

Fiducial: Redmine - Project Monitor

Francisco Rafael Dias Pereira

Nº 21156 – Regime Pós-laboral

Orientação

Eva Oliveira

Ano letivo 2022/2023

Licenciatura em Engenharia de Sistemas Informáticos

Escola Superior de Tecnologia

Instituto Politécnico do Cávado e do Ave

**Identificação do Aluno**

Francisco Rafael Dias Pereira

Aluno número 21156 regime pós-laboral

Licenciatura em Engenharia de Sistemas Informáticos

**Orientação**

Eva Ferreira de Oliveira

Professora adjunta

**Informação sobre o Estágio**

Fiducial

Rua dos Padres Carmelitas, Centro de Negócios Ideia Atlântico CX 192,

União Freguesias Nogueiró, Tenões, 4719-005 Braga

Orientação de Juan Sarmiento

RESUMO

Este relatório retrata todo o trabalho desenvolvido no Estágio de licenciatura nas instalações da Fiducial. O trabalho consistiu em entender o negócio da empresa através de formação providenciada pela Fiducial, definição de requisitos para o projeto “Redmine Project Monitor”, desenvolvimento e integração.

A Fiducial é uma empresa que desenvolve Software para a área financeira, originária de León, França. Dispõe de perto de 20 mil colaboradores, dispersos por mais de 1000 localizações, e faturou 1.83 Biliões de acordo com os dados providenciados em 2021 [1].

O projeto “Redmine Project Monitor” consiste no desenvolvimento de uma API REST, que permite interligar os diferentes projetos em desenvolvimento da Fiducial hospedados no Redmine. Desta forma será possível à Fiducial, recolher informação de todos os projetos, condensar e tratar a informação, e por último, fazer uma análise geral em uma *Dashboard*.

O “Redmine Project Monitor” foi desenvolvido em PHP com a *Framework* *Symfony* [2] seguindo o paradigma de desenvolvimento MVC.

ABSTRACT

This document reports all work of the Bachelour internship done at Fiducial. This work focuses in understanding what Fiducial business is, requirements definition for the project “Redmine Project Monitor” and thus development and integration.

Fiducial is a company that develops Software for the financial sector, from León, France. It has near to 20 thousand collaborators, dispersed from more than 1000 locations, and makes near to 1.83 Billion in revenue according to the provided finantial data from 2021 [1].

The “Redmine Project Monitor” focuses on the development of a REST API that enables to Exchange information between diferente projects hosted in Redmine from Fiducial. This way, it’s possible for Fiducial to retrieve information about all projects, process this information to show in a Dashboard for overall analysis.

The “Redmine Project Monitor" is developed in PHP using the Symfony Framework [2] and it follows the MVC Software Development concept.

AGRADECIMENTOS

Gostaria de aproveitar a oportunidade para agradecer à minha família e ao IPCA por me ter providenciado esta oportunidade. Este estágio foi muito importante para o meu início da carreira profissional que permitiu desenvolver algum entendimento acerca de como as carreiras de Desenvolvimento de Software são.

Gostaria de agradecer à Fiducial, em particular ao Juan Sarmiento e ao Huender por todo o tempo despendido durante o meu tempo na empresa e pela forma exemplar como fui tratado.

Para terminar, gostaria de agradecer à Professora Eva Oliveira, por me ter acompanhado em todo o Estágio e me ter apoiado na elaboração deste relatório.

ÍNDICE

[1. Introdução 1](#_Toc141864901)

[1.1. Objetivos 1](#_Toc141864902)

[1.2. Estrutura do documento 2](#_Toc141864903)

[2. Tecnologias Utilizadas 3](#_Toc141864904)

[2.1. PHP 3](#_Toc141864905)

[2.2. Docker 3](#_Toc141864906)

[2.3. Github 4](#_Toc141864907)

[2.4. Jetbrains PHPStorm 4](#_Toc141864908)

[2.5. Framework Symfony 5](#_Toc141864909)

[2.6. PostMan 5](#_Toc141864910)

[2.7. XAMPP 5](#_Toc141864911)

