ThesisMan

João André 56380
João Costa 58226
Rafael Ferreira 57544

Introdução

Neste relatório apresentamos o nosso design para a arquitectura do **ThesisMan**, sistema de gestão de Teses de Mestrado. Nesta fase, abordamos a camada de negócio e a camada de dados. Utilizamos o padrão **Domain Model** para a camada de negócios e o padrão **Data Mapper** para a camada de dados, utilizando JPA com Spring Data para persistir os objetos numa base de dados relacional.

Casos de uso

Pretendemos suportar os **casos de uso** referidos no enunciado, aqui repetidos por conveniência:

A. Login com autenticação da universidade (Nota: vamos fazer mock desta funcionalidade e qualquer palavra passe será aceite)
B. Registo de utilizadores empresariais
C. Login de utilizadores empresariais
D. Submissão de temas por parte dos docentes
E. Submissão de temas por parte dos utilizadores empresariais
F. Listar os temas disponíveis neste ano lectivo, por parte dos alunos
G. Candidatura a um tema (limite de 5), por parte dos alunos
H. Cancelamento da candidatura a um tema
I. Atribuição dos temas aos alunos (da parte do administrador)
J. Submissão do documento de proposta de tese, por parte dos alunos
K. Marcação da Defesa da proposta de tese, por parte do orientador da tese
L. Registo da nota da defesa da proposta de tese, por parte do orientador da tese
M. Submissão do documento final de tese, por parte dos alunos
N. Marcação da Defesa de tese, por parte do orientador da tese, incluindo a nomeação do júri.
O. Registo da nota da defesa da proposta de tese, por parte do presidente do júri.
P. Recolha de estatísticas sobre taxa de sucesso dos alunos.

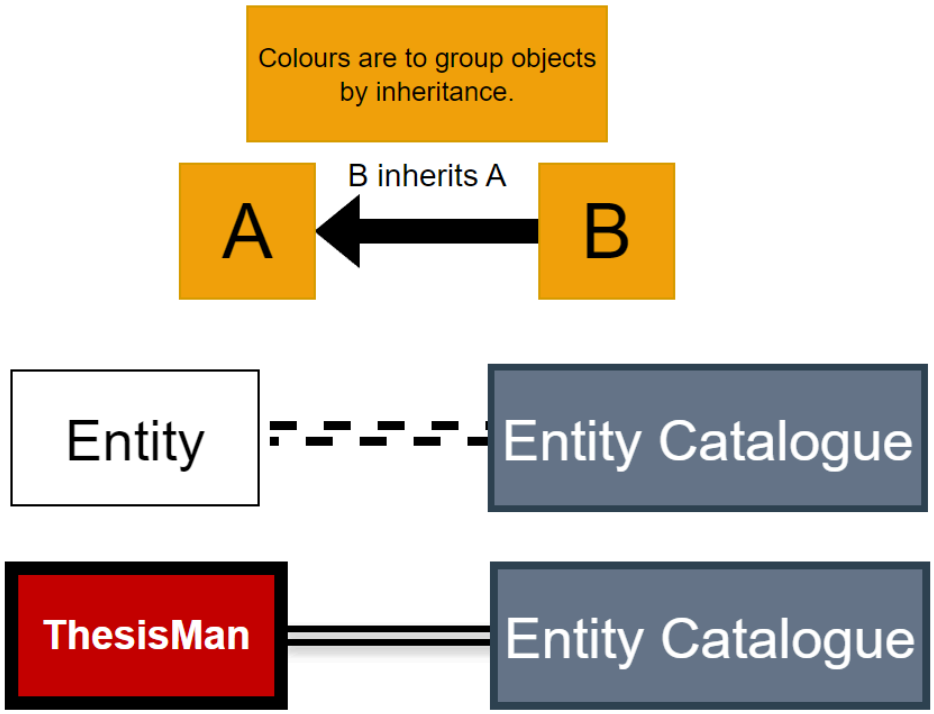
Requisitos não Funcionais

Pretendemos cumprir os requisitos não funcionais referidos no enunciado, aqui repetidos por conveniência:

