-02, from http://janote.tumblr.com/post/1532197788/git-git-architecture-3-types-of-diff

Gonçalves, D., Cruz, B., & Varajão, J. (2008). Particularidades dos diferentes tipos de projectos de desenvolvimento de software. Paper presented at the 21.° Congresso Internacional de Administração - Gestão estratégica na era do conhecimento (ADM), Brazil.

Gonzálvez-Gallego, N., Molina-Castillo, F. J., Soto-Acosta, P., Varajao, J., & Trigo, A. (2015). Using integrated information systems in supply chain management. Enterprise Information Systems, 9(2), 210-232. doi: 10.1080/17517575.2013.879209

Gonzálvez-Gallego, N., Soto-Acosta, P., Trigo, A., Molina-Castillo, F. J., & Varajão, J. (2010). El papel de las TIC en el rendimiento de las cadenas de suministro: el caso de las grandes empresas de España y Portugal. Universia Business Review, 28, 102-115.

Gouveia, A. J., Oliveira, P. C. R. C., & Varajão, J. (2007). Portais Web: enquadramento conceptual. Paper presented at the Conferência Ibero-Americana WWW/Internet 2007, Vila Real.

Group, T. P. (2014). History of PHP. from http://php.net/manual/en/history.php.php

Hinkle, J. (2013). Retrieved 08/2015, 2015, from https://github.com/jasonhinkle/phreeze

Instagram, F. (2015). React Tutorial Retrieved 2015-02, from http://facebook.github.io/react/docs/tutorial.html

IPMA. (2015). ICB: IPMA Competence Baseline. Retrieved 2015-02, from http://ipma.ch/resources/ipma-publications/ipma-competence-baseline/

Joyent, I. (2015). About Node.js. Retrieved 2015-02, 2015, from http://nodejs.org/about/

Liberato, M., Varajão, J., & Martins, P. (2015). CMMI implementation and results: The case of a software company Modern Techniques for Successful IT Project Management (pp. 48-63): IGI Global.

LLC, Y. S. (2015a). About Yii Framework. 2015-02, from http://www.yiiframework.com/about/

LLC, Y. S. (2015b, 2015). Static structure of a Yii application. Retrieved 2015-02, from http://www.yiiframework.com/doc/guide/1.1/en/basics.mvc

OMG. (2015a). Business Process Model and Notation. Retrieved 2015-02, from http://www.bpmn.org/

OMG. (2015b). Getting Started with UML. Retrieved 2015-02, 2015, from http://www.uml.org/

OutSystems. (2015). Retrieved 04/2015, from http://www.outsystems.com/pricing-and-licensing/

Paiva, A., Varajão, J., Domínguez, C., & Ribeiro, P. (2011). Principais aspectos considerados na avaliação do sucesso de projectos de desenvolvimento de software – há alguma relação com o que é considerado noutras indústrias? Interciencia, 36(3), 200-204.

PMI. (2008). *A guide to the project management body of knowledge (PMBOK guide)* (4th ed.). Newtown Square, Pa.: Project Management Institute, Inc.

PMI, P. M. I., Inc. (2015). PMBOK® Guide and Standards. Retrieved 2015-02, from http://www.pmi.org/PMBOK-Guide-and-Standards.aspx

Q-Success. (2015). Usage of server-side programming languages for websites. Retrieved 2015-02, from http://w3techs.com/technologies/overview/programming_language/all

Ribeiro, P., Paiva, A., Varajão, J., & Dominguez, C. (2013). Success evaluation factors in construction project management - some evidence from medium and large Portuguese companies. KSCE Journal of Civil Engineering, 17(4), 603-609. doi: 10.1007/s12205-013-0019-4

Rijo, R., Varajão, J., & Gonçalves, R. (2012). Contact center: Information systems design. Journal of Intelligent Manufacturing, 23(3), 497-515. doi: 10.1007/s10845-010-0389-0

Santos, C., Santos, V., Tavares, A., & Varajão, J. (2014). Project Management success in health - the need of additional research in public health projects. Paper presented at the ProjMAN - International Conference on Project MANagement. Procedia Technology, 16.

Santos, V., & Varajão, J. (2015). PMO as a Key Ingredient of Public Sector Projects' Success–Position Paper. Paper presented at the ProjMAN - International Conference on Project MANagement, Portugal. Procedia Computer Science, 64.