[3. Análise e Modelação do Sistema 7](#_Toc141864912)

[3.1. Levantamento de Requisitos 7](#_Toc141864913)

[3.1.1. Requisitos Funcionais 7](#_Toc141864914)

[3.1.2. Requisitos Não Funcionais 8](#_Toc141864915)

[3.2. Metodologias Aplicadas 9](#_Toc141864916)

[3.3. Diagrama de casos de uso 10](#_Toc141864917)

[3.4. Diagrama de classes 10](#_Toc141864918)

[3.5. Diagramas de sequência 11](#_Toc141864919)

[4. Implementação 13](#_Toc141864920)

[4.1. Arquitetura 13](#_Toc141864921)

[4.2. Apresentação da solução desenvolvida 15](#_Toc141864922)

[4.2.1. Extração dos projetos do Project Monitor 15](#_Toc141864923)

[4.2.2. Criação de Projetos no Redmine 17](#_Toc141864924)

[4.3. Análise de resultados 18](#_Toc141864925)

[5. Conclusão 21](#_Toc141864926)

[5.1. Apreciação Final 21](#_Toc141864927)

[5.2. Trabalho Futuro 21](#_Toc141864928)

ÍNDICE DE FIGURAS

[Figura 1 - Logo PHP [4] 3](#_Toc141864929)

[Figura 2 - Logo Docker [5] 4](#_Toc141864930)

[Figura 3 - Github logo [8] 4](#_Toc141864931)

[Figura 4 - Jetbrains PHP Storm IDE [10] 5](#_Toc141864932)

[Figura 5 - Framework Symfony [11] 5](#_Toc141864933)

[Figura 6 - Postman logo [6] 5](#_Toc141864934)

[Figura 7 - XAMP logo [13] 6](#_Toc141864935)

[Figura 8 - Modelo de desenvolvimento de Software em Cascata [14] 7](#_Toc141864936)

[Figura 9 - Diagrama de caso de uso da solução desenvolvida 10](#_Toc141864937)

[Figura 10 - Diagrama de referentes ao Redmine e Project Monitor 11](#_Toc141864938)

[Figura 11 - Diagrama de sequência da interação entre os diferentes atores 11](#_Toc141864939)

[Figura 12 - Diagrama de componentes do sistema e interfaces 14](#_Toc141864940)

[Figura 13 - Estrutura do projeto (Symfony) 14](#_Toc141864941)

[Figura 14 - HTTP POST ao Redmine 17](#_Toc141864942)

[Figura 16 - Configuração da API key e resultado 18](#_Toc141864943)

[Figura 17 - Execução do processo de sincronização com sucesso 19](#_Toc141864944)

[Figura 18 - Verificação da sincronização dos projetos no Redmine 19](#_Toc141864945)

[Figura 19 - Gestão de sincronismo no caso de os projetos já existirem 20](#_Toc141864946)

[Figura 20 - Anexo, Tarefa PMIR-1 XXIII](#_Toc141864947)

[Figura 21 - Anexo, Tarefa PMIR-2 XXIII](#_Toc141864948)

[Figura 22 - Anexo, Tarefa PMIR-3 XXIV](#_Toc141864949)

[Figura 23 - Anexo, Tarefa PMIR-4 XXIV](#_Toc141864950)

[Figura 24- Anexo, Tarefa PMIR-5 XXV](#_Toc141864951)

[Figura 25- Anexo, Tarefa PMIR-6 XXV](#_Toc141864952)

[Figura 26- Anexo, Tarefa PMIR-7 XXVI](#_Toc141864953)

[Figura 27- Anexo, Tarefa PMIR-8 XXVII](#_Toc141864954)

[Figura 28- Anexo, Tarefa PMIR-9 XXVIII](#_Toc141864955)

[Figura 29- Anexo, Tarefa PMIR-10 XXIX](#_Toc141864956)

ÍNDICE DE TABELAS

[Tabela 1 - Requisitos Funcionais 8](#_Toc141864957)

[Tabela 2 - Requisitos Não funcionais 9](#_Toc141864958)

[Tabela 3 - Listagem e descrição dos módulos do sistema 15](#_Toc141864959)

[Tabela 4 - Descrição dos campos enviados pelo Project Monitor 16](#_Toc141864960)

[Tabela 5 - Mapeamento entre as chaves do Redmine e Project Monitor 18](#_Toc141864961)

Glossário

**Framework** – É uma estrutura composta por vários módulos de Software diferente (programas, bibliotecas, ...) e que permite construir outros Softwares complexos.

