

[Título: subtítulo]

Nuno Alexandre Gonçalves de Sousa Maia

Nº 21200 – Regime Diurno

Orientação

Patrícia Leite

Ano letivo 2022/2023

Licenciatura em Engenharia Informática Médica

Escola Superior de Tecnologia

Instituto Politécnico do Cávado e do Ave

**Identificação do Aluno**

Nuno Alexandre Gonçalves de Sousa Maia

Aluno número 21200, regime diurno

Licenciatura em Engenharia Informática Médica

**Orientação**

Patrícia Leite

Professora Adjunta

**Informação sobre o Estágio**

Nome da Empresa: Buzina

Morada Completa: Largo 12 de Dezembro nº25, Pisos 0 e 1, 4705-259. Braga

Orientação de Carlos Manuel Ribeiro Ferreira

RESUMO

Este trabalho foca-se no desenvolvimento duma aplicação web. Foi optado o uso da framework de PHP Laravel. O objetivo é a gestão interna da empresa no que toca à organização de dados sobre os seus clientes e projetos.

A funcionalidade principal da aplicação é a gestão das prioridades dos empregados, que permite um controlo eficiente das tarefas e responsabilidades nos vários projetos.

O objetivo é melhorar o fluxo de trabalho, reduzir erros, e providenciar uma clara visão sobre o trabalho a fazer.

ABSTRACT

This project focuses on the development of a web application. The use of the PHP framework Laravel was chosen. The goal is the internal management of a company regarding managing data related to their clients and projects.

The main functionality is the management of employee priorities, to achieve an efficient control over the tasks and responsibilities of various projects.

The goal was to improve the workflow, reduce errors, and provide a clear vision over the remaining work.

ÍNDICE

[1. Introdução 1](#_Toc177573230)

[1.1. Objetivos 1](#_Toc177573231)

[1.2. Contexto 1](#_Toc177573232)

[1.3. Estrutura do documento 1](#_Toc177573233)

[2. Estado de arte 3](#_Toc177573234)

[2.1. Método Kanban 3](#_Toc177573235)

[2.2. Outras soluções 4](#_Toc177573236)

[3. Trabalho desenvolvido 5](#_Toc177573237)

[3.1. Análise e especificações 5](#_Toc177573238)

[3.1.1. Requisitos funcionais 5](#_Toc177573239)

[3.1.2. Requisitos não funcionais 6](#_Toc177573240)

[3.2. Arquitetura da solução 7](#_Toc177573241)

[3.2.1. MVC 7](#_Toc177573242)

[3.2.2. Arquitetura geral 7](#_Toc177573243)

[3.3. Tecnologias utilizadas 9](#_Toc177573244)

[4. Conclusão 10](#_Toc177573245)

[4.1. Conclusão geral 10](#_Toc177573246)

[4.2. Trabalho futuro 10](#_Toc177573247)

ÍNDICE DE FIGURAS

[Figura 1 – Diagrama ER 8](#_Toc177573248)

Siglas e Acrónimos

**PHP** – PHP: Hypertext Preprocessor (PHP: Pré-processador de Hipertexto)

**CSS** – Cascading Style Sheets (Folha de Estilo em Cascatas)

**SEO** – Search Engine Optimization (Otimização de Motores de Pesquisa)

**ORM** – Object-Relational Mapper (mapeador de objetos relacional)

**CRUD** – Create, Read, Update, Delete (Ler, Escrever, Atualizar, Apagar)

# Introdução

## Objetivos

Pretende-se o desenvolvimento duma ferramenta capaz de gerir as prioridades dos empregados duma empresa. Esta deve também guardar registos quanto aos vários clientes para os quais se trabalha.

## Contexto

Este projeto foi realizado no âmbito de estágio curricular no curso de Engenharia Informática Médica, na empresa Buzina em Braga. O estágio começou no dia 28 de fevereiro e acabou no dia 21 de junho; inicialmente no regime de quatro dias por semana, passando mais tarde a cinco.

A Buzina é uma empresa fundada em 2008 com escritórios em Lisboa, Porto e Braga. Esta desenvolve soluções à medida para empresas, com o objetivo de fazer o que for necessário.

Atualmente a Buzina já desenvolveu mais de 500 projetos com diferentes graus de exigência e de dimensão. Todos os projetos tiveram como resultado a satisfação e consequente sucesso dos seus clientes.

## Estrutura do documento

Este documento está dividido em 4 grandes partes, cada uma destas com os respetivos subcapítulos.

Neste primeiro capítulo fala-se um bocado do projeto em geral, bem como duma contextualização do projeto desenvolvido.

