

INSTITUTO POLITÉCNICO DE TOMAR

ESCOLA SUPERIOR DE TECNOLOGIAS DE TOMAR

Repositório de Projetos do IPT

**Mestrado em Engenharia Informática – Internet das Coisas**

Engenharia de Software

2016/2017

**Autores  
*António Paulo Rodrigues***  
*<estt2208@ipt.pt> - nº 2208*

***Miguel Filipe Redol Cotralha Inácio Coelho***  
*<aluno18284@ipt.pt> - nº 18284*

***Nelson Miguel Pereira Gomes****<estt16037@ipt.pt> - nº 16037*

***Pedro André Casimiro Nunes***  
*<aluno16811@ipt.pt> - nº 16811*

***Pedro Henriques Dias***  
*<aluno17108@ipt.pt> - nº 17108*

***Pedro Miguel Pinheiro De Matos***  
*<aluno13441@ipt.pt> - nº 13341*

***Vasco Manuel Joaquim Marques***  
*<estt11046@ipt.pt> - nº 11046*

**Orientador  
*Renato Panda****<renato.panda@ipt.pt>*

# Resumo

Este documento documenta o projecto “Repositório de Projectos do IPT”, realizado no âmbito da disciplina de Engenharia de Software, Mestrado de Engenharia Informática (*Internet of Things*) do Instituto Politécnico de Tomar. Nesta disciplina foram introduzidas metodologias ágeis de desenvolvimento de *software,* assim como tecnologias e conceitos novos, tais como Ruby, Rails, mapeamento de objectos relacionais (ORM), ambientes de desenvolvimento virtuais, testes automatizados e ferramentas colaborativas, que serviram de base para o desenvolvimento do projecto.

## Sumário

O objectivo deste trabalho foi criar um repositório de projectos académicos que agregue todos os trabalhos desenvolvidos pelos alunos do IPT. Pretendeu-se com isto dotar a instituição de uma plataforma web que sirva de ponto central para quem quiser conhecer o que tem sido feito dentro da instituição, aumentando assim a sua exposição para o exterior.

O trabalho foi realizado utilizando metodologias SCRUM, sendo dividido em 1+4 iterações. A primeira iteração serviu para contacto com ferramentas e linguagens novas (Ruby) e para definição da plataforma a implementar. Esta iteração deu origem a um conjunto inicial de “*user stories*” que foram sido implementadas nas quatro iterações seguintes, em conjunto com novas histórias que foram surgindo. O projecto foi desenvolvido em Ruby 2.3, usando a framework Ruby on Rails 5. Segue o paradigma Model-View-Controller, tendo recorrido às frameworks / bibliotecas bootstrap 4 e jQuery para um interface moderno e adaptado a diversos dispositivos – responsivo.

O projecto e respectivo código-fonte estão disponíveis na página deste e no repositório respectivo.

## Summary

Lorem ipsum… o mesmo mas em inglês ☺

----- ui ----- ainda falta ----

**Palavras-chave**

Repositório, projectos, Ruby, Ruby on Rails, Bootstrap, responsive

Índice

[Resumo 1](#_Toc472457963)

[Sumário 2](#_Toc472457964)

[Summary 2](#_Toc472457965)

[Lista de Acrónimos e Abreviaturas 4](#_Toc472457966)

[1 Introdução 5](#_Toc472457967)

[1.1 Desenvolvimento Ágil 5](#_Toc472457968)

[1.2 Padrão Model-View-Controller (MVC) 7](#_Toc472457969)

[1.3 Ruby on Rails 9](#_Toc472457970)

[1.4 Bootstrap 4 e jQuery 9](#_Toc472457971)

[2 Desenvolvimento da Aplicação 10](#_Toc472457972)

[2.1 Planeamento (Iteração 0) 11](#_Toc472457973)

[2.2 Desenvolvimento (Iterações 1 a 4) 13](#_Toc472457974)

[2.2.1 Estrutura do Projecto 16](#_Toc472457975)

[2.2.2 Bibliotecas Utilizadas (Gems) 18](#_Toc472457976)

[2.2.3 Modelos 20](#_Toc472457977)

[2.2.4 Controladores 22](#_Toc472457978)

[2.2.5 Vistas 23](#_Toc472457979)

[2.2.6 Como experimentar / colaborar? 25](#_Toc472457980)

[3 Conclusão 26](#_Toc472457981)

[3.1 Trabalho futuro 27](#_Toc472457982)

[4 Referências 28](#_Toc472457983)

# Lista de Acrónimos e Abreviaturas

**IPT** - Instituto Politécnico de Tomar

**CEO** – Chief Executive Officer (Diretor Executivo)

**PO** - Product Owner

**Freeware** – Software livre de utilização

**TOP12** – Posição mais elevada numa lista / Herança / Classificação

**RoR** – Ruby on Rails

**Framework** – Estrutura que une código num projeto de software

**MVC** – Model-View-Controller

**Feeds –** Informação sobre novos projectos

**Tags –** Referencias a palavras contidas nos projectos inseridos

**VBox** – Virtualização de sistemas operativos

# Introdução

No decorrer das aulas de engenharia de software, esta equipa de alunos, ficou encarregue de desenvolver uma aplicação web, onde iria ficar os projectos desenvolvidos pelos os alunos, do IPT ou alargado a outras escolas. Estes projetos organizacionais são constantes, recai o problema de onde os guardar e como o publico alvo pode pesquisar sobre os desenvolvimentos efetuados.

É neste problema que nos propusemos desenvolver uma plataforma vanguardista na resolução destas questões levantadas e observadas. As questões mais relevantes, propõem uma oportunidade única, demonstrando o conhecimento adquirido nas escolas, e em particular no IPT, podendo ser alargada a outras escolas de ensino. Observou-se que não existe a possibilidade dos orientadores e alunos guardarem os seus trabalhos.

A estrutura vai disponibilizar via internet, todos os estudos efetuados dos projetos, deixando os contatos dos desenvolvedores patentes e associados ao seu projeto.

## Desenvolvimento Ágil

Nos conceitos existentes hoje em dia, o desenvolvimento de software cada vez mais segue novos padrões, deixando de ser um desenvolvimento rígido de inicio e fim, em que o software é apresentado ao cliente como uma ferramenta completa. As tendências apontam para grandes ganhos no desenvolvimento ágil, podendo assim criar um software de inicio e fim com a modulação e estrutura em crescimento constante em cada etapa definida pela equipa de desenvolvimento e pelo seu requisitante.

Nestes termos modernos e testados é nesta base que nos propomos a desenvolver o projeto de repositório dos projetos desenvolvidos nas escolas.

O desenvolvimento ágil recai sobre um conceito flexível de desenvolvimento, más com prazos curtos de apresentação dos trabalhos desenvolvidos, no entanto mais eficaz na sua metodologia ágil, esta metodologia escolhida fui as *Scrum* que recai sobre;

O *Scrum* é sem duvida, o método ágil mais usado nos nossos dias, focando na gestão de projetos e com base no planeamento interativo e incremental, estas desdobram-se em fases em se dá o nome sprints.

O projeto desde o inicio, define-se a lista de funcionalidades a serem desenvolvidas, chamadas de “*product backlog*”, cada desta se trona uma “sprint” que cada desta passa a designar “*sprint backlog*”, são atividades distribuídas entre os membros da equipa, que as desenvolvem dentro de um prazo estabelecido. No fim de cada “Sprint” é feita uma reunião do que foi desenvolvido e o que ficou por desenvolver, esta chamada “*Sprint review meeting*”. Após esta reunião começam a planear a próxima “*Sprint*” e assim continuamente até chegar ao fim do desenvolvimento.

No desenvolvimento ágil existem três (3) papeis fundamentais;

- **Product Owner**: é um CEO do produto, este membro faz a ligação entre o cliente e “Scrum Team”, e é o responsável pelo projeto.

- **Scrum Master**: é basicamente o “Coach” da equipa, o orientador e responsável da equipa de desenvolvimento e motivador.

- **Scrum Team**: equipa de desenvolvimento do projeto.

Funcionamento do *Scrum*

***Sprint*** – são interações de 2 a 4 semanas, em cada uma delas inicia-se com uma reunião “*Sprint Planning Meeting*”, onde são decididos o que será desenvolvido, com a apresentação dos itens com maior prioridade e são divididos pela equipa.