**Redmine** – É um Software livre de gestão de projetos que providencia uma interface web.

Siglas e Acrónimos

**API** – “Application Programming Interface”, isto é, uma interface de programação aplicacional. Normalmente apresentada por assinaturas de funções/métodos localizados nas fronteiras do componente/módulo.

**IDE –** “Integrated Development Environment”, isto é, um ambiente de desenvolvimento integrado que contém um editor, ferramentas de compilação e depuração.

**MVC** – “Model View Controller”, isto é, um paradigma de desenvolvimento que permite estruturar componentes de um Software e localizá-los em modelos, ou vistas ou controladores. Um modelo é responsável por fazer interface com uma base de dados, as vistas são responsáveis por atender a pedidos e os controladores são responsáveis pela lógica e interface entre vistas e modelos.

# Introdução

Este relatório documenta todo o trabalho desenvolvido na Fiducial durante o estágio curricular de licenciatura. Nos momentos iniciais do estágio foi definido um projeto enquadrado com o negócio da empresa de nome “Redmine: Project Monitor”. O objetivo do projeto é de sincronizar os projetos hospedados no sistema “Project Monitor”, da Fiducial, em um sistema de gestão de projetos Redmine. Apenas informação não-sensível será migrada.

De forma a contextualizar melhor o projeto, é necessário entender o que é o sistema “Projeto Monitor” e o sistema “Redmine”, ambos existentes na Fiducial.

O “Project Monitor” é um portal web da intranet da Fiducial, com o objetivo de armazenar informação relativa a todos os projetos (como financiamento, tarefas, responsáveis, entregas). Este é de acesso restrito apenas a funcionários da empresa, sendo que, como estagiário apenas tinha informação acerca do formato das mensagens e não do conteúdo.

O “Redmine”, é um sistema web de gestão de projetos aberto [3], hospedado na intranet da Fiducial. A este sistema é de acesso livre, dentro da intranet da Fiducial.

Através da conclusão deste projeto, a Fiducial terá forma de consultar informação acerca de todos os projetos e apresentar dentro da intranet Fiducial a todos os funcionários, de forma a aumentar a transparência e nunca partilhando informação sensível. Sem esta solução, os funcionários não conseguem ter acesso ao “Project Monitor”, e necessitam de perguntar à gestão informação acerca dos processos, traduzindo-se assim em ineficiência dentro da organização.

## Objetivos

O objetivo deste projeto consiste em desenvolver uma API que permite obter todos os projetos do “Project Monitor” da Fiducial, e criar no sistema de gestão de projetos “Redmine”. Esta solução terá de ser desenvolvida sobre *containers* Docker de forma a garantir a modularidade e portabilidade da solução

A empresa definiu as tecnologias de desenvolvimento, PHP [4] e Symfony [2] respetivamente. Sendo assim, os objetivos específicos são:

* Estudo do problema de sincronização de dados entre o “Project Monitor” e “Redmine”
* Criação do projeto que permite implementar a solução
* Estudo da linguagem de programação PHP.
* Estudo da *framework* Symfony [2]
* Estudo do sistema de conteinerização Docker [5]
* Criação de um sistema de testes utilizando Postman[6]

## Estrutura do documento

Este documento será apresentado de acordo com a seguinte estrutura:

2. Tecnologias Utilizadas : Apresentação do conjunto de tecnologias que permitiu desenvolver este projeto.

3. Análise e Modelação do Sistema : Levantamento e análise dos requisitos funcionais e não funcionais. Descrição dos casos de uso, atividades e classes.