• A autenticação deverá ser feita com o sistema de autenticação da FCUL.
• Toda a informação deverá ser armazenada numa base de dados relacional.
• A plataforma deverá ser implementada em Spring Boot, de forma a ter um custo de desenvolvimento baixo, e a usar a linguagem Java, que é a linguagem para a qual é mais fácil de contratar engenheiros.
• A camada de dados deverá usar JPA e Spring Data.
• A aplicação deverá lidar com pedidos concorrentes, sem criar inconsistências.
• A camada de negócios deverá usar o Domain Model, com meta-dados suficientes para que o JPA faça o mapeamento para a base de dados.
• A interface deverá ser através da web, num momento inicial.
• Deverá ser exposta uma API REST que os clientes (web ou mobile) poderão usar para interagir com a aplicação.
• O repositório deverá aceitar apenas código com o nível de qualidade aceite pela equipa (usando o pre-commit).
• O projecto deverá correr dentro de um docker, que será a forma como vai ser deployed no servidor de produção.
• O controlo da participação de cada membro da equipa será feita através da actividade no repositório git.
• As decisões técnicas deverão estar justificadas num relatório técnico.

Modelo de domínio

Legenda



Explicação alto nível de decisões do desenho:

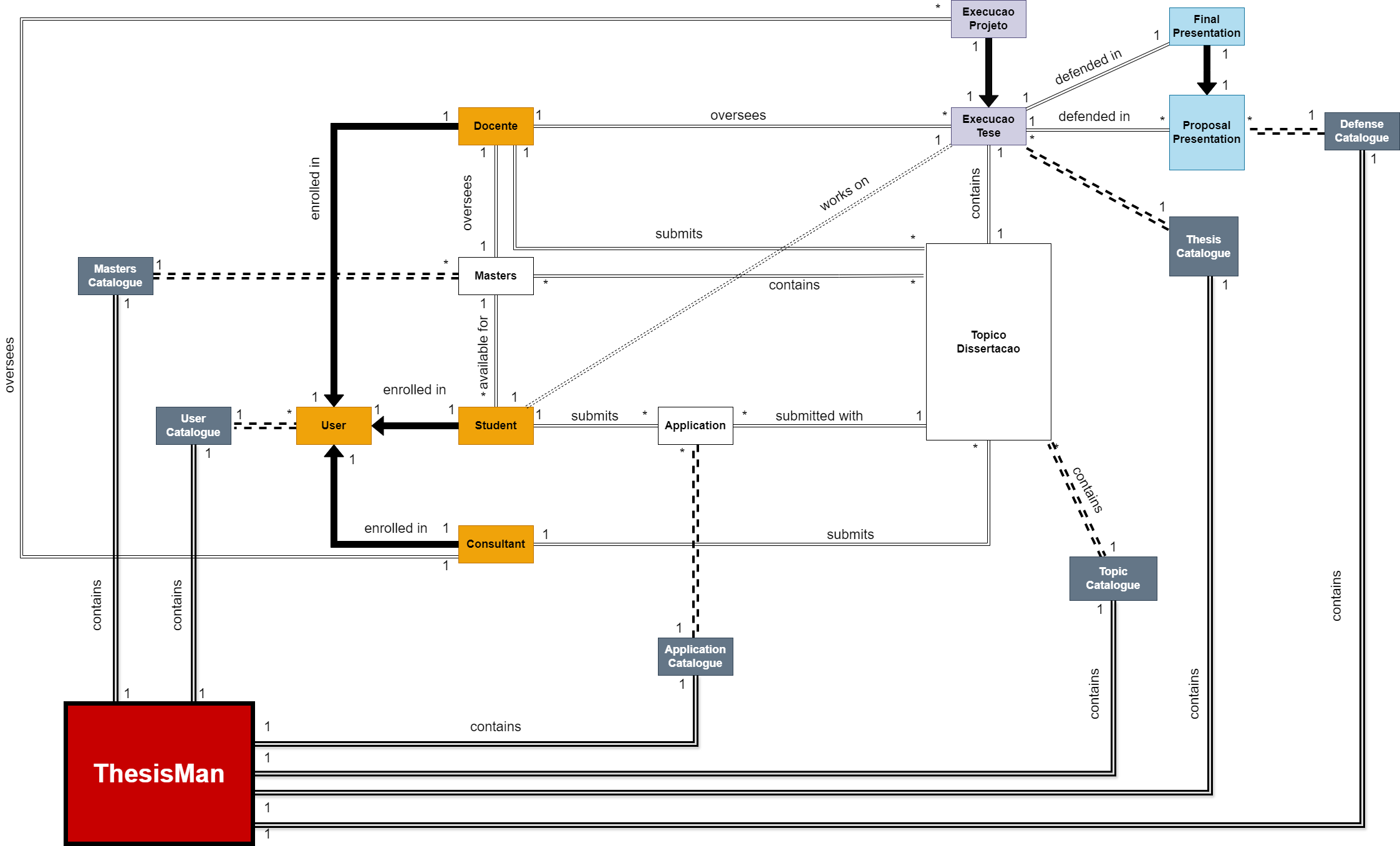
User: Daqui saem as Entidades **Student**, **Consultant** e **Professor**. Estas relacionam-se com o User, contudo não se relacionam entre si, havendo então uma especialização para cada uma das Entidades.
DissertationTopic: Esta Entidade reúne informações como título, orientador interno e externo, Mestrados compatíveis, entre outros. Neste cluster também incluímos as Entidades **Masters** e **Application**. Onde **Masters** é apenas o nome do mestrado e o professor que o coordena. A **Application** é uma candidatura a um tópico de dissertação.
ThesisDefense: Tem a responsabilidade de marcar uma defesa sendo esta presencial ou remota. Apesar de na realidade serem diferentes, decidimos que podia ser representada por um atributo e 2 métodos diferentes. Esta entidade dá origem à sub-entidade **FinalThesisDefense** que apenas se refere à defesa final, que é única e tem um presidente que irá coordenar os trabalhos.
ThesisExecution: Esta é a Entidade que concretiza todas as outras. Guardando a nota da defesa final, o tópico da dissertação e os restantes dados relevantes da tese do aluno. Decidimos que como a Entidade **ProjectExecution** apenas difere da **ThesisExecution** no campo do orientador externo, então esta última seria a Entidade mãe, por ser mais dinâmica e flexível.

