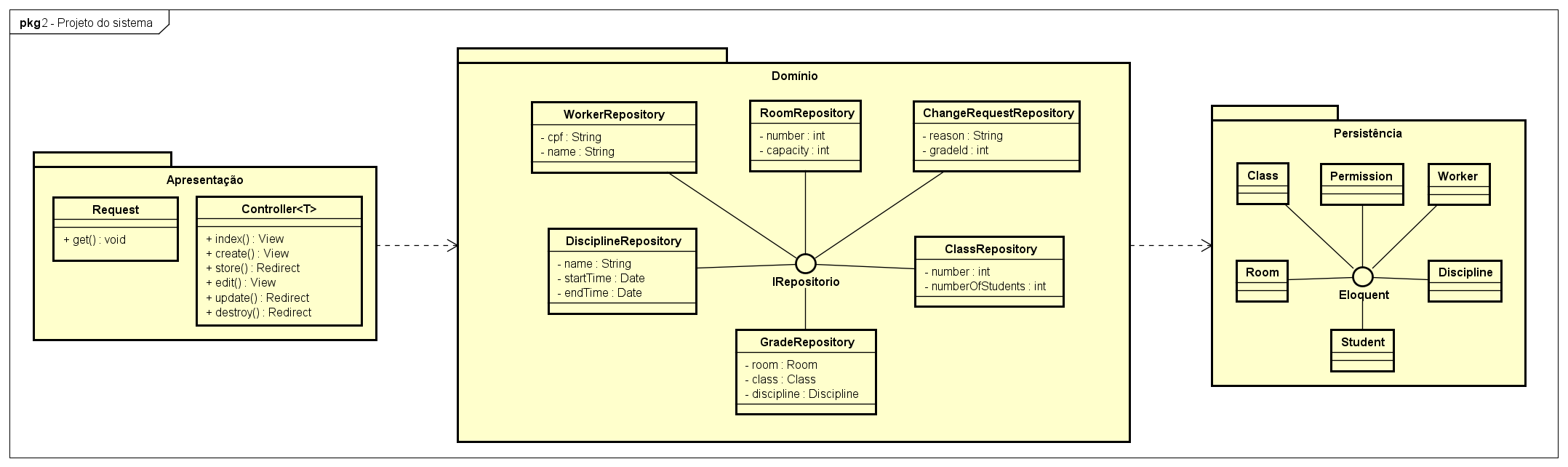
# Gerenciamento de espaço físico

# Projeto do sistema – *Design*

There is guidance within this template that appears in a style named InfoBlue. This style has a hidden font attribute that allows you to toggle whether it is visible or hidden in this template. Use the Microsoft® Word® menu **Tools > Options > View > Hidden Text** check box to toggle this setting. There is also an option for printing: **Tools > Options > Print**.

## Estrutura do projeto



## Subsistemas

## Persistência

Responsável por realizar a comunicação com o banco de dados, armazenar, remover, visualizar e editar dados. É através deste pacote que a aplicação gerencia todos os seus dados. Todas as classes representam entidades do banco de dados para fins de diferenciação entre as entidades do domínio. Visto que utilizamos o framework *Laravel*, todas estas classes implementam a interface *Eloquent*, que realiza o gerenciamento de CRUD da entidade com o banco.

## Domínio

O domínio armazena as entidades de negócio em forma de repositórios. Estes repositórios servem para diferenciar as entidades de negócio das entidades do banco. Também são responsáveis pela conversão dos objetos entre entidade de banco para negócio e vice-versa, além de realizar as requisições para a camada de persistência. Desta forma, ele devolve para a apresentação somente o que ela pode visualizar, sem realizar nenhuma alteração no banco.

## Apresentação

A camada de apresentação gerencia as telas de exibição para o usuário, contendo todas as páginas HTML, as bibliotecas *jQuery* e *bootstrap*, além dos logos, imagens e CSS’s da aplicação. Ela apenas consome dados do domínio, não tendo contato com a persistência. Quem faz o intermédio entre a apresentação e a persistência é o domínio. Também é responsável por gerenciar as rotas de URLs do sistema.

## Padrões

## Model-View-Controller

O padrão Model-View-Controller (MVC) é bastante utilizado em sistemas web, pois permite dividir o que é regra de negócio, o que é uma entidade e o que é apenas visualização. Como utilizaremos uma camada de apresentação e uma de domínio, encaixa-se perfeitamente, visto que ajudará a organizar a codificação.

## Permissões

O padrão de permissões serve para sistemas que possuem tipos diferentes de usuário, onde um possui mais ou menos privilégios e acessos que outro. Geralmente são sistemas onde há um administrador que tem acesso a tudo, e usuários “comuns” que podem somente acessar seus próprios dados. Nosso sistema é motivado a utilizar este padrão pois há uma diferença entre o tipo de funcionário: funcionário da secretaria (que pode fazer o *CRUD* dos dados, além de gerar grades) e o professor (que pode somente visualizar dados e requisitar alterações).