4. Implementação: Descrição detalhada da implementação, apresentando a estrutura do projeto e aspetos relevantes da implementação.

5. Conclusão: Análise geral dos resultados obtidos com o desenvolvimento do projeto de estágio e retrospetiva acerca da aquisição de conhecimentos.

# Tecnologias Utilizadas

Neste subcapítulo são apresentadas as tecnologias utilizadas na implementação da solução. As tecnologias apresentadas, como referido anteriormente, foram maioritariamente definidas pela empresa.

## PHP

O PHP [4] é uma linguagem de programação interpretada de código livre cujo objetivo inicial é de desenvolvimento de aplicações web do lado do servidor. Isto significa que o PHP permite gerar respostas de pedidos HTTP [7] de forma customizada. Foi uma linguagem importante na criação do *web* 2.0, isto é, na criação da segunda geração de internet que permitia maioritariamente a interação e colaboração do cliente na partilha de conteúdo.



Figura - Logo PHP [4]

## Docker

O Docker [5] é um conjunto de serviços que permite criar módulos (*Containers*) de *Software*, isolados e independentes, virtualizados ao nível do sistema operativo. Isto significa que podemos criar um container Docker em uma máquina *Linux* e executar esse mesmo container em uma máquina *Windows*, desde que, a aplicação Docker esteja instalada.

Desta forma é possível desenvolver serviços em pequenas unidades, garantindo a independência dos blocos e fazer com que estes blocos comuniquem entre si. Assim, perde-se a dependência da máquina e permite de forma customizável migrar funcionalidades entre diferentes sistemas de forma simples, rápida e segura. Acerca da performance, o Docker criar *containers* de tamanho reduzido, o que permite um *deployment* mais rápido.

Uma imagem com Gráficos, logótipo, clipart, captura de ecrã

Descrição gerada automaticamente

Figura - Logo Docker [5]

## Github

O Github [8] é um serviço de hospedagem e colaboração de desenvolvimento de *Software*. O Github integra com a ferramenta Git [9] e permite ter gestão de versões, *bugs*, *features* e mais funções que aumentam a produtividade no desenvolvimento de *Software*.

Uma imagem com preto, escuridão

Descrição gerada automaticamente

Figura - Github logo [8]

## Jetbrains PHPStorm

O Jetbrains PHP Storm [10] é um *Integrated Development Environment* (IDE) desenvolvido em Java. Ele possui um conjunto de ferramentas que permitem aumentar a produtividade no desenvolvimento de PHP, como *auto-complete*, validação de código, gestão de ficheiros, *deployment*, ... A grande vantagem de utilizar este IDE deve-se ao facto de automatizar muitas tarefas.



Figura - Jetbrains PHP Storm IDE [10]

## Framework Symfony

A Framework Symfony [2] é um conjunto de bibliotecas grátis e open-source, escritas em PHP, com o objetivo de providenciar blocos reutilizáveis de Software que permitem acelerar o desenvolvimento de aplicações Web escritas em PHP.



Figura - Framework Symfony [11]

## PostMan

O Postman [6] é uma ferramenta que permite testar e gerir serviços web (HTTP/HTTPS), providenciando uma interface simples e intuitiva. A missão do Postman é de simplificar os testes de APIs.



Figura - Postman logo [6]

## XAMPP

O XAMPP [12] é um pacote que contém os principais servidores de código aberto. O nome XAMP, é um acrónimo com o seguinte significado: X – Qualquer sistema Operativo, A – Apache, M – MariaDb, P – PHP e por último P – Perl.

O XAMPP, além das funções mencionadas anteriormente, dispõe ainda de um servidor FTP e inclui outros módulos como o OpenSSL e o PHPMyAdmin.



Figura - XAMP logo [13]

# Análise e Modelação do Sistema

Neste capítulo é descrito todo o trabalho de análise, modelação e levantamento de requisitos. O objetivo é facilitar a compreensão do sistema permitindo estruturar a informação de forma simples e metódica. A estrutura utilizada será muito aproximada ao descrito no modelo de desenvolvimento de Software em cascata [14].