No “*2. Estado de arte”* são exploradas outras opções que poderiam ter sido utilizadas, evitando o desenvolvimento deste projeto, e razões para ter sido optada uma solução de raiz.

No capítulo seguinte, “3. *Trabalho desenvolvido*”é documentado todo o processo de desenvolvimento e a maneira como as tecnologias apresentadas se relacionam com o projeto desenvolvido.

No final, encontra-se a conclusão, onde se fala acerca do trabalho desenvolvido e de trabalho futuro.

# Estado de arte

O conceito da aplicação é bastante simples. Pode-se resumir a “manter registo das prioridades dos empregados”. É então óbvio que há já bastantes opções que permitem fazer isso. Não há como falar dessas soluções sem começar pela base da maioria delas, o método *Kanban*.

## Método Kanban

O método Kanban é uma abordagem visual para a gestão de trabalho, criada pela Toyota. Foi criado para otimizar o processo de produção industrial, mas acabou por ser adaptado para a área de desenvolvimento de software.

O Kanban tem como base quatro princípios fundamentais:

* Visualização do trabalho

Todo o processo do trabalho deve estar visível, para facilitar a distinção entre as tarefas pendentes / em progresso ou concluídas.

* Limitar o trabalho em progresso (*WIP*)

Há um limite de *WIP*. Isto é essencial para garantir que nunca há demasiadas tarefas por fazer, permitindo que todo o foco esteja numa quantidade limitada de tarefas ao longo do projeto.

* Gestão de fluxo

Todas as tarefas são, individualmente, pequenas. Isto visa que nunca se passe muito tempo numa só tarefa, melhorando o fluxo de trabalho.

* Melhoria contínua

Podem sempre ser adicionadas novas tarefas. Isto permite que um trabalho só seja considerado terminado assim que a empresa o deseje. Pode-se sempre adicionar funcionalidades.

**Vantagens**

O método *Kanban* é simples de compreender e de utilizar. Todo o trabalho, nos seus vários estados (feito / a fazer / por fazer) está visível para todos os membros da equipa. Permite também ver aquelas tarefas que estão a demorar demasiado tempo, para que se evitem bloqueios.

## Outras soluções

Há algumas soluções que tratam do nosso problema. Por exemplo: *Trello*, *Asana* e *Microsoft Planner*. A nossa solução foi feita devido à necessidade duma aplicação privada, sobre a qual teríamos completo controlo, e a liberdade de adicionar as funcionalidades desejadas.

Obviamente que alguma inspiração para o projeto foi retirada destas soluções. Foi utilizado o método *Kanban*, tal como no *Trello* e no *Microsoft Planner*.

Tal como no *Trello*, os projetos são divididos em tarefas, que estão associadas a empregados.

Uma das caraterísticas do *Asana* é a possibilidade de “pedir ajuda” a outras equipas nos projetos; isto também está presente na nossa solução sobre a forma de adicionar colaboradores aos projetos / tarefas.

Uma caraterística presente em todas essas soluções, e na nossa, é a simples *interface*, que permite a interação sem interrupções pelos projetos da equipa.

A nossa solução permite também a fácil navegação entre os vários clientes e projetos da empresa. Devido a isto, o projeto tem uma estrutura bastante modular, o que ajuda na sua utilização visto que as várias partes são semelhantes entre si.

# Trabalho desenvolvido

Neste capítulo será mencionado o trabalho efetuado antes da implementação. Este ponto é essencial para garantir que temos sempre em mente um objetivo claro. Assim podemos também evitar redundâncias, e a implementação de funcionalidades desnecessárias. É aqui feita uma análise dos requisitos funcionais e não funcionais, bem como uma simples demonstração da arquitetura do sistema e das tecnologias utilizadas.

## Análise e especificações

Nas especificações são mencionados e identificados todos os requisitos que se pretende ter na aplicação. Estes estão divididos entre requisitos funcionais (RF) e requisitos não funcionais (RNF).

### Requisitos funcionais

Os requisitos funcionais definem aquilo que o sistema pode fazer; as suas funcionalidades e interações.