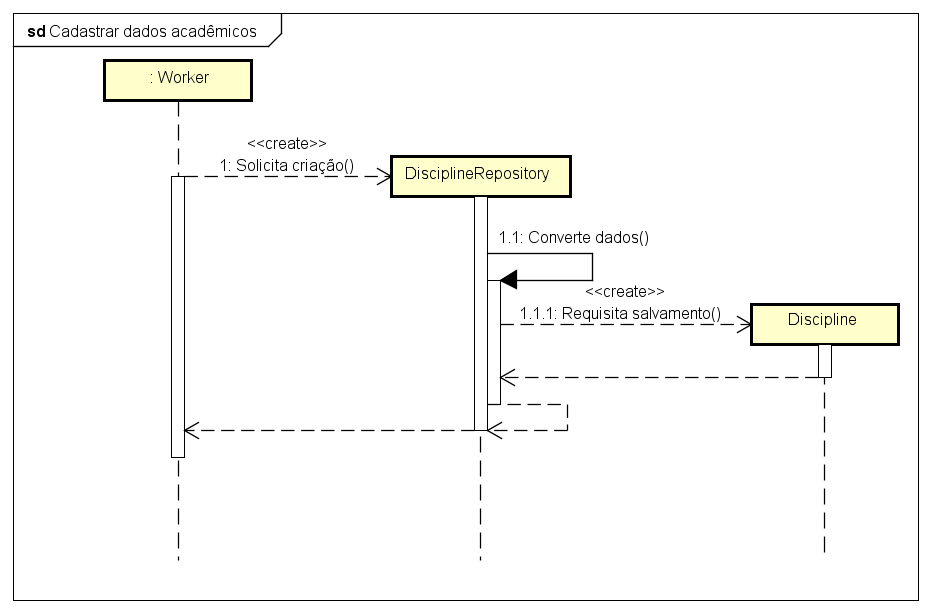
## Realização dos requerimentos

## Cadastrar dados acadêmicos

## Visão dos participantes

O funcionário acessa a página de “turmas” e deseja inserir ou editar dados de uma turma. Ele clica na opção desejada e um formulário é apresentado. Após inserir os dados, o funcionário solicita o salvamento dos mesmos. A *controller* de turma valida os dados e, caso estejam corretos, chama o repositório de turmas. Este converte os dados para entidade do banco e então realiza o cadastro através do *Laravel*. Caso os dados não estejam corretos, a *controller* simplesmente retorna uma mensagem de erro.

## Cenário

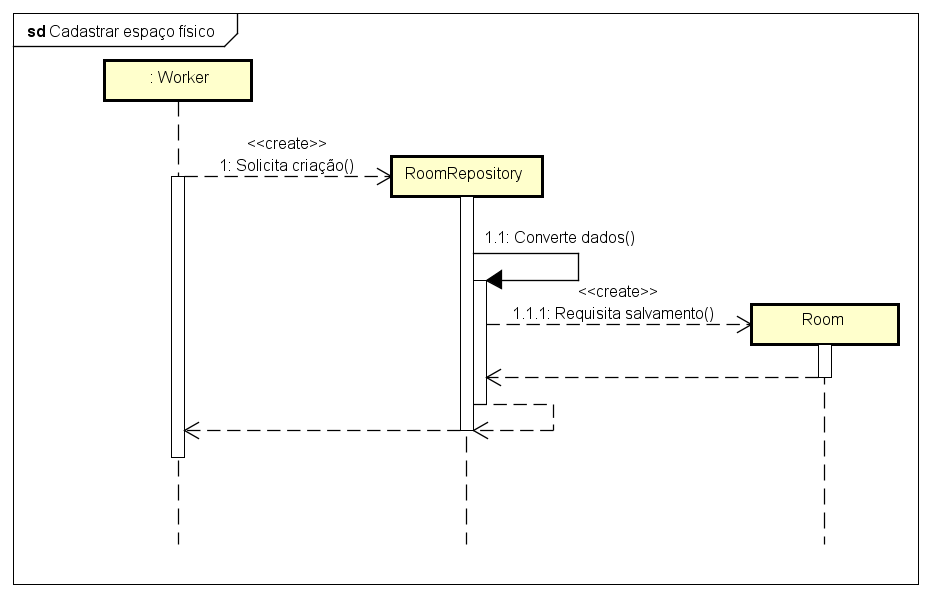


## Cadastrar espaço físico

## Visão dos participantes

O funcionário acessa a página de “salas” e deseja inserir ou editar dados de uma sala. Ele clica na opção desejada e um formulário é apresentado. Após inserir os dados, o funcionário solicita o salvamento dos mesmos. A *controller* de salas valida os dados e, caso estejam corretos, chama o repositório de turmas. Este converte os dados para entidade do banco e então realiza o cadastro através do *Laravel*. Caso os dados não estejam corretos, a *controller* simplesmente retorna uma mensagem de erro.

## Cenário



## Consultar alocação de uma sala

## Visão dos participantes

O funcionário acessa a página de “salas” e visualizar a lista de salas cadastradas. A *controller* de sala solicita para o banco a listagem de salas e os devolve para a *view*.

## Cenário