***Scrum Team*** - todos os dias iniciam com uma reunião até 15 minutos, para sincronizar o trabalho a ser desenvolvido e informa o “*Scrum Master”* de algum impedimento no desenvolvimento.

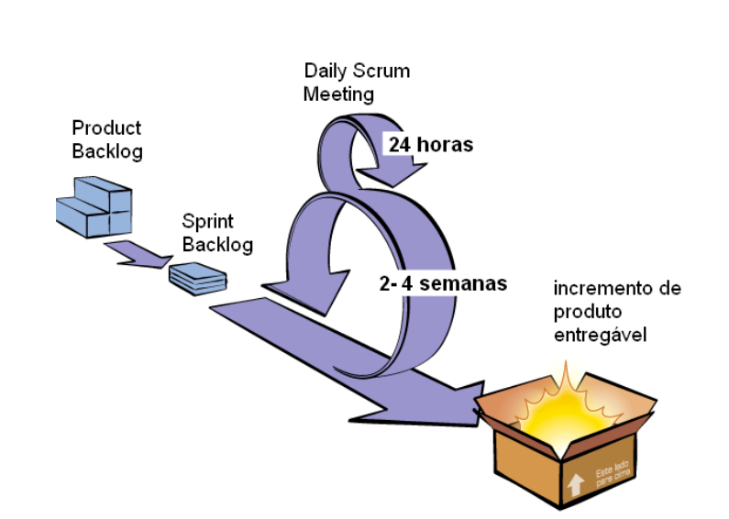


fig.1 – Processo *Scrum*

No fim de cada “*Sprint”* ocorre a “*Sprint Review Meeting”*, reunião na qual o “*Srcum Team”* apresenta para PO o que foi desenvolvido durante a “*Sprint”*.

Após esta, o *Scrum* Master efetua uma reunião com o *Scrum* Team numa ultima reunião da “*Sprint”* e faz uma respetiva do desenvolvimento (“*Retrospective Meeting*”), motivando a equipa para a próxima Sprint e melhorias a fazer.

***Vantagens:***

**- Motivação maior dos programadores**, tem metas para atingir.

**- Visualização do projecto**, fica mais fácil toda a equipa visualizar o projecto.

**- Diminuição dos bugs**, a qualidade passa a ser o mais importante nos prazos, logo existe uma maior preocupação em não falhar na programação.

**- Prioridades podem ser alteradas**, com as tarefas definidas em cada interação, podem ser alteradas prioridades na próxima interação, desde que faça sentido ou a pedido do requisitante.

**- Funcionalidade que agregam valor vêm em primeiro**, um dos itens que independente da metodologia pode ser aplicado, contudo, no Scrum esse critério é bastante enfático, pois o que agrega maior valor ao cliente.

***Desvantagens:***

- **Projecto não documentado**, pelos prazos de desenvolvimento a documentação fica para segundo plano ou mesmo anulada.

**- Prazo**, como nesta metodologia os prazos são muito apertados, e o cliente quer resultados, às vezes a implementação ou o desenvolvimento é incompleto, no ponto de vista do desenvolvedor, porque fica sempre algo por melhorar ou fazer.

- **Falta de planeamento de papel**, em vista de não haver tempo para fazer desenhos, os desenvolvedores arrancam com o desenvolvimento sem um layout ou estrutura definida ou consistente.

**- Cargos indefinidos**, em Scrum existem três (3) papeis definidos com base, muitas das vezes esses papeis / cargos não funcionam, pois existe prazos curtos e todos têm de ajudar, e os cargos muitas das vezes alteram-se.

Neste perfil, a nossa equipa segui-o a metodologia ágil apresentada.

## Padrão Model-View-Controller (MVC)

MVC é uma arquitetura de software separado por três (3) camadas, os “*View*” a camada de interação do utilizador, os “*Model*” a camada de manipulação de dados e a camada de controlo são os “*Controller*”.

“**View**” – paginas apresentadas ao utilizador, com a exibição dos dados.

“**Model**” – Manipulação dos dados, leitura e escrita dos dados e suas validações.

“**Controller**” – Responsável por receber os pedidos do utilizador, controla qual o “model” a usar e qual “view” será mostrada ao utilizador.



fig.2 – Diagrama MCV

Com o aumento da complexidade das aplicações desenvolvidas, com uma programação orientada a objectos, torna-se importante a separação entre dados e a apresentação das aplicações. Assim as alterações feitas ao “layout” não afetam a manipulação de dados, e estes dados podem também ser reorganizados sem alterar o “layout”. O sistema fica mais organizado quando está estruturado em MVC, ficando mais simplificado a compreensão da sua arquitetura. Outro ponto importante do MVC é a segurança da transição de dados, através da camada de “controller” é possível evitar que qualquer dado inconsistente chegue à camada de “model” para persistir na base de dados. Por outras palavras podemos imaginar o “controller” ser um espécie de “Firewall”, se o utilizador enviar dados, a camada “controller” responsabiliza-se por bloquear esses dados que venham a causar inconsistência.

***Vantagens:***

- Utilização de padrões e camadas

- Utilização de funções pré-construídas

- Escalável e portável

- Múltiplos visualizadores

- Segurança

- Documentação

- Comunidade de suporte

- Facilidade de integração de novos membros na equipa.

***Desvantagens:***

- Precisa de treino adequado na Framework

- A equipa precisa de ser multo disciplinada

- Requer mais tempo de planeamento

- Por ser *open-source*, pode ser explorado vulnerabilidades de uma forma rápida.

- Complexidade para projectos pequenos.

## Ruby on Rails

O “*Ruby*”[1], uma linguagem de programação que nasceu em meados do ano 1995 no Japão, tendo o seu criador “*Yukihiro Matsumoto*”, pensado numa linguagem dinâmica em “*Script*”, inspirada em outras linguagens existente o “*Ruby”* vem mostrar o seu poder de resolução de problemas conhecidas nas existentes, tais como, todas as variáveis são objetos, os métodos de geração de código são em tempo real, utiliza “*RubyGems*” que são bibliotecas desenvolvidas para facilitar e interagir com a aplicação, utiliza herança múltipla e dinâmica. Estando esta no TOP12 das linguagens segundo índice “Tiobe”[2].

fig.3 – Logotipo Ruby

O RoR é uma “*Framework* *freeware*”, desenvolvida com base no padrão de arquitetura MVC, que assenta na linguagem “*Ruby”*, com o seu criador “*David Heinemeier Hansson*”, num projeto desenvolvido chamado “*BaseCamp*” foi lançado em miados de 2003.

Existem muitas MVC no desenvolvimento de aplicações para a internet, más o RoR, ganhou mercado na sua facilidade de desenvolvimento e na sua velocidade de desenvolvimento, com uma estrutura pré-definida, e com as bibliotecas como suporte (gems), veio trazer a esta “*Framework”* um poder enágua e uma posição vanguardista. Esta assenta em conceitos bastantes simples, “*Active Record, Action Pack, Action Mailer, Active Suport, Action WebService*”.

## Bootstrap 4 e jQuery

Foram utilizadas ferramentas de auxilia ao design do layout e funcionalidades, estas com características especificas que ermos descrever.

***Bootstrap***

O Bootstrap é uma *Framework front-end*, que facilita a vida dos desenvolvedores de aplicações web, com uma tecnologia *responsivo* para os dispositivos. Esta Framework permite ter um layout desejado sem escrever uma única linha de CSS, fazendo as chamadas no código HTML. O Bootstrap possui uma diversidade de componentes (Plugins) em javascript (jQuery) que ajudam o designer a implementar, *tootlip*, *menu-dropdown*, *modal*, *carousel*, *slideshow*, entre outos, com as maiores facilidades, fazendo algumas configurações e chamadas ao código. Assim o desenvolvedor não necessita de escrever scripts na nua aplicação.

A nossa aplicação foi desenvolvida com suporte a esta Framework, podendo dar um design mais ajustado ao layout.

***jQuery***

O jQuery é uma biblioteca de *javascript* cross-browser, desenvolvida para simplificar o código do lado do cliente, que integram o HTML. Podemos chamar uma Framework de forma resumida, esta tem uma coleção de funções e métodos prontos para serem utilizados de forma pré-definida. O jQuery permite de forma dinâmica e fácil criar menus, efeitos nas páginas, chamadas ao AJAX para pesquisa e inventos associados a botões, entre outros.