Uma imagem com Gráficos, círculo, captura de ecrã, design gráfico

Descrição gerada automaticamente

Figura - Modelo de desenvolvimento de Software em Cascata [14]

## Levantamento de Requisitos

O levantamento de requisitos é o primeiro passo no processo de planeamento e desenvolvimento de uma aplicação de Software. Este passo é importante pois permite detalhar e esquematizar detalhadamente as funcionalidades do sistema.

Este processo foi desenvolvido de forma iterativa com os *stakeholders* do projeto (Fiducial), e o resultado desta análise foi a definição das tarefas em anexo (Figura 19 - Figura 28).

### Requisitos Funcionais

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | TAG | Requisito | Descrição | Prio. | Tipo R |
| 1.1 | Ligação ao Sistema “Project monitor” | Recolha de Projetos | O sistema deve permitir conectar-se com o sistema “Project Monitor” através do protocolo HTTP GET. O “Project Monitor” devolve a lista de projetos ativos. Cada projeto tem os seguintes campos: id (inteiro), nome (string), ativo (boolean), descrição (string) e chave (string). | MUST | RF01 |
| 2.1 | Ligação ao “Redmine” | Criação de Projetos | O sistema deve permitir criar projetos no sistema “Redmine” através do protocolo HTTP POST e envio dos seguintes campos: id (inteiro), nome (string), is\_public (boolean) e descrição (string) | MUST | RF02 |
| 2.2. | Consultar Projetos | O sistema deve permitir conectar-se com o sistema “Redmine” através do protocolo HTTP GET. O “Redmine” devolve a lista de projetos ativos. Cada projeto tem os seguintes campos: id (inteiro), nome (string), is\_public (boolean) e descrição (string). | COULD HAVE | RF03 |
| 2.3 | Apagar projetos | O sistema deve permitir apagar projetos do “Redmine” através do protocolo HTTP DELETE e envio do id do projeto. | COULD HAVE | RF04 |
| 2.4 | Login na plataforma “Redmine” | O sistema deve permitir fazer login no sistema “Redmine”. Após o login será devolvido um *token* de sessão que terá de ser usado no RF02, RF03 e RF04. | MUST | RF05 |

Tabela - Requisitos Funcionais

### Requisitos Não Funcionais

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | TAG | Requisito | Descrição | Prio. | Tipo R |
| 1.1 | Produto” | Portabilidade | O sistema deve funcionar em sistemas operativos Windows e Linux. | MUST | RNF01 |
| 1.2 | Eficiência | O sistema deve ser capaz de processar vários pedidos em paralelo | MUST | RNF02 |
| 1.3 | Confiabilidade | O sistema deverá funcionar sem problemas indefinidamente. | MUST | RNF03 |
| 2.1 | Organizacionais | Implementação | A plataforma deverá ser desenvolvida em PHP utilizando a Framework *Symfony* | MUST | RNF04 |
| 2.2 | Padrões | A plataforma deverá utilizar programação orientada a objetos e no formato MVC (Model-View-Controller) | MUST | RNF05 |
| 3.1 | Externos | Interoperabilidade | O sistema deverá permitir obter projetos do sistema Project Monitor | MUST | RNF06 |
| 3.2 | Interoperabilidade | O sistema deverá permitir criar projetos no sistema Redmine | MUST | RNF07 |
| 3.3 | Éticos | O sistema não deverá permitir aceder a informação sensível da empresa a utilizadores não credenciados | MUST | RNF08 |

Tabela - Requisitos Não funcionais

## Metodologias Aplicadas

O desenvolvimento do projeto foi feito através da definição de reuniões semanais com a equipa onde era avaliado o ponto de situação e definidas as próximas tarefas. Como resultado destas reuniões, eram criadas tarefas no Jira onde seria feita a monitorização do estado e os resultados. Neste caso a monitorização das tarefas era feito por um engenheiro da empresa que respondia ao orientador deste estágio.

O código foi desenvolvido utilizando o git, sendo que as entregas eram feitas utilizando o método de *Pull-Request*, onde o engenheiro que monitorizava a execução das tarefas fazia *review* do código e aprova as alterações sugeridas.