Observações:

Opções de implementação:

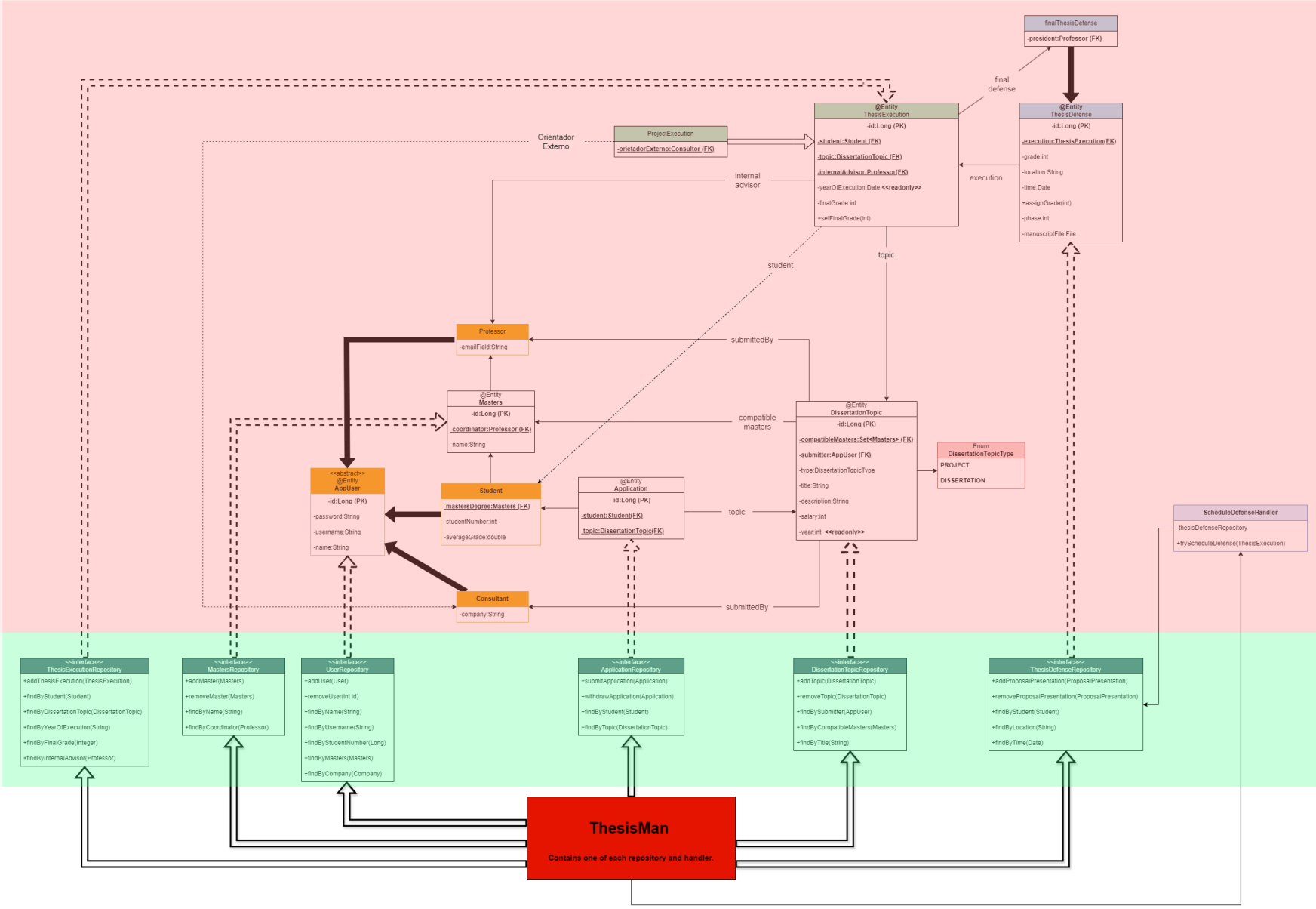
Não representamos os **Administradores** como uma entidade. Para os casos de uso que envolvem os administradores, escolhemos verificar se o utilizador atual é administrador consultando a tabela dos mestrados e os seus coordenadores.