# Desenvolvimento da Aplicação

O desenvolvimento foi divido em 1 + 4 iterações detalhadas de seguida. A primeira iteração serviu para criar uma visão geral do projecto pretendido e planeamento do mesmo. Pretende-se com a aplicação um repositório documental, logo estruturamos o trabalho pela equipa da seguinte for;

A equipa foi composta pelos alunos e suas tarefas:

- **António P. Rodrigues**

- Programador da estrutura para upload files e Facebook new  
  
- **Miguel F. R. l. C. I. Coelho**

- Programador da estrutura para Escolas, Cursos, documentos e ligações entre tabelas.

- **Nelson M. P. Gomes**

- Design do Layout, gestor do github  
  
- **Pedro A. C. Nunes**

- Design do Layout, teste app  
  
- **Pedro H. Dias**

- Design da base de dados, participantes  
- **Pedro M. P. De Matos**

- Programador Estrutura projectos, Users, Administração e ligações entre tabelas.  
  
- **Vasco M. J. Marques**  
 - Programador da estrutura para tags, facebook new

## Planeamento (Iteração 0)

A interação inicial (0), sérvio para a equipa organizar o seu trabalho e verificar os requisitos necessários para se iniciar o desenvolvimento. Nesta reunião inicial, obteve-se o consenso e verificou-se as dificuldades de alguns requisitos, permitindo assim retirar os requisitos que consideramos mais importantes para a 1ª interação. Nesta mesma reunião saiu o primeiro esboço do layout, a primeira visão da base de dados a ser criada com os seus atributos e ligações possíveis. Assim considerou-se;

**Requisitos Funcionais**  
Os requisitos funcionais foram levantados em função do requerente e obteve-se;

1. Permitir o utilizador autenticar-se com o email da instituição, deverá ser identificado se é da escola superior de tecnologia de tomar ou de Abrantes ou da escola de gestão de tomar.
2. O sistema deverá permitir a um utilizador a recuperação da password de acesso à sua conta.
3. O sistema deverá permitir ao utilizador a alteração dos seus dados pessoais (password incluída).
4. Eu quero que na página inicial apareça os projetos mais novos em primeiro.
5. Eu quero poder ser relembrado se esquecer do nome do utilizador e password.
6. Eu quero poder criar o meu próprio utilizador (user).
7. Eu quero poder mudar a password.
8. Eu quero que os visitantes possam dar estrelas aos projetos.
9. Eu quero poder dar autorização a determinados utilizadores, para poderem baixar o arquivo pdf.
10. Eu quero que as pessoas que visita poderem comentar os projetos.
11. Eu quero poder pesquisar por curso.
12. Eu quero poder pesquisar pela data ou ano.
13. Eu quero poder pesquisar pelo nome do projeto.
14. Eu quero poder modificar projetos inseridos.
15. Eu quero poder eliminar projetos inseridos.
16. Eu quero um Design atrativo.
17. Eu quero inserir projetos feitos no IPT
18. Eu quero fazer login com as contas das redes sociais.
19. Eu quero fazer login com a conta do IPT.
20. O Utilizador ao Registar o Projeto deve incluir numa Área de Projeto Existente (Informática, Eletricidade, Design...) ou tag.
21. Eu quero poder pesquisar projetos na página principal.
22. O Utilizador Registado deve poder criar novos projetos.
23. O Utilizador deve poder gerir comentários aos seus projetos.
24. O Utilizador deve poder consultar os dados pessoais.
25. Uma página de pesquisa de projectos? Em que seria feita por ano lectivo, nome aluno, numero aluno, etc.
26. O Utilizador deve poder fazer logout.
27. Só utilizadores registados podem ver os projectos e os nao registados ficam pela pagina inicial e log in.
28. Utilizadores não registados podem ver os projetos todos.
29. A pagina inicial do site deve de mostrar uma pequena descrição do site com botão efetuar login.
30. Eu quero poder associar o projeto a tag’s.
31. Eu como utilizador registado quero poder fazer upload de vídeos, fotos para associar aos projetos.

[**Requisitos Não-Funcionais**](http://www.profissionaisti.com.br/2013/02/a-importancia-dos-requisitos-nao-funcionais/)  
Os requisitos identificados para esta aplicação web foram;

**Confiabilidade**  
Ser responsivo com os dispositivos atuais, em particular com os moveis.

**Portabilidade**  
A estrutura ser fácil de migrar para outro sistema operativo / servidor.

**Segurança**  
Ter a segurança que permita só as pessoas registadas possam inserir, modifica e eliminar projectos.

**Usabilidade**  
A aplicação desenvolvida seja fácil de usar, que não necessite de formação especifica para o utilizador registado insira um projeto ou o modifique e mesmo o elimine.

Desenho do Layout inicial:

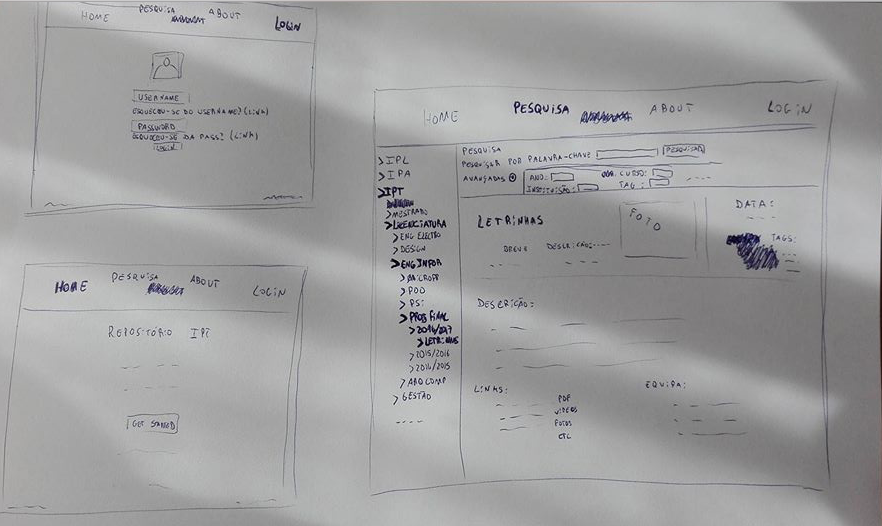


fig.4 – Layout inicial

Desenho da base de dados inicial:

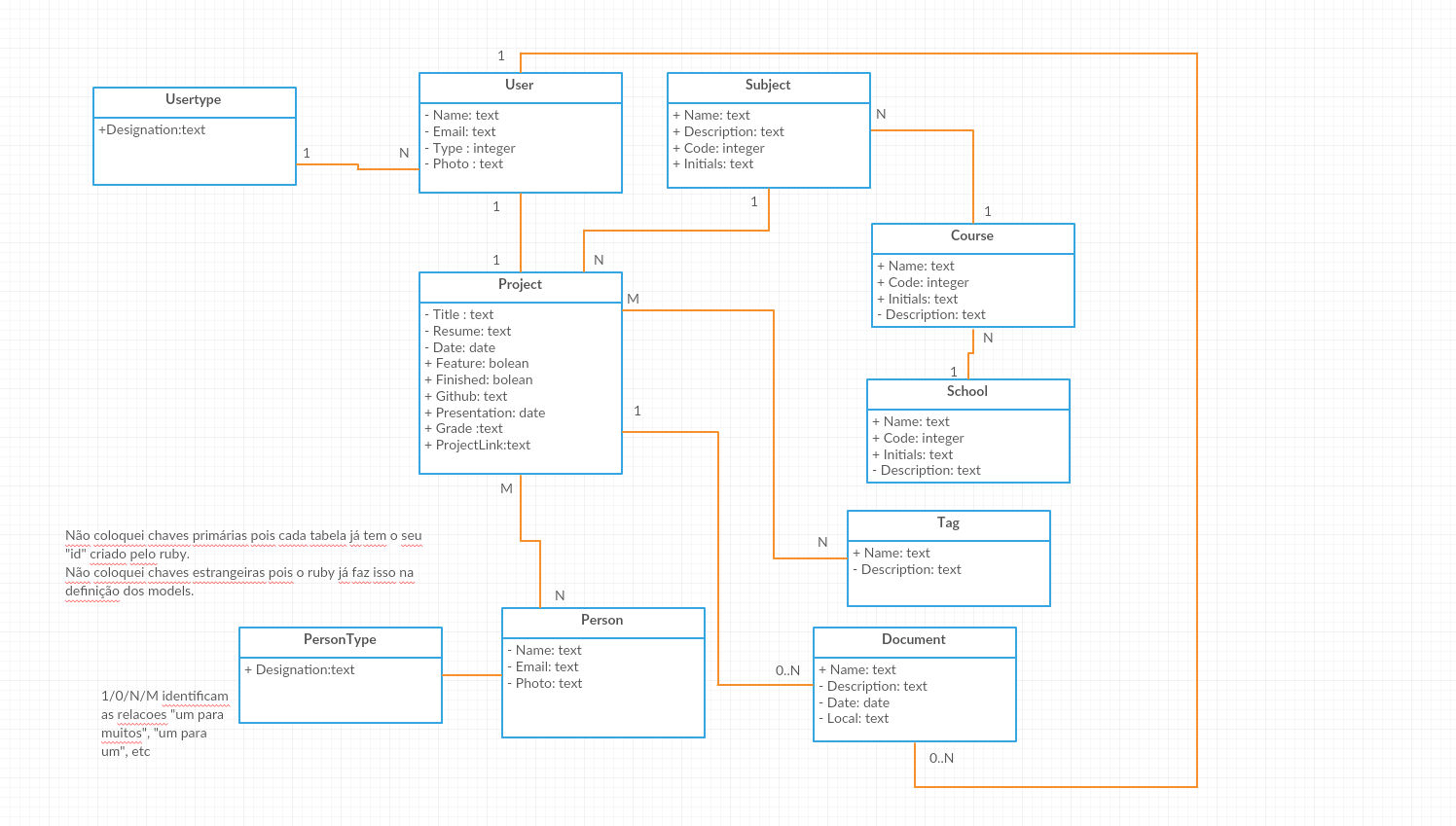


fig.5 – Diagrama da base de dados inicial

## Desenvolvimento (Iterações 1 a 4)

O desenvolvimento decorreu durante o resto do semestre, sendo dividido em 4 iterações de cerca de duas semanas cada. Entre cada iteração foram feitas as reuniões de scrum típicas (revisão, retrospectiva, planeamento – numa linha o que se faz em cada uma delas). No início de cada aula prática foi ainda feita a típica “Daily Meeting” para cada membro informar a equipa do que fez na semana anterior e o que iria fazer na semana seguinte.

Com os requisitos identificados iniciamos as interações, estas definidas com o desenvolvimento ágil (“*Scrum*”), definimos as nossas interações (“*Sprint”);*

*Requisitos 1ºSprint:*

Eu quero ver os projetos sem estar registado.

Eu quero um sistema de inserir projetos fácil e simples.

Eu quero que sejam apresentados na página inicial os projetos mais recentes.

Eu quero poder fazer logout.

Eu quero como utilizador registado poder criar novos projetos.

Eu como utilizador registado quero poder modificar projetos.

Eu como utilizador devo poder pesquisar projetos independentemente de este estar registado ou não.

Eu como utilizador normal quero ver as páginas sem login.

***1º - Sprint:***

- Desenvolver / estudar a base de dados em papel.

- Criar sistema de Virtualização com VBox e Vagrant.

- Instalar o RoR

- Criação da estrutura de projetos

- Criação da estrutura escolas

- Criação da estrutura cursos e Disciplinas

- Criação da estrutura users

*Requisitos 2ºSprint*

Eu como utilizador registado quero poder recuperar a password da minha conta.

Eu como utilizador registado quero poder alterar os meus dados pessoais.

Eu como utilizador quero ver os projetos mais recentes primeiro, na página inicial.

Eu como utilizador quero poder ver as classificações dados pelas pessoas aos projetos.

Eu como utilizador quero poder ver as classificações dados pelas pessoas aos projetos.

Eu como utilizador registado quero poder alterar a minha password.

Eu como utilizador registado quero poder criar projetos.

Eu como utilizador registado quero poder adicionar vídeos e fotografias aos projetos.

Eu como utilizador quero poder pesquisar por curso.

Eu como utilizador quero poder pesquisar por tag´s.

Eu como utilizador quero poder pesquisar pela data ou ano do projeto.

Eu como utilizador registado quero poder eliminar projetos.

Eu como utilizador registado quero poder modificar projetos inseridos.

Eu como utilizador registado quero poder eliminar projetos.

Eu como utilizador quero um design atrativo.

Eu como utilizador quero poder pesquisar por projetos na página principal.

***2º - Sprint***

- Design da página inicial

- Criação da estrutura participantes

- Criação da estrutura de inserir files

- Criar o sistema Tag

- Criar o sistema de pesquisa

*Requisitos 3ºSprint*

Adicionar mensagem flash de log in / Log out do utilizador.

Ver integração de facebook e twitter com rails

O projecto é o principal do nosso site, é aquilo para o qual ele foi feito. Deve mostrar toda a informação e as associações existentes.

O utilizador \*\*não registado\*\* deve ser capaz de listar todos os projectos existentes, vendo primeiro os projetos mais recentes.

O utilizador \*\*não registado\*\* deve ser capaz de, a qualquer altura, pesquisar por um projeto.

O utilizador \*\*não registado\*\* deve ser capaz de ver os detalhes de um projeto.

O utilizador \*\*não registado\*\* deve, nos detalhes do projeto, conseguir ver a que escola, curso e disciplina pertence o projeto, sendo capaz de navegar para detalhes de cada um desses campos.

O utilizador \*\*não registado\*\* deve, nos detalhes do projeto, ser capaz de ver as pessoas associadas ao projeto (equipa e orientadores).

O utilizador \*\*não registado\*\* deve, nos detalhes do projeto, ser capaz de ver informação da apresentação (se houve ou não), o dia, hora, sala e os júris.

O utilizador \*\*não registado\*\* deve, nos detalhes do projeto, ser capaz de ver as várias imagens associadas a este.

O utilizador \*\*não registado\*\* deve, nos detalhes do projeto, ser capaz de ver as tags associadas ao projeto.

O utilizador \*\*registado\*\* deve ser capaz de adicionar ou editar um projeto, sendo obrigado a escolher o curso / disciplina a que está associado

O utilizador \*\*registado\*\* deve ser capaz de apagar um projeto, apagando os objetos associados a este (p. ex. apresentações).

O utilizador \*\*registado\*\* deve ser capaz de adicionar uma ou várias imagens a um projeto.

O utilizador \*\*registado\*\* deve ser capaz de adicionar e remover pessoas (equipa, orientadores) ao projeto.

O utilizador \*\*não registado\*\* deve conseguir ver informação da disciplina tal como o ano e semestre desta, a que curso pertence os últimos projetos desta.

O utilizador \*\*registado\*\* deve ser obrigado a escolher o curso a que a disciplina pertence ao criar/editar.

O utilizador \*\*registado\*\* deve ser obrigado a confirmar que quer apagar uma disciplina. Ao apagar a disciplina, os seus projetos também terão que ser apagados.

O utilizador \*\*não registado\*\* deve ser capaz de listar as instituições existentes.

O utilizador \*\*não registado\*\* deve ser capaz de ver os detalhes de uma instituição.

O utilizador \*\*registado\*\* deve conseguir adicionar uma nova instituição.

O utilizador \*\*registado\*\* deve conseguir editar uma nova instituição.

O utilizador \*\*registado\*\* deve ser capaz de apagar uma instituição, apagando assim tudo o que lhe está associado.

O utilizador \*\*registado\*\* deve receber mensagens de erro claras ao tentar adicionar / alterar uma instituição com dados inválidos.

O utilizador \*\*não registado\*\* ao ver os detalhes de uma escola deve conseguir ver a que instituição pertence.

O utilizador \*\*registado\*\* ao adicionar/editar uma escola terá que escolher a instituição a que este pertence.

O utilizador \*\*não registado\*\*, ao ver os detalhes de uma escola, deve ver a lista de cursos (degrees) agrupados por tipo (degree type).

O utilizador \*\*não registado\*\*, ao ver os detalhes de uma escola, deve ver também uma lista dos projetos dessa escola, ordenados por data (os últimos primeiro)" \*\*deixar esta para depois, não prioritária, seria com :through\*\*.