## Diagrama de casos de uso

O diagrama apresentado na Figura 9 ilustra a funcionalidade desenvolvida e as suas interfaces e dependências. O objetivo deste projeto é criar uma API que permita sincronizar o *Project Monitor* (sistema interno de gestão do projetos da Fiducial) e o Redmine [3]. Este sincronismo é basicamente a leitura de todos os projetos do *Project Monitor* e armazenar no Redmine. A única forma de executar a sincronização é através da execução de comandos na *bash*, o que justifica assim que o utilizador desta funcionalidade apenas possa ser um *Software Developer* interno na Fiducial.

Uma imagem com texto, captura de ecrã, diagrama, Tipo de letra

Descrição gerada automaticamente

Figura - Diagrama de caso de uso da solução desenvolvida

## Diagrama de classes

O diagrama apresentado na Figura 10 representa as estruturas de dados em cada um dos sistemas: Redmine e *Project Monitor*. Este diagrama de classes não dispõe de nenhuma informação porque cada uma destas classes encontra-se localizadas em sistemas diferentes. O objetivo da solução desenvolvida é de ler a estrutura de dados do Project Monitor e converter para a estrutura de dados do Redmine.

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

Figura - Diagrama de referentes ao Redmine e Project Monitor

## Diagramas de sequência

Como referido anteriormente, a funcionalidade principal do sistema é de sincronizar projetos do sistema *ProjectMonitor* com o *Redmine* [3]. O processo inicia com um operador executar um HTTP POST /create-project que será recebido pelo *APIController*. Este seguidamente, irá ler todos os projetos do *ProjectMonitor* e posteriormente irá fazer HTTP POST ao sistema de Redmine configurado com a informação retirada do *ProjectMonitor*. Após este processo ser concluído, o operador receberá a confirmação do HTTP POST com o código 200 (sucesso).

Uma imagem com texto, diagrama, file, Paralelo

Descrição gerada automaticamente

Figura - Diagrama de sequência da interação entre os diferentes atores

# Implementação

Neste capítulo são abordados os detalhes de implementação do solução. Aqui será apresentada a arquitetura do sistema e documentados todos os detalhes de implementação. O sistema desenvolvido é maioritariamente distribuído e permite fazer a sincronização de informação de dois locais com características e modelos de dados distintos.

É importante reter que foi utilizada a tecnologia de *containers* do Docker quer permitiu isolar o sistema *Redmine* [3].

## Arquitetura

A arquitetura da solução encontra-se ilustrada na Figura 12. O sistema é constituído por dois componentes de alto nivel: O *localhost* (ou seja, a máquina onde corre este projeto) e o “Fiducial Servidor de teste” que é uma máquina que simula um servidor da Fiducial. No *localhost,* o sistema é decomposto no componente do “*Project Monitor”* e em um container Docker que providencia um sistema de Redmine [3].

De forma a entrar no detalhe dos componentes, é importante reter que a linguagem utilizada para o desenvolvimento da solução foi o PHP com a *framework* *Symfony*, como documentada na secção 2.5. Esta *framework* usa como padrões de desenvolvimento de Software MVC (*Model*, *View/Service* and *Controller*). Esta característica justifica o porquê do *Component* do “*ProjectMonitor”* deter como subcomponentes:

* *Service* – Responsável por gerir a receção de pedidos HTTP
* *Model* – Responsável por gerir o modelo de dados interno da aplicação Web
* *Controller* – Responsável pelas ações de controlo e lógica. Neste caso, é responsável por aceder ao *Project Monitor* e *Redmine* respetivamente.

Uma imagem com texto, diagrama, Esquema, Retângulo

Descrição gerada automaticamente

Figura - Diagrama de componentes do sistema e interfaces

Como ilustrado na Figura 13, a estrutura de ficheiros da solução segue a arquitetura proposta. Detalhadamente, a estrutura MVC encontra-se localizada na pasta *src*.