Escolhemos ter apenas uma entidade para os tópicos de projeto e dissertação, e separar apenas na fase de execução. Considerámos a alternativa de ter apenas uma entidade para ambos os tipos de execução, utilizando um ENUM para os distinguir. A representação na base de dados seria semelhante. Mas a separação pareceu-nos mais natural.


Execucao Projeto
Final Presentation
defended in
Docente
oversees
Execucao Tese
defended in
Proposal Presentation
Defense Catalogue
enrolled in
oversees
works on
contains
submits
Thesis Catalogue
Masters Catalogue
Masters
contains
available for
Topico Dissertacao
oversees
enrolled in
User Catalogue
User
Student
submits
Application
submitted with
contains
enrolled in
Consultant
submits
Topic Catalogue
contains
contains
contains
Application Catalogue
contains
contains
contains
ThesisMan

Separação em camadas

Domain Layer



Data Layer

JPA
Spring Data

Postgres

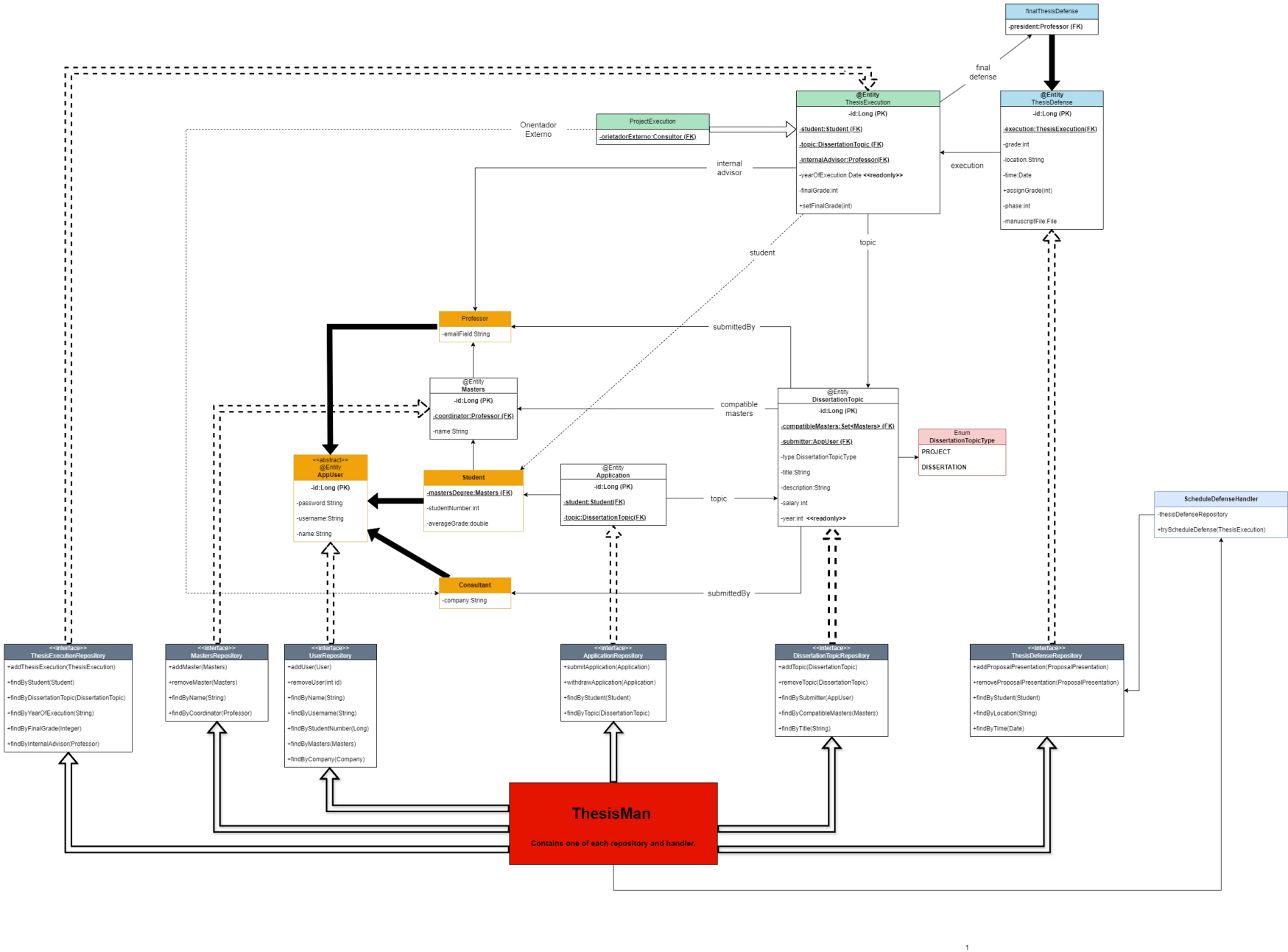
Modelo de classes

Domain Layer

Data Layer

JPA
Spring Data

Postgres



Mapeamento de hierarquias:
Na relação entre User, Professor, Student e Consultant decidimos achatar numa só tabela. Neste tipo de mapeamento o acesso a atributos é mais rápido, no entanto há um maior desperdício de espaço (embora acreditemos que não tenhamos um número de utilizadores de uma magnitude que exponha esta desvantagem).