***3º - Sprint***

- Melhorar o sistema Tags

- Criação da estrutura administração

- Melhorar a estrutura de inserir files (projetos)

- Recuperação da password por email

*Requisitos 4ºSprint*

Style Password recuperação via email

Auto complete tags e search

Funcionalidades users vs admins

Recolher informação real sobre cursos, escolas, disciplinas?

Testes UX com users tipo.

Eliminar possíveis efeitos cansativos, nº passos para fazer certas ações, teste diferentes Resoluções, otimização do fundo, tipos de letra, tamanhos, menus, vistas de cada uma das coisas.

Integração social e rss de projetos

Search a funcionar

Layout verificar altura máxima das fotos página principal

Projetos Relacionados, onde está visível por tag's colocar em vista como na 1ª página

Escolas, mostrar só cursos com projetos

Teste do app.

***4º - Sprint***

- Melhorar o design

- Melhorar a pagina de inserir projetos

- Melhorar o sistema de pesquisa

- Criar sistema feed noticias para pagina do facebook (IoT.IPT)

- Testar

- Mover a estrutura RoR para o servidor

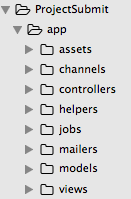
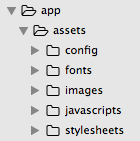
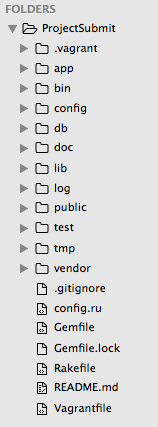
- Testar

---- ainda falta ver melhor as interações ----

### Estrutura do Projecto

Tendo uma estrutura simplificada, ainda assim existe aspetos importantes que qualquer programador deve conhecer como parte importante no desenvolvimento [3];

**Estrutura principal:**



- **.vagrant**:ficheiro onde guarda as configurações do utilizador do desenvolvimento.

- **app**: onde se encontra (**MVC**)

/**assets**

/javascripts

/stylesheets

/**controller**

/**models**

/**views**

/assets/Javascript e /Stylesheets, são pastas onde guardam as frameworks de design, que referimos anteriormente.

Fig.6 – Estrutura MVC RoR

/Controller onde se encontra as paginas controllers

/model onde se encontra as paginas models e ligações entre modelos

/views onde se encontra as paginas que são apresentadas ao utilizador

- **config**: routes, environments, initializers.

- **db**: onde se encontra tudo sobre a base de dados: migrate, seeds, schema.

- **doc**: Pasta onde se encontra a documentação de desenvolvimento.

- **public**: paginas gerais que mostrar erros de app e outras informações.

- **test**: onde se configura os teste da app

- **Gemfile**.rb: onde estão todas as gems a serem utilizadas.

- **Vagrantfile** : ficheiro de configuração da maquina virtual “vagrant” que contem o Linux.

Existe outras pastas e ficheiros na estrutura, más consideramos para segundo plano, já que não são essenciais para o desenvolvimento propriamente dito, estás funcionam com o conjunto da MVC. [5]

Definimos criar uma estrutura para responder aos requisitos apresentados;

**Projetos**: recipiente para guardar a informação em pdf, fotos e os colaboradores.

**Escolas**: Registar escolas na plataforma para associar aos projetos.

**Cursos e Disciplinas**: Registar cursos referentes às escolas.

**Participantes**: Registar alunos e professores referentes aos projetos.

**Tags**: Poder-se pesquisar por palavras pretendidas.

**Administração**: Poder-se administrar facilmente todos os registos e utilizadores.

**Divulgação**: Poder-se ter um sistema de “*feeds*” noticias, quando é registado um novo projeto.

**Base de dados**: Guardar toda a informação estruturada e com referencias.

**Github** – Repositório partilhado de desenvolvimento.

**Vagrant** – Utilização *VBox* com sistema linix, para utilização dos desenvolvedores ser uniforme.

**Trello** – Controlo das interações e tarefas a executar pela equipa.

### Bibliotecas Utilizadas (Gems)

A Gens no RoR, são bibliotecas de terceiros disponíveis para auxiliar ao desenvolvimento da aplicação. As gems podem ser vistas como bibliotecas reutilizáveis de código Ruby, embora algumas podem ter código nativo em C, Java ou .Net. O RubyGens é um gestor de pacotes de gems, que vai mantendo atualizado e disponibiliza as bibliotecas para o uso dos desenvolvedores, de um modo livre.

#### Paperclip

Esta biblioteca permite gerir de forma mais fácil o envio de ficheiros (*upload*) pelo utilizador. Foi utilizada no modelo *Person* para a foto das pessoas e no modelo Project para fazer toda a gestão de documentos, slides e imagens relativas a cada projecto.

Link: <https://rubygems.org/gems/paperclip>

#### Devise

Esta biblioteca permite criar uma estrutura de autenticação (login, logout, Resister) para utilizadores. Foi usada para fazer a autenticação de utilizadores que inserem projectos.

Link: <https://rubygems.org/gems/devise>

#### Rails\_Admin

O Rails\_Admin permite criar uma estrutura completa de gestão de todo o site feito (backoffice), em que poderemos defenir um administrador do site, com poderes de registar, modificar e eliminar utilizadores e todos os conteúdos.

Link: <https://rubygems.org/gems/rails_admin>

#### will\_paginate

Will\_paginate fornece uma API simples para realizar consultas paginadas com Active Record, DataMapper e Sequel e inclui ajudantes para renderização de links de paginação em aplicativos da web Rails.

Link: <https://rubygems.org/gems/will_paginate>

#### sqlite3

O sqlite3 é um modulo de interação de base de dados, utilizada no Ruby on Rails, de modo simplificado e leva. Foi usado para a criação da nossa estrutura de dados, e onde guarda toda a informação / conteúdos.

Link: <https://rubygems.org/gems/sqlite3>

#### bootstrap

Esta Framework é a mais responsive para o desenvolvimento de aplicações moveis (mobile), em termos de HTML e CSS. Foi usada no nosso desenvolvimento, para termos um layout agradável e compatível com dispositivos moveis.

Link: https://rubygems.org/gems/bootstrap

#### will\_paginate-bootstrap

Esta biblioteca cria de forma rápida uma estrutura de paginação na pagina com vários registos, mostrando um limite defenido de registo e no fim a navegação entre paginas (registos). Foi usada na página inicial mostrando os projectos com um limite defenido.

Link: <https://rubygems.org/gems/will_paginate-bootstrap>

#### filesize

Esta biblioteca é uma class para manipular tamanhos, permite definir os tamanhos que desejarmos. Poderemos definir um 1G para xxxKb. Foi utilizado .........

Link: <https://rubygems.org/gems/filesize>

#### font-awesome-sass

Biblioteca de estilo de letras que trabalha em conjunto com SASS. Foi utilizado no projeto juntamente com estilos do layout.

Link: <https://rubygems.org/gems/font-awesome-sass>

#### jquery-rails

Bibioteca de jquery é o nosso javascript compactado se podemos chamar assim. Fui utilizado juntamente com outras frameworks para o layout.

Link: <https://rubygems.org/gems/jquery-rails>

#### coffee-rails

Uma biblioteca para responder as solicitações de javascript, ..... .Foi usado .....

Link: <https://rubygems.org/gems/coffee-rails>

#### sass-rails

A biblioteca de SASS, é um adaptador do rails para pipeline. Foi usado em......

Link: https://rubygems.org/gems/sass-rails

#### kaminari

O kaminari é uma biblioteca de paginação Scope, limpa e poderosa, com poder de ser prosonalizada e sofistada, serve para criar paginas dinâmicas de vistas com os limites definidos. Foi usada ........

Link: <https://rubygems.org/gems/kaminari>

### Modelos

Os modelos correspondem a todos os dados relacionados que estão a ser transferidos entre os views e os controllers.

