Construção de Sistemas de Software

2023/2024

Relatório Fase 2

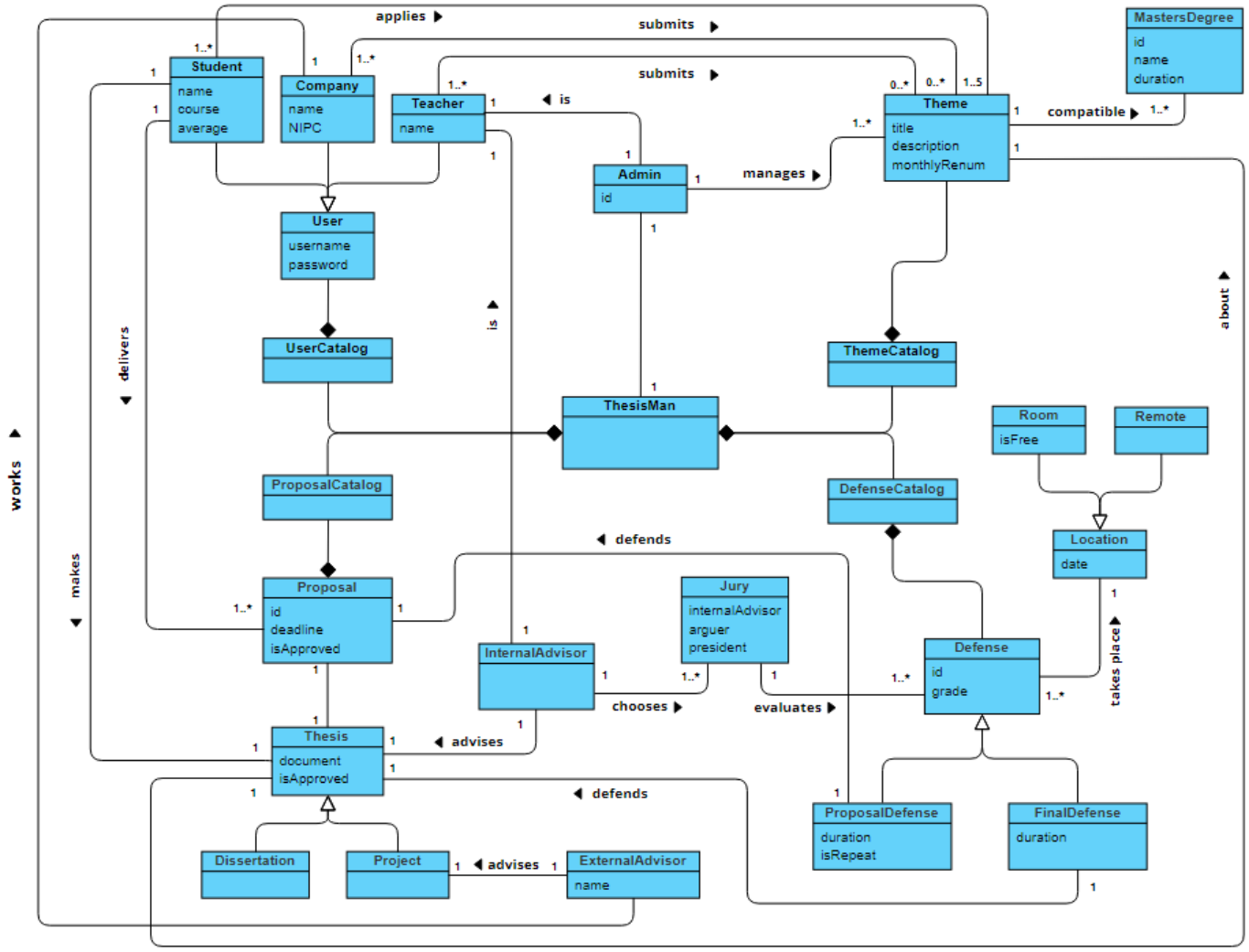
(Contém também o relatório da Fase 1)

Realizado por:

- Eduardo Proença 57551
- Tiago Oliveira 54979
- Manuel Barral 52026

Fase 1

Modelo de Domínio



Caso de uso K – SSD

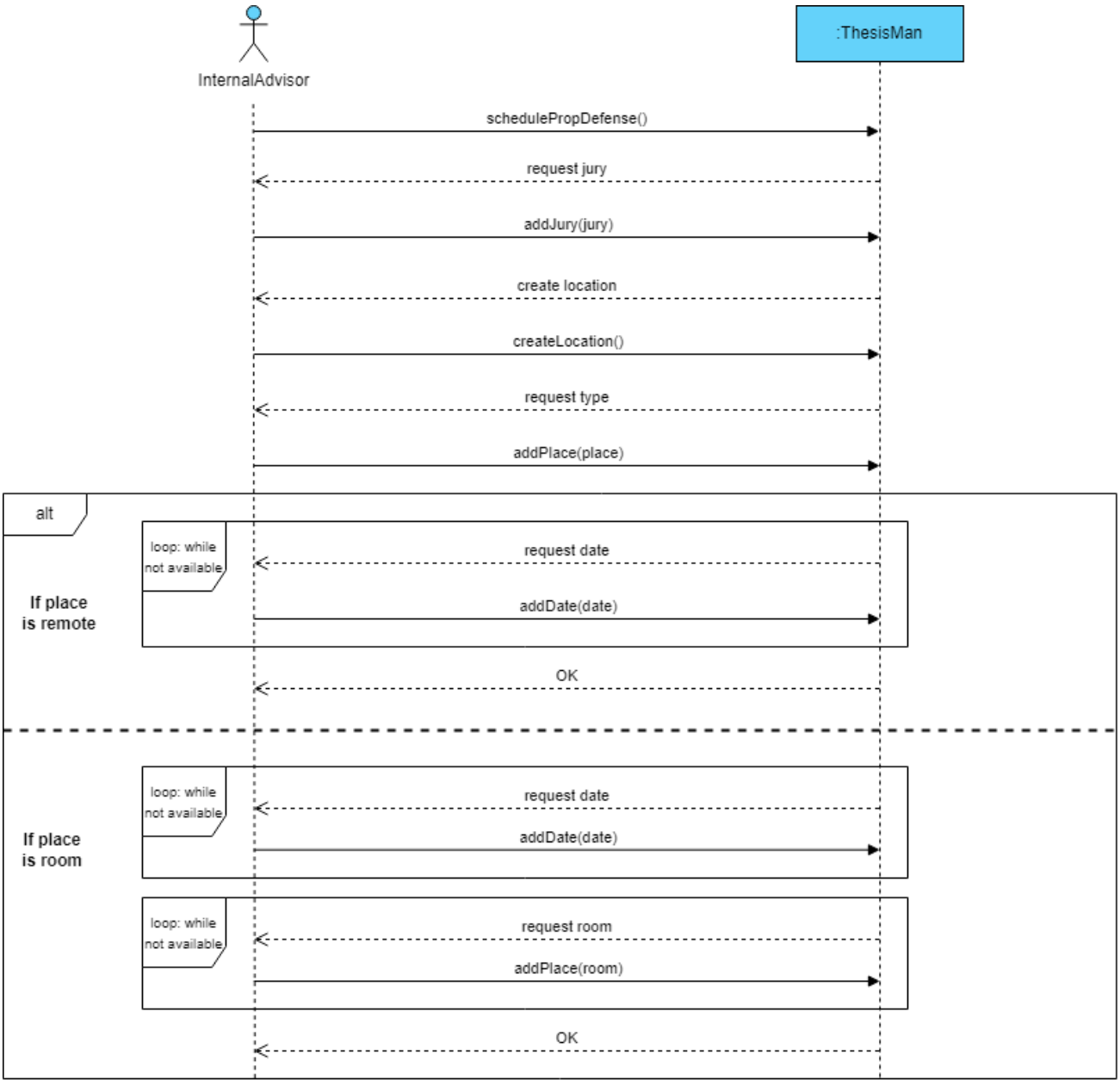
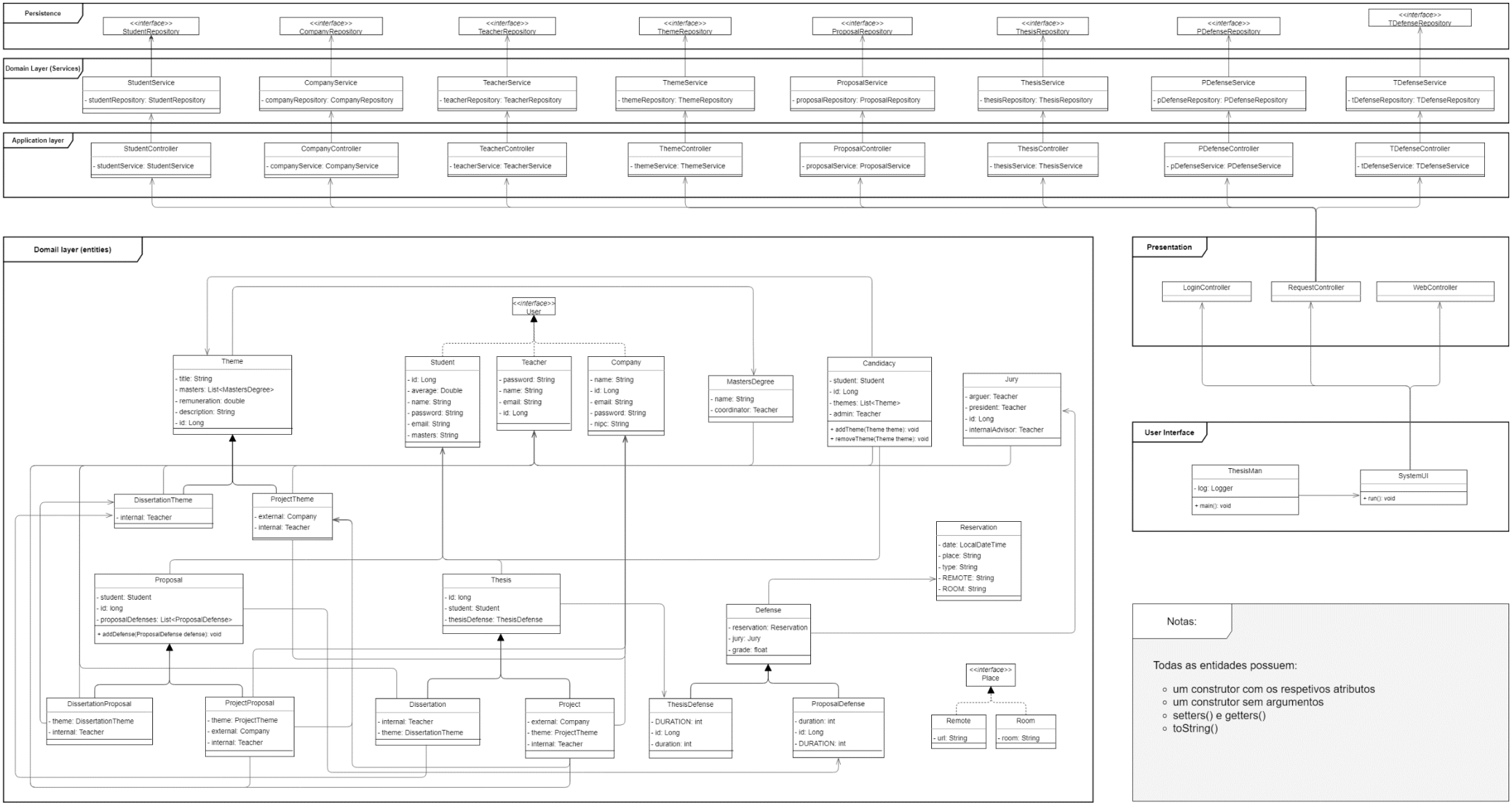