Na associação entre um tópico de tese (DissertationTopic) e um mestrado compatível foi escolhido o tipo Set em vez de List com o objetivo de evitar objetos repetidos.

Relações ManyToMany:
Na associação entre topico de dissertação/projeto e mestrados compatíveis escolhemos criar uma tabela de relação. Pelo que percebemos, este é o único metodo suportado nativamente pelo JPA, e nem todas as bases de dados suportam outros métodos.

Non-Null fields:
Anotamos com @NonNull ou @Nullable e Column(nullable=true/false), para que haja verificação ao nível do Java e ao nível da coluna, na base de dados.

Herança vs Composição

Escolhemos, no caso dos Utilizadores, Execuções de Tese e Defesas de Tese utilizar Herança, para tirar proveito do polimorfismo. Reduzindo drasticamente a duplicação de código e distribuindo responsabilidades por cada classe específica sem que elas se conheçam entre si.

No caso dos estudantes, docentes e consultores, sendo estes extensões de um User, podem partilhar atributos comuns (username, password, nome) e podem ser tratados como igual em termos de registo, login e gestão da conta.

No caso das execuções da tese, escolhemos ter uma Execução de Projeto que herda a Execução da Dissertação. A diferença entre as duas é a presença de um orientador externo. Isto permite-nos implementar a lógica da marcação das defesas apenas uma vez.