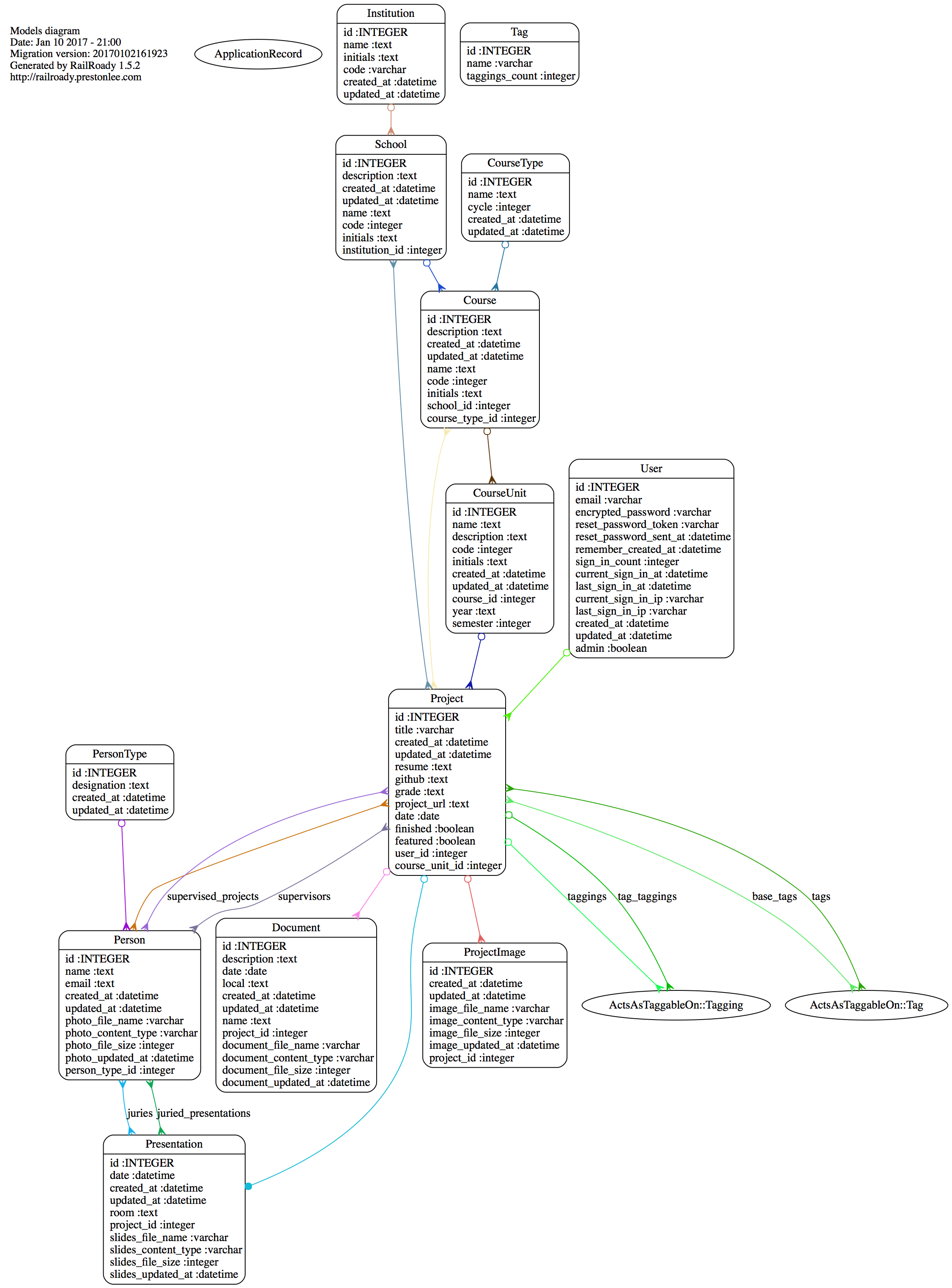


fig.7 – Diagrama dos Modelos

### Controladores

Os controladores funcionam como uma interface entre os model e views, para processar todas as solicitações feitas do utilizador, manipula os dados usando os models e interagem com os views.

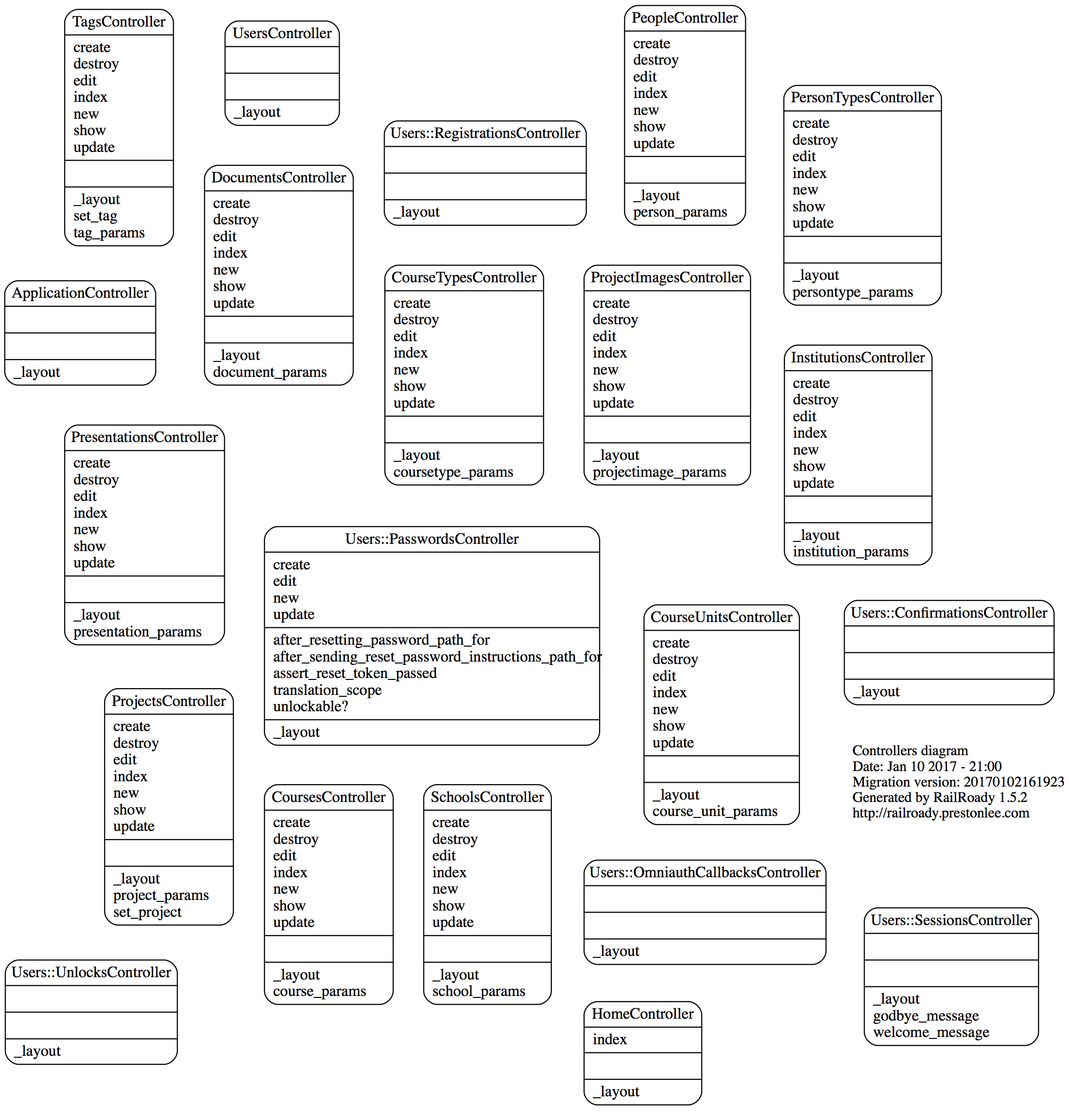


fig.8 – Diagrama dos controladores

### Vistas

As views são usadas para toda a lógica da estrutura do aplicativo, ou seja, o que o utilizador pode interagir.