Diagrama de Classes



Notas:

Todas as entidades possuem:

- um construtor com os respetivos atributos
- um construtor sem argumentos
- setters() e getters()
- toString()

Padrões de desenvolvimento

Ao longo desta fase, foram utilizados padrões de abstração, assim como generalização, nos packages theme, thesis, proposal e defense. Foi também utilizado composição nos packages user e location.

Mapeamento

O mapeamento nesta fase foi pensado utilizando o padrão *Domain Model.* Desta maneira, foram identificadas 16 entidades:

- *Student*
- *Teacher*
- *Company*
- *Theme (DissertationTheme e ProjectTheme)*
- *Application*
- *Proposal (DissertationProposal e ProjectProposal)*
- *Thesis (Dissertation e Project)*
- *ProposalDefense*
- *ThesisDefence*
- *Jury*

Para as entidades *Theme*, *Proposal* e *Thesis*, foram utilizadas anotações como: @Inheritance(strategy = InheritanceType.SINGLE_TABLE), @DiscriminatorColumn e @DiscriminatorValue, para juntar as respetivas subclasses na mesma tabela.

Para as entidades *ProposalDefense* e *ThesisDefence*, foi utilizado na superclasse *Defence*, a anotação @MappedSuperClass, para guardar a informação da mesma nas respetivas tabelas.

Utilizando o padrão DataMapper, foram criados também repositórios para gerir o acesso à base de dados.

Fase 2

- Todos os casos de uso foram implementados.

- Para a gestão da sessão e autenticação dos casos de uso Web, é usado um Bean com scope *Session* que guarda se o utilizador está autenticado e os seus dados.

- De modo a facilitar o movimento dos dados através da API e a sua serialização foram criados *DTOs* (*Data Transfer Objects*) relacionados com entidades e casos de uso.

- Tiramos partido da arquitetura em camadas, utilizando *Controllers*, Handlers, *Services* e *Repositories*. A separação por camadas facilita qualquer modificação ou migração que possa vir a ser feita.

- Optamos por criar uma entidade *Stats* para guardar as estatísticas. A estatísticas irão ser atualizadas sempre que uma defesa for avaliada. Desta maneira, sempre que um utilizador quiser consultar as estatísticas, estas não precisam de ser calculadas. Esta abordagem é útil para aplicações com vários utilizadores, pois se calculássemos as estatísticas a cada consulta, iria consumir muitos recursos computacionais.

- Na interface Web, quando ocorre um erro, seja de validação de dados providenciados pelo utilizador ou outro tipo de erros, será mostrada uma mensagem de erro ao utilizador de forma a avisar o que correu mal.

- Os documentos de teses e propostas são guardados no sistema de ficheiros do servidor, através do *StorageService* num diretório chamado *files.*

- Foram feitas algumas modificações relativamente à fase 1, de forma a corrigir os problemas detetados e a fazer melhorias, nomeadamente a relações entre entidades.

- Qualquer password é aceite para os professore e alunos, visto que objetivo é fazer *mock* do sistema de autenticação da FCUL. As empresas podem ser registadas e a sua password é guardada seguramente em *hash* dentro da base de dados.

Utilizadores criados para testar a aplicação podem ser consultados no ficheiro README.