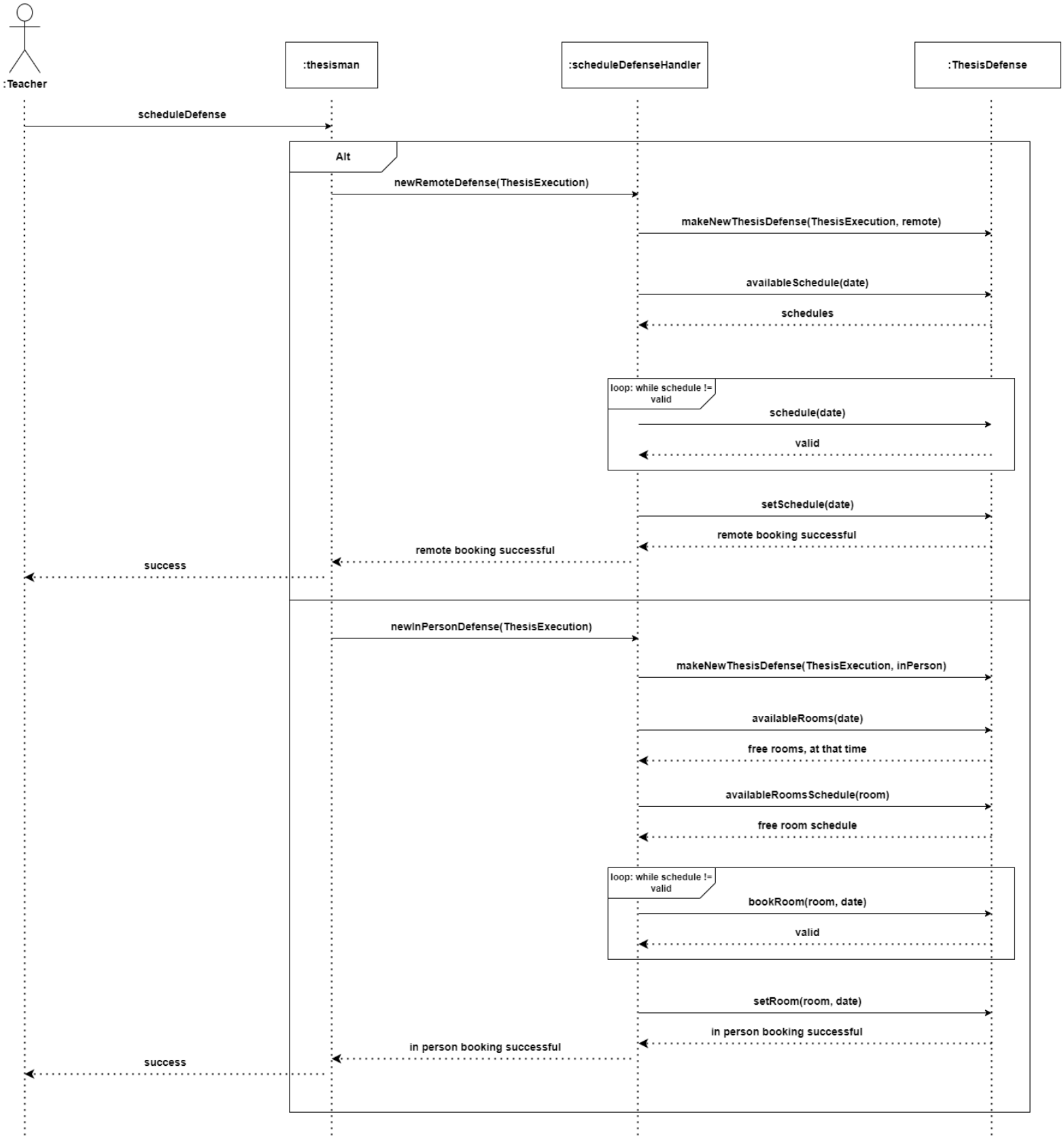
No caso das defesas, escolhemos ter uma Defesa Final que herda a Defesa de Tese. Quando implementamos a interface gráfica, isto permite-nos tratar da atribuição de nota da mesma maneira.

SSD:

Na realização do SSD partimos do princípio que:

1. O user já está autenticado e tem permissões para realizar a operação de marcação.
2. Existe uma tese válida para submeter dentro da defesa.
3. Assume-se que o Handler tem na sua posse o objeto ThesisDefense correto.

O diagrama pode ser visto na página seguinte:


:Teacher
:thesisman
:scheduleDefenseHandler
:ThesisDefense
scheduleDefense
Alt
newRemoteDefense(ThesisExecution)
makeNewThesisDefense(ThesisExecution, remote)
availableSchedule(date)
schedules
loop: while schedule != valid
schedule(date)
valid
setSchedule(date)
remote booking successful
remote booking successful
success
newInPersonDefense(ThesisExecution)
makeNewThesisDefense(ThesisExecution, inPerson)
availableRooms(date)
free rooms, at that time
availableRoomsSchedule(room)
free room schedule
loop: while schedule != valid
bookRoom(room, date)
valid
setRoom(room, date)
in person booking successful
in person booking successful
success