* RF01 – Tipos de cliente
  + RF01.1 – O administrador e o colaborador podem adicionar e editar tipos de cliente.
  + RF01.2 – Apenas o administrador pode eliminar tipos de cliente
* RF02 – Níveis de acesso
  + RF02.1 – Apenas o administrador pode adicionar, editar ou eliminar níveis de acesso
* RF03 – Estados
  + RF03.1 – O administrador e o colaborador podem adicionar e editar estados de projeto
  + RF03.2 – Apenas o administrador pode eliminar estados de projeto.
* RF04 – Clientes
  + RF04.1 – O administrador e o colaborador podem adicionar e editar os clientes.
  + RF04.2 - Apenas o administrador pode eliminar clientes.
* RF05 – Projetos
  + RF05.1 – Apenas o administrador pode adicionar ou eliminar projetos.
  + RF05.2 – O administrador e o colaborador podem editar projetos.
* RF06 – Prioridades
  + RF06.1 – O colaborador só pode alterar as suas próprias prioridades.
  + RF06.2 – O administrador pode criar, editar e eliminar as prioridades.
* RF07 – Férias
  + RF07.1 – Se um colaborador estiver de férias, deverá ser impossível editar as suas prioridades.
* RF08 – Rotina
  + RF08.1 – O sistema deve automaticamente bloquear a edição de prioridades após as 20:00 horas de sexta-feira.

### Requisitos não funcionais

Os requisitos não funcionais referem-se a como o sistema deve operar. Isto inclui a usabilidade esperada, bem como o seu desempenho.

* RNF01 – Tempo de resposta
  + RNF01.1 – Nenhuma ação deve demorar mais de 2 segundos a ser processada.
* RNF02 – Sinalização de tempo nos projetos
  + RNF02.1 – Os projetos devem ter uma sinalização de tempo identificada por uma cor. Verde deve sinalizar que foi terminado a tempo, azul que demorou tanto quanto esperado, e vermelho que excedeu o tempo esperado.
* RNF03 – Drag & Drop
  + RNF03.1 – Deve ser possível editar a ordem das prioridades apenas ao arrastar estas entre si.
  + RNF03.2 – Deve ser possível editar o estado das prioridades apenas ao arrastar estas entre listas.

## Arquitetura da solução

### MVC

*Model-View-Controller* é um padrão de design utilizado para estruturar aplicações. Este divide a aplicação em três componentes ligados.

* *Model*
  + Representa os dados ou lógica da aplicação.
  + Trata diretamente das regras da aplicação.
  + É responsável por interagir com a base de dados, receber dados e mandar estes para o *Controller*.
* *View*
  + Responsável por dar display aos dados providenciados pelo Modelo num outro formato (normalmente HTML).
  + Representa a interface da aplicação.
* *Controller*
  + Age como intermédio entre a *View* e o *Model*.
  + Processa requests, manipula modelos, e escolhe a *view* certa para dar display aos dados.

Como funciona o MVC:

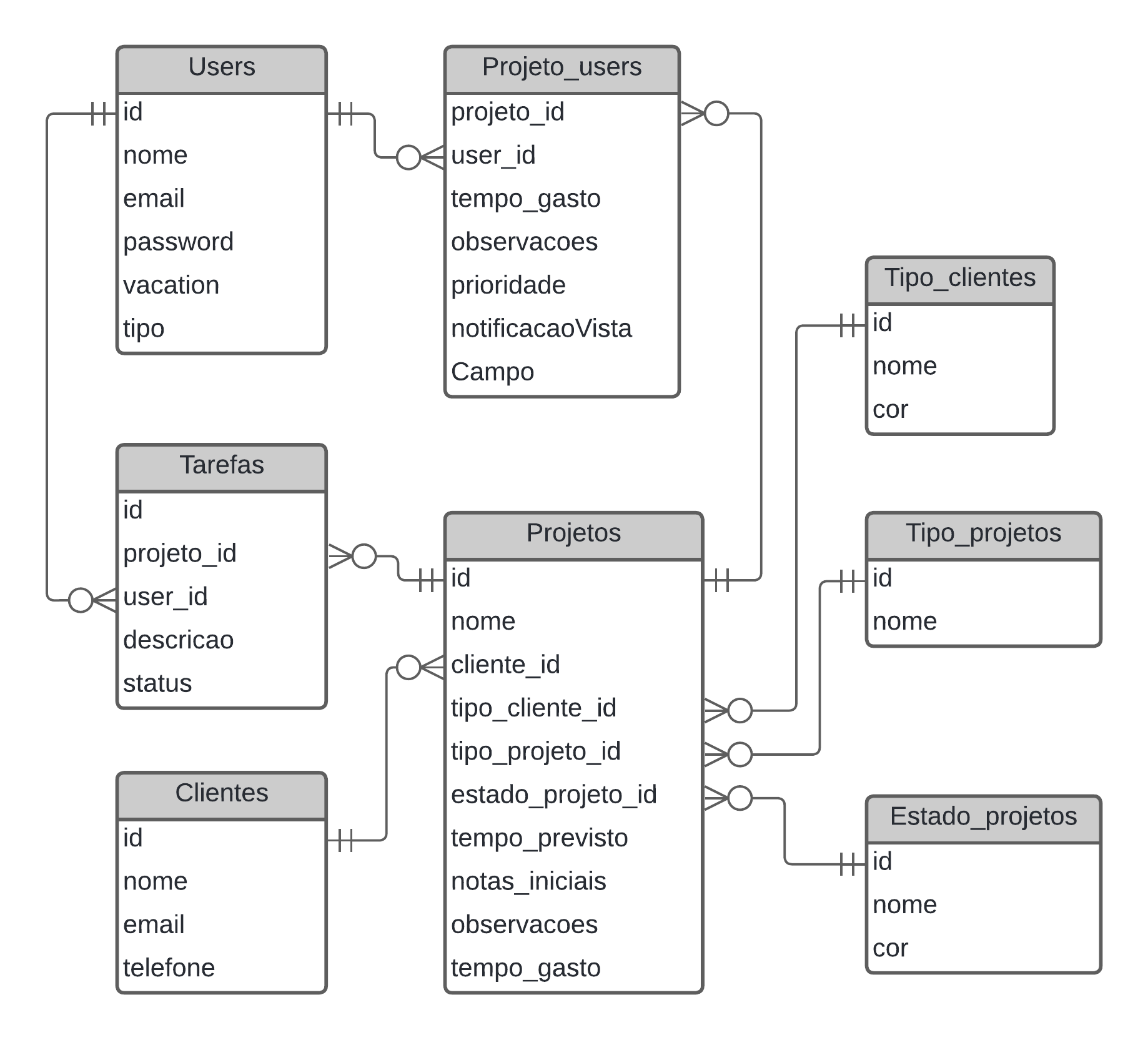
1. O *Controller* recebe um *request*.
2. O *Controller* interage com o *Model* para receber ou atualizar os dados necessários.
3. Quando o *Controller* recebe os dados do *Model*, manda estes para a *View*.
4. A *View* dá display aos dados ao utilizador (browser).