*Página inicial:*

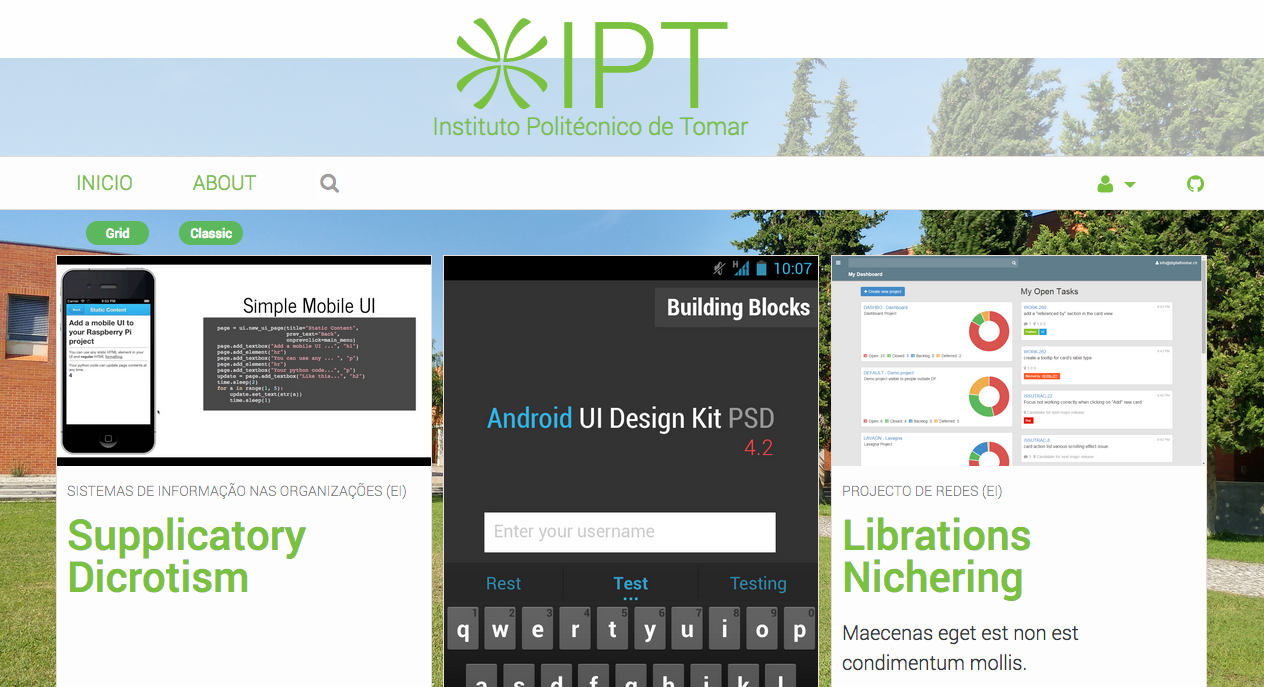


fig.9 – Design da pagina inicial

*Login:*

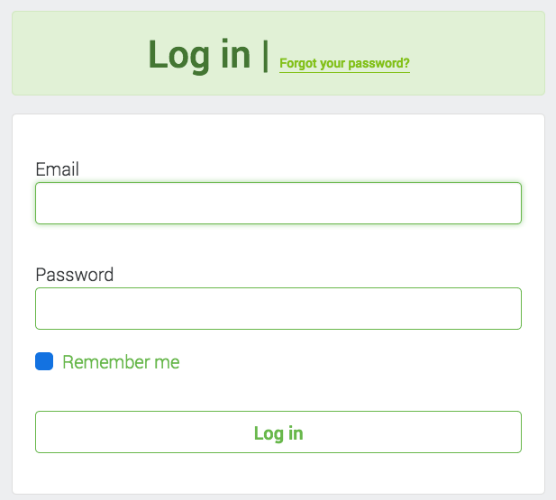


fig.10 – login de orientadores

*Novo Projeto:*

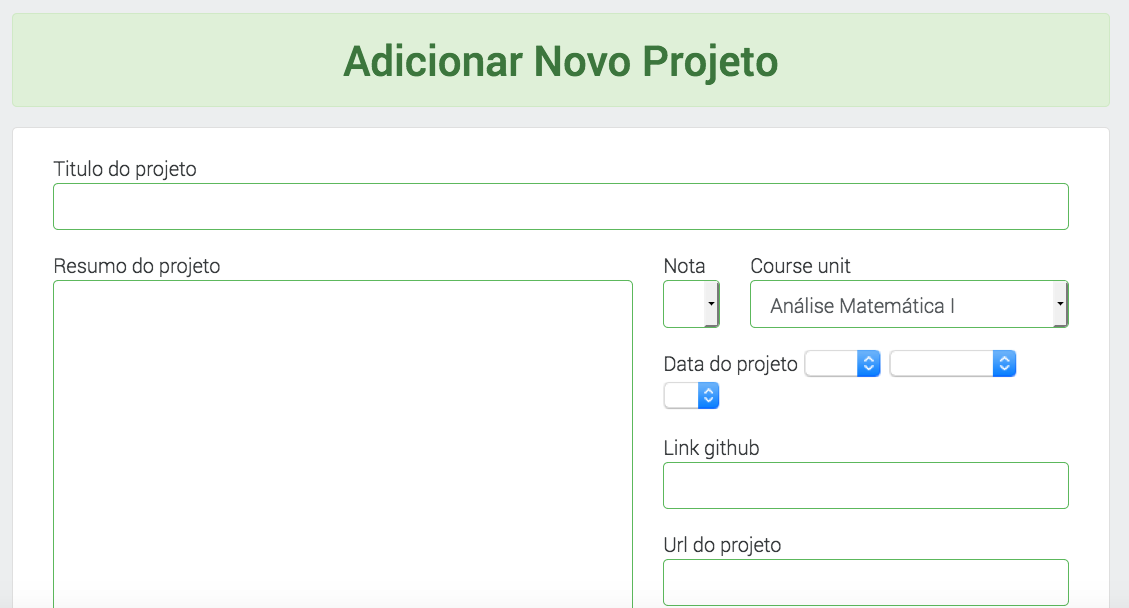


fig.11 – Adicionar novo projeto

*Administração:*

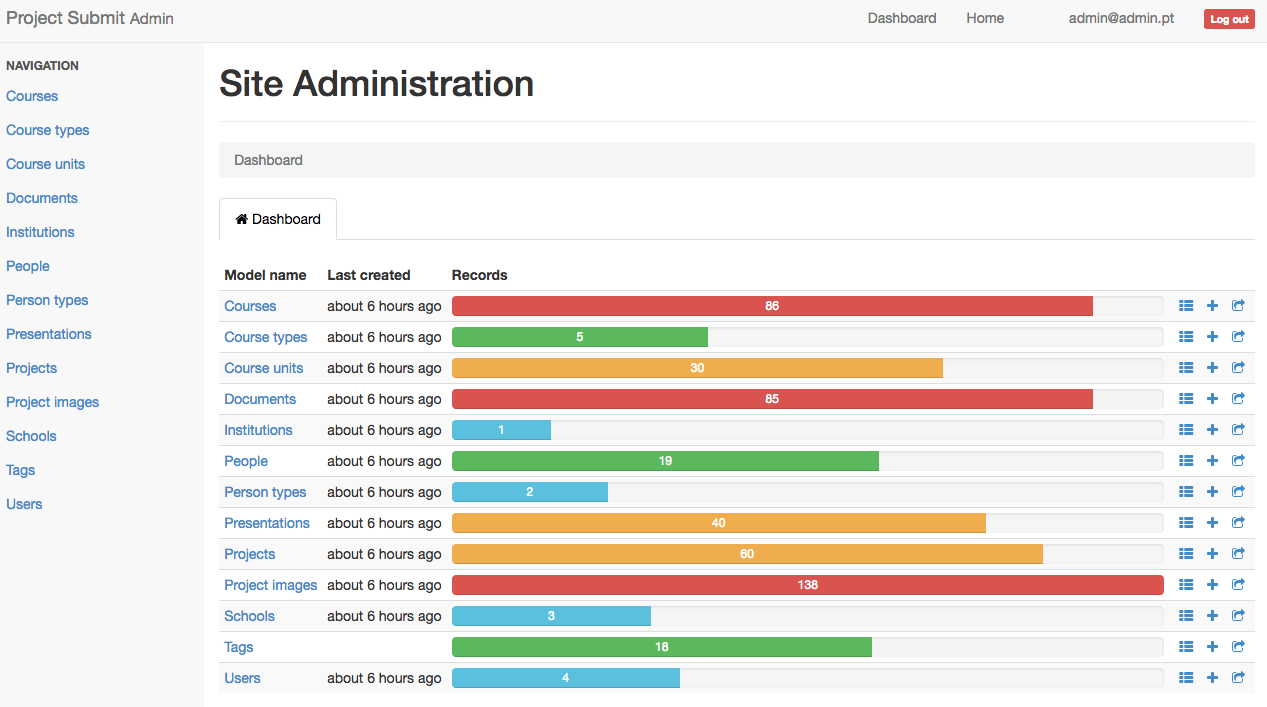


fig.12 - Administração

*Escolas:*

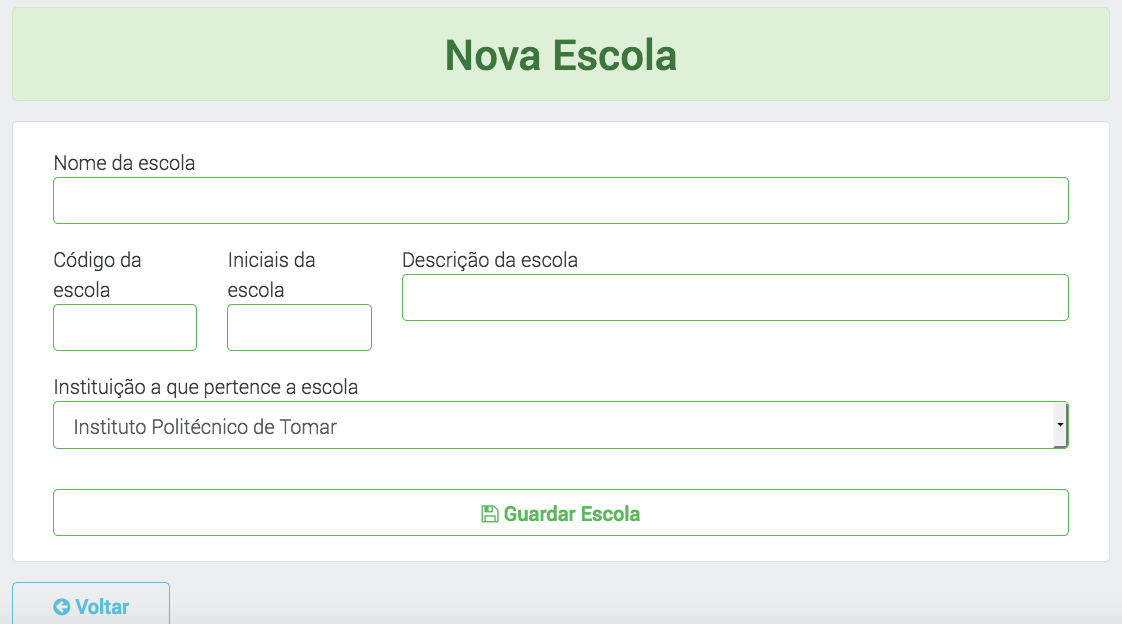


fig.13 – Nova escola

*Cursos / Disciplina:*

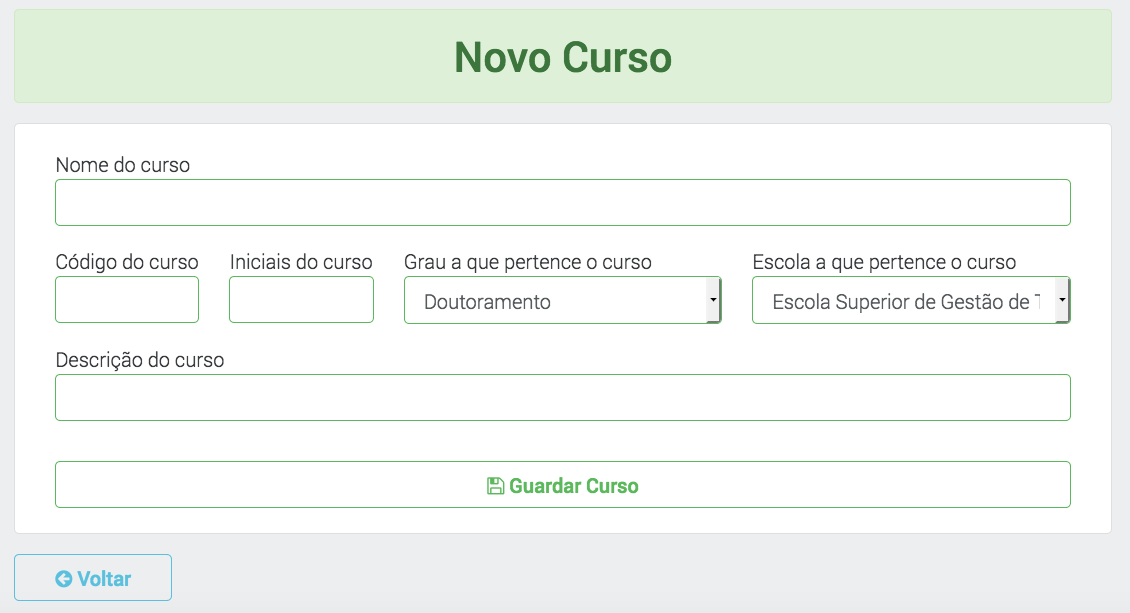


fig.14 – Novo Curso

### Como experimentar / colaborar?

Pode experimentar o projecto ou participar no desenvolvimento fazendo:

Instalar vagrant e virtualbox

Vagrant: <https://www.vagrantup.com/downloads.html>

VirtualBox: <https://www.virtualbox.org/wiki/Downloads>

Git clone : <https://github.com/nelsonmpg/IPTomar-MEI-IoT-ES.git>

* Modo de utilização:
* Abrir a consola
  + cd ~/IPTomar-MEI-IoT-ES/ProjectSubmit/
  + vagrant up
  + vagrant ssh
  + cd /vagrant
  + bundle install
  + sudo apt-get install imagemagick –y
  + rails db:setup
  + rails s –b0.0.0.0
* No browser escrever: localhost:3000

# Conclusão

O desenvolvimento ágil foi uma novidade entre o grupo, no entanto todos tiveram uma adaptação fácil. Compreendeu-se as importâncias das interações, que se realizavam semanalmente (aulas), onde ficava notoriamente vincado o desenvolvimento do trabalho por parte da equipa, ainda se nutava o espirito de grupo e a entreajuda entre membros na resolução de problemas. A equipa conclui que este método é bastante pertinente no desenvolvimento de app para a web, e que se nutou um desenvolvimento mais rápido e conciso.

No que respeita ao desenvolvimento do projeto, considera-se que é um avanço no mundo tecnológico, já que se passa a disponibilizar os projetos online dos estudantes. Assim poderão ser evoluídos e ou mesmo aproveitados para a investigação de novos conceitos, dando a estes um paradigma diferente do que se passa no momento atual, em que se encontrão escondidos / guardados algures, onde ninguém os pode consultar porque não sabe onde consultar.