Base de Dados

As anotações JPA irão gerar a seguinte base de dados.

```
                    List of relations
 Schema |                  Name                  | Type  |  Owner
--------+----------------------------------------+-------+----------
 public | app_user                               | table | postgres
 public | application                            | table | postgres
 public | dissertation_topic                     | table | postgres
 public | dissertation_topic_compatible_masters  | table | postgres
 public | masters                                | table | postgres
 public | thesis_defense                         | table | postgres
 public | thesis_execution                       | table | postgres
(7 rows)
```

```
testdb=# \d app_user
                      Table "public.app_user"
     Column     |          Type          | Collation | Nullable | Default
----------------+------------------------+-----------+----------+---------
 average_grade  | double precision       |           |          |
 student_number | integer                |           |          |
 fk_masters_id  | bigint                 |           |          |
 id             | bigint                 |           | not null |
 dtype          | character varying(31)  |           | not null |
 company        | character varying(255) |           |          |
 name           | character varying(255) |           | not null |
 password       | character varying(255) |           | not null |
 username       | character varying(255) |           | not null |
Indexes:
    "app_user_pkey" PRIMARY KEY, btree (id)
    "app_user_username_key" UNIQUE CONSTRAINT, btree (username)
Foreign-key constraints:
    "fk54cihvrbjntyhxn6ge0e3vn2n" FOREIGN KEY (fk_masters_id) REFERENCES masters(id)
Referenced by:
    TABLE "thesis_execution" CONSTRAINT "fk5wnb3ou0b9qpxicijh2ffa56q" FOREIGN KEY (fk_consultant_id) REFERENCES app_user(id)
    TABLE "thesis_execution" CONSTRAINT "fkbdi34n49uqpaxlwnipm6b74xc" FOREIGN KEY (fk_student_id) REFERENCES app_user(id)
    TABLE "dissertation_topic" CONSTRAINT "fkhotu6y3nstrrsjdlqav1q8e0h" FOREIGN KEY (fk_submitter_id) REFERENCES app_user(id)
    TABLE "masters" CONSTRAINT "fkip14gq0nq79dh0tsta2t3suuj" FOREIGN KEY (fk_professor_id) REFERENCES app_user(id)
    TABLE "thesis_execution" CONSTRAINT "fkjaaw0gl61dkkf18v0x5ocj3uk" FOREIGN KEY (fk_internal_advisor_id) REFERENCES app_user(id)
    TABLE "application" CONSTRAINT "fkkl48bl5wpumu4bmcornwdtb9w" FOREIGN KEY (fk_student_id) REFERENCES app_user(id)
    TABLE "thesis_defense" CONSTRAINT "fkp6kvb8x9c6olb24epbld0qaoi" FOREIGN KEY (fk_president_id) REFERENCES app_user(id)
```

```
testdb=# \d application
                    Table "public.application"
          Column           |  Type  | Collation | Nullable | Default
---------------------------+--------+-----------+----------+---------
 fk_dissertation_topic_id | bigint |           | not null |
 fk_student_id            | bigint |           | not null |
 id                       | bigint |           | not null |
Indexes:
    "application_pkey" PRIMARY KEY, btree (id)
Foreign-key constraints:
    "fk9tecoktpnjjnmd4465ti9j1rx" FOREIGN KEY (fk_dissertation_topic_id) REFERENCES dissertation_topic(id)
    "fkkl48bl5wpumu4bmcornwdtb9w" FOREIGN KEY (fk_student_id) REFERENCES app_user(id)
```

```
testdb=# \d dissertation_topic
                    Table "public.dissertation_topic"
     Column      |          Type          | Collation | Nullable | Default
-----------------+------------------------+-----------+----------+---------
 salary          | double precision       |           | not null |
 type            | smallint               |           | not null |
 fk_submitter_id | bigint                 |           | not null |
 id              | bigint                 |           | not null |
 description     | character varying(255) |           | not null |
 title           | character varying(255) |           | not null |
Indexes:
    "dissertation_topic_pkey" PRIMARY KEY, btree (id)
Check constraints:
    "dissertation_topic_type_check" CHECK (type >= 0 AND type <= 1)
Foreign-key constraints:
    "fkhotu6y3nstrrsjdlqav1q8e0h" FOREIGN KEY (fk_submitter_id) REFERENCES app_user(id)
Referenced by:
    TABLE "thesis_execution" CONSTRAINT "fk30k9fa8uppo4sv3if19yxvdad" FOREIGN KEY (fk_dissertation_topic_id) REFERENCES dissertation_topic(id)
    TABLE "application" CONSTRAINT "fk9tecoktpnjjnmd4465ti9j1rx" FOREIGN KEY (fk_dissertation_topic_id) REFERENCES dissertation_topic(id)