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Figura - Estrutura do projeto (Symfony)

A Tabela 3 apresenta a descrição e listagem dos módulos que realizam a arquitetura. Estes, são os mesmos ficheiros apresentados na estrutura do projeto ilustrada na Figura 13.

Tabela - Listagem e descrição dos módulos do sistema

|  |  |  |
| --- | --- | --- |
| Modulo | Tipo | Descrição |
| ApiController.php | Controlador | Responsável por disponibilizar o serviço post “create-project”. Este é o ponto de entrada da execução da sincronização |
| ProjectMonitor.php | Modelo | Modelo de dados do *Project Monitor* |
| Redmine.php | Modelo | Modelo de dados do *Redmine* |
| ProjectMonitorService | Serviço | Responsável pela ligação ao *Project Monitor,* fazer a leitura dos projetos e criar o modelo de dados respetivo |
| RedmineService | Serviço | Responsável pela ligação ao *Redmine* e que permite criar projetos, traduzindo assim o modelo de dados do *Project Monitor* para o *Redmine*. |

## Apresentação da solução desenvolvida

A solução desenvolvida pode ser explicada em etapas sequenciais. A primeira fase, é a ligação ao *Project Monitor* para obtenção dos projetos em um formato JSON. A segunda fase, é a conversão dos dados recebidos em um modelo interno de dados que os representa. A terceira fase, é a conversão do modelo de dados do *Project Monitor* para um modelo de dados compatível com o sistema *Redmine*. Por último, a quarta fase é a criação dos projetos no Redmine, utilizando o modelo de dados criado na fase anterior. De reter que, esta última fase garante que se um projeto já existir no sistema Redmine uma mensagem de erro é retornada e o projeto existente não é alterado.

### Extração dos projetos do Project Monitor

Como descrito em 4.2, a primeira e segunda fase são responsáveis por obter e criar os modelos dos projetos existentes no *Project Monitor*. Na execução da primeira fase, as estruturas dos dados recebidos são idênticas aos apresentados **Erro! A origem da referência não foi encontrada.**.

Uma imagem com texto, captura de ecrã, Tipo de letra, documento

Descrição gerada automaticamente

Na Tabela 4 encontra-se a descrição de cada um dos campos.

Tabela - Descrição dos campos enviados pelo Project Monitor

|  |  |
| --- | --- |
| Chave | Descrição do campo |
| “id” | Identificador único do projeto em formato inteiro |
| “libelle” | Nome/titulo do projeto |
| “actif” | Variável que define se o projeto se encontra ativo ou não |
| “description” | Descrição do projeto |
| “key” | Chave do projeto. Igual a “id”, mas em formato string. |

É importante dar enfase ao campo “id”/”key”. Este é um valor único e que será também utilizado para identificar os projetos no Redmine. Este valor cria uma ponte lógica na sincronização entre os projetos do *Redmine* e *Project Monitor*.

### Criação de Projetos no Redmine

Após a execução com sucesso da primeira e segunda fase (4.2.1), todos os dados armazenados serão convertidos para a estrutura de dados que o *Redmine* suporta. A Figura 14 apresenta um pedido utilizando o *PostMan (2.6*) ao serviço do *Redmine*. O modelo de dados do Redmine encontra-se definido na parte superior, no campo Body do HTTP Post request.

Uma imagem com texto, captura de ecrã, software, Página web

Descrição gerada automaticamente

Figura - HTTP POST ao Redmine

É importante definir nesta fase, qual a ligação entre os dados recebidos pelo *Project Monitor* e o que é enviado para o *Redmine*. A Tabela 5 apresenta a ligação entre os campos/chaves dos dois sistemas.

Tabela - Mapeamento entre as chaves do Redmine e Project Monitor

|  |  |
| --- | --- |
| Chave Redmine | Chave Project Monitor (Tabela 4) |
| “name” | “libelle” |
| “identifier” | “key” |
| “description” | “description” |
| “is\_public” | “actif” |

Um pré-requisito que não foi mencionado anteriormente necessário na comunicação com o Redmine é a definição do campo “X-Redmine-API-Key”. Este campo é um identificador e é descrito como API key [15]. Este campo tem que estar definido no Headers do HTTP POST, como apresentado na Figura 15.