Em sumo poderemos dirigir-mos ao conjunto do nosso projeto e poderemos dar nota positiva, tanto na aprendizagem como no desenvolvimento com o Ruby on Rails, dois sistemas novos, com uma grande potencia de mercado e ágil no seu desenvolvimento. Considerando-se um desenvolvimento rápido e eficaz, esta será uma tag a ter em conta nas nossas pesquisas para elaboração de novos trabalho.

## Trabalho futuro

A equipa considera que existe espaço para desenvolver mais a aplicação web, desde a um layout mais flexível às considerações de um manuseamento mais eficaz dos projectos.

As aplicações em RoR, são bastantes flexíveis e permitem uma boa integração de novos conteúdos, podendo assim a ser alargados a um conceito de guardar projectos, más também ao conceito de guardar informação necessária ao aulo, tal como horários, salas, exames, recursos, este são muito vagos no site do IPT, com uma difícil pesquisa e demorada. Assim em suma, poderemos dizer que seria um portal do aluno para o aluno, caso que não acontece até agora.

Aconselha-mos um desenvolvimento ágil na estrutura corrente, podendo assim melhorar e interagir todos os conteúdos nela existentes.

# Referências

[1] Ruby, Ruby. Disponível em: [http://www.ruby-lang.org/pt/](http://www.ruby-lang.org/pt/%20) . Acesso em 17 de janeiro de 2017.

[2] TIOBE Index for January 2017, TIOBE. Disponível em : <http://www.tiobe.com/tiobe-index/>. Acesso em 17 de Janeiro de 2017.

[3] Ruby S., Thomas D., Hansson D.,H.,(2016). *Agile Web Development with Rails 5*. ISBN-13:978-1-68050-171-1. Raleigh, North Caroline.

[4] Fowler C.,(2016). *Rails Recipes*. ISBN 0-9776166-0-6. Dallas, Texas.

[5] RubyGems. Disponivel em : <https://rubygems.org>. Acesso em 09 de janeiro de 2017.