Scrum.org. (2015). About Scrum.org. Retrieved 2015-02, 2015, from https://www.scrum.org/about

Silva, A., Moreira, F., & Varajão, J. (2010). The Enterprise 2.0 concept: Challenges on data and information security. Paper presented at the 3rd World Summit on the Knowledge Society, WSKS 2010, Corfu. http://link.springer.com/chapter/10.1007%2F978-3-642-16318-0_42

Steve Burbeck, P. D. (1992). Applications Programming in Smalltalk-80™: How to use Model-View-Controller (MVC). *Smalltalk-80 TM*.

Sutherland, K. S. J. (2013). The Definitive Guide to Scrum: The Rules of the Game (pp. 16): Scrum.org.

Team, G. (2015). Git - About. Retrieved 2015-02, 2015, from http://git-scm.com/about

Trigo, A., Varajão, J., Figueiredo, N., & Barroso, J. (2007). Information systems and technology adoption by the Portuguese large companies. Paper presented at the 4th European and Mediterranean Conference on Information Systems, EMCIS 2007, Valencia. http://www.scopus.com/inward/record.url?eid=2-s2.0-84857494861&partnerID=40&md5=85f2b5ed8858720f80b391f3d22afeb3

Trigo, A., Varajão, J., Soto-Acosta, P., González-Callego, N., & Castillo, F. J. M. (2015). Influence of firm size on the adoption of enterprise information systems: insights from Iberian firms International Journal of Information Technology and Management, 14(2), 233-252. doi: 10.1504/IJITM.2015.072046

Varajão, J. (1997). A Arquitectura da Gestão de Sistemas de Informação. (Master's thesis), University of Minho.

Varajão, J. (2001). Outsourcing de Serviços de Sistemas de Informação: FCA - Editora de Informática.

Varajão, J. (2003a). Função de Sistemas de Informação: contributos para a melhoria do sucesso da adopção de tecnologias de informação e desenvolvimento de sistemas de informação nas organizações. (Doctoral thesis), University of Minho.

Varajão, J. (2003b). O comércio electrónico. Revista Informática, Associação Comercial e Industrial dos concelhos de Monção e Melgaço.

Varajão, J. (2005). A Arquitectura da Gestão de Serviços de Sistemas de Informação (3rd ed.): FCA - Editora de Informática.

Varajao, J. , Ribeiro, A. T. , Figueiredo, N., & Barroso, J. (2007). Motivações inerentes à adopção de Tecnologias e Sistemas de Informação nas grandes empresas portuguesas. Paper presented at the CISCI - 6ta Conferencia Iberoamericana En Sistemas, Cibernetica E Informatica, Memorias, Vol I, US. Proceedings Paper retrieved from

Varajão, J., Cardoso, J., Gonçalves, D., & Cruz, J. B. (2008). Análise à gestão de projectos de desenvolvimento de software em grandes empresas portuguesas. Semana Informática, 10-12.

Varajão, J., & Cruz-Cunha, M. M. (2013). Using AHP and the IPMA Competence Baseline in the project managers selection process. International Journal of Production Research, 51(11), 3342-3354. doi: 10.1080/00207543.2013.774473

Varajão, J., Domingues, C., Ribeiro, P., & de Paiva, A. (2014). Failures in software project management–are we alone? A comparison with construction industry. The Journal of Modern Project Management, 2(1).

Varajão, J., Dominguez, C., Ribeiro, P., & Paiva, A. (2014). Critical success aspects in project management: Similarities and differences between the construction and the software industry. Tehnicki Vjesnik, 21(3), 583-589.

Varajão, J., Trigo, A., & Barroso, J. (2009). Motivations and Trends for IT/IS Adoption: Insights from Portuguese Companies. International Journal of Enterprise Information Systems, 5(4), 34-52. doi: 10.4018/jeis.2009090203

Varajão, J., Trigo, A., Figueiredo, N., & Barroso, J. (2007a). TI nas empresas nacionais. Revista CXO, 19-23.

Varajão, J., Trigo, A., Figueiredo, N., & Barroso, J. (2007b). TI nas empresas nacionais: Importância das TI na produtividade das empresas. Computerworld, 9.

Varajão, J., Trigo, A., Figueiredo, N., Barroso, J., & Bulas-Cruz, J. (2009). Information systems services outsourcing reality in large Portuguese organisations. International Journal of Business Information Systems, 4(1), 125-142. doi: 10.1504/IJBIS.2009.021606