Uma imagem com texto, captura de ecrã, software, Página web

Descrição gerada automaticamente

Figura - Configuração da API key e resultado

## Análise de resultados

Para executar todo o processo documentado em 4.2, é necessário enviar um pedido de HTTP POST para o Webserver do projeto. No caso em concreto do exemplo ilustrado na Figura 16, o serviço encontra-se em http://localhost:8000/create-project.

Uma imagem com texto, captura de ecrã, número, software

Descrição gerada automaticamente

Figura - Execução do processo de sincronização com sucesso

Após a execução do serviço, o projeto retornou uma lista vazia de erros (Figura 16, na parte inferior, Body).

Para verificar a veracidade do processo, a página do Redmine pode ser aberto como ilustrado na Figura 17, e verificamos que os projetos se encontram criados.

Uma imagem com texto, captura de ecrã, software, Página web

Descrição gerada automaticamente

Figura - Verificação da sincronização dos projetos no Redmine

No caso do processo ilustrado na Figura 16 ser executado novamente, serão retornados vários erros mencionando que os projetos já existem. Neste caso, não é feita qualquer sincronização nos projetos já existentes. Este comportamento encontra-se ilustrado na Figura 18.

Uma imagem com texto, captura de ecrã, software, Página web

Descrição gerada automaticamente

Figura - Gestão de sincronismo no caso de os projetos já existirem

# Conclusão

## Apreciação Final

## Trabalho Futuro

Criar assincronamente projetos quando for atualizada alguma coisa no project monitor

Bibliografia

|  |  |
| --- | --- |
| [1] | Fiducial Informática, “About Fiducial,” [Online]. Available: https://fiducial.com/about. |
| [2] | Symfony, “Symfony Framework,” [Online]. Available: https://symfony.com/. |
| [3] | Redmine, “Redmine,” [Online]. Available: https://www.redmine.org/. |
| [4] | PHP, “PHP,” [Online]. Available: https://www.php.net/. |
| [5] | Docker, “Docker,” [Online]. Available: https://www.docker.com/. |
| [6] | Postman, “Postman,” [Online]. Available: https://www.postman.com/. |
| [7] | Wikipedia, “Hypertext Transfer Protocol,” [Online]. Available: https://pt.wikipedia.org/wiki/Hypertext\_Transfer\_Protocol. |
| [8] | Github, “Github,” [Online]. Available: https://github.com/. |
| [9] | Git, “Git,” [Online]. Available: https://git-scm.com/. |
| [10] | Jetbrains PHP Storm, “PHP Storm,” [Online]. Available: https://www.jetbrains.com/phpstorm/. |
| [11] | Symfony, “Symfony,” [Online]. Available: https://symfony.com/. |
| [12] | XAMPP, “XAMPP,” [Online]. Available: https://www.apachefriends.org/. |
| [13] | Apache Friends, “XAMPP,” [Online]. Available: https://www.apachefriends.org/. |
| [14] | Intelegain, “Waterfall Development Model,” [Online]. Available: https://www.intelegain.com/waterfall/. |
| [15] | Redmine, “How to setup a Redmine API key,” [Online]. Available: https://www.redmine.org/boards/2/topics/53956/. |

ANEXOS

Uma imagem com texto, captura de ecrã, documento

Descrição gerada automaticamente

Figura - Anexo, Tarefa PMIR-1

Uma imagem com texto, captura de ecrã, software, Tipo de letra

Descrição gerada automaticamente

Figura - Anexo, Tarefa PMIR-2

Uma imagem com texto, Tipo de letra, software, Página web

Descrição gerada automaticamente

Figura - Anexo, Tarefa PMIR-3

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Anexo, Tarefa PMIR-4

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Anexo, Tarefa PMIR-5

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Anexo, Tarefa PMIR-6

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Anexo, Tarefa PMIR-7

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Anexo, Tarefa PMIR-8

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Anexo, Tarefa PMIR-9

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Anexo, Tarefa PMIR-10