### Arquitetura geral

Esta é uma solução orientada a negócios. Isto significa que todo o trabalho foi efetuado para responder às necessidades duma empresa; neste caso, a gestão das prioridades dos seus empregados.

Abaixo encontra-se um diagrama de entidade-relação a simplificar a estrutura da aplicação.

Figura 1 – Diagrama ER



De maneira simplificada:

A cada utilizador é atribuído um ou mais projetos, que por sua vez podem ter vários utilizadores com tarefas diferentes atribuídas a si. Cada cliente (em caso geral, as empresas) tem um ou mais projetos, cada um com um tipo (ex: fazer website, ou SEO), e um estado (pendente / em progresso / completo).

## Tecnologias utilizadas

Para fazer todo este projeto, as seguintes tecnologias foram utilizadas:

* Laravel

Laravel é uma *framework* de PHP open-source. Esta basicamente reviveu o PHP, que até 2011 tinha estado em declínio.

* Composer

Ferramenta utilizada para controlar as dependências dum projeto PHP.

* *Eloquent*

Incluído em todos os projetos de Laravel, o Eloquent é um ORM que facilita a implementação MVC que é a base do Laravel. Este permite a fácil interação com a base de dados.

* *TailwindCSS*

TailwindCSS é, como o nome indica, uma framework de CSS, como o bootstrap. Este agrupa e parametriza classes comuns de CSS, para reduzir a quantidade de código e simplificar os estilos de cada elemento.

* *Git*

Em conjunto com o Github, o Git é a mais utilizada ferramenta para controlo de versões. O seu uso é esperado em todas as aplicações feitas no dia a dia.

* *XAMPP*

XAMPP foi a solução escolhida para lidar com a criação de servidores locais. Isto é essencial para facilitar a réplica de como o nosso trabalho se comporta quando é testado “a sério”.

# Conclusão

## Conclusão geral

Com um bocado de esforço é possível fazer o que for necessário.

Neste estágio foram desenvolvidas várias competências relacionadas com programação web no geral, que serão úteis no futuro. Foi efetuado trabalho relacionado com o design de páginas, manuseamento de requests e operações CRUD.

Todo o projeto está disponível em <https://github.com/Deonap/taskApp>.

## Trabalho futuro

O trabalho pode ser considerado não terminado. Há algumas maneiras para o complementar, como por exemplo:

* Melhorar o tempo de resposta.
* Implementar um melhor sistema de notificações.
* Algumas mudanças na UI para melhorar a experiência do utilizador.