    TABLE "dissertation_topic_compatible_masters" CONSTRAINT "fkhgl2kahkbywb0eret668k8ert" FOREIGN KEY (dissertation_topic_id) REFERENCES dissertation_topic(id)
```

```
testdb=# \d dissertation_topic_compatible_masters
      Table "public.dissertation_topic_compatible_masters"
        Column         |  Type  | Collation | Nullable | Default
-----------------------+--------+-----------+----------+---------
 dissertation_topic_id | bigint |           | not null |
 master_id             | bigint |           | not null |
Indexes:
    "dissertation_topic_compatible_masters_pkey" PRIMARY KEY, btree (dissertation_topic_id, master_id)
Foreign-key constraints:
    "fkhgl2kahkbywb0eret668k8ert" FOREIGN KEY (dissertation_topic_id) REFERENCES dissertation_topic(id)
    "fkm95qmk1yu6gwtym3h3xppe5vh" FOREIGN KEY (master_id) REFERENCES masters(id)
```

```
testdb=# \d masters
                        Table "public.masters"
     Column      |          Type          | Collation | Nullable | Default
-----------------+------------------------+-----------+----------+---------
 fk_professor_id | bigint                 |           | not null |
 id              | bigint                 |           | not null |
 name            | character varying(255) |           | not null |
Indexes:
    "masters_pkey" PRIMARY KEY, btree (id)
    "masters_fk_professor_id_key" UNIQUE CONSTRAINT, btree (fk_professor_id)
Foreign-key constraints:
    "fkip14gq0nq79dh0tsta2t3suuj" FOREIGN KEY (fk_professor_id) REFERENCES app_user(id)
Referenced by:
    TABLE "app_user" CONSTRAINT "fk54cihvrbjntyhxn6ge0e3vn2n" FOREIGN KEY (fk_masters_id) REFERENCES masters(id)
    TABLE "dissertation_topic_compatible_masters" CONSTRAINT "fkm95qmk1yu6gwtym3h3xppe5vh" FOREIGN KEY (master_id) REFERENCES masters(id)
```

```
testdb=# \d thesis_defense
                          Table "public.thesis_defense"
        Column         |              Type              | Collation | Nullable | Default
-----------------------+--------------------------------+-----------+----------+---------
 grade                 | double precision               |           |          |
 fk_president_id       | bigint                         |           |          |
 fk_thesis_execution_id | bigint                         |           | not null |
 id                    | bigint                         |           | not null |
 time                  | timestamp(6) without time zone |           | not null |
 dtype                 | character varying(31)          |           | not null |
 location              | character varying(255)         |           | not null |
Indexes:
    "thesis_defense_pkey" PRIMARY KEY, btree (id)
Foreign-key constraints:
    "fk92togxt6r2ncjbd5tmact6emi" FOREIGN KEY (fk_thesis_execution_id) REFERENCES thesis_execution(id)
    "fkp6kvb8x9c6olb24epbld0qaoi" FOREIGN KEY (fk_president_id) REFERENCES app_user(id)
```

```
testdb=# \d thesis_execution
                        Table "public.thesis_execution"
          Column           |          Type          | Collation | Nullable | Default
---------------------------+------------------------+-----------+----------+---------
 final_grade               | integer                |           |          |
 fk_consultant_id          | bigint                 |           |          |
 fk_dissertation_topic_id  | bigint                 |           | not null |
 fk_internal_advisor_id    | bigint                 |           | not null |
 fk_student_id             | bigint                 |           | not null |
 id                        | bigint                 |           | not null |
 dtype                     | character varying(31)  |           | not null |
 year_of_execution         | character varying(255) |           | not null |
Indexes:
    "thesis_execution_pkey" PRIMARY KEY, btree (id)
    "thesis_execution_fk_dissertation_topic_id_key" UNIQUE CONSTRAINT, btree (fk_dissertation_topic_id)
    "thesis_execution_fk_student_id_key" UNIQUE CONSTRAINT, btree (fk_student_id)
Foreign-key constraints:
    "fk30k9fa8uppo4sv3if19yxvdad" FOREIGN KEY (fk_dissertation_topic_id) REFERENCES dissertation_topic(id)
    "fk5wnb3ou0b9qpxicijh2ffa56q" FOREIGN KEY (fk_consultant_id) REFERENCES app_user(id)
    "fkbdi34n49uqpaxlwnipm6b74xc" FOREIGN KEY (fk_student_id) REFERENCES app_user(id)
    "fkjaaw0gl61dkkf18v0x5ocj3uk" FOREIGN KEY (fk_internal_advisor_id) REFERENCES app_user(id)
Referenced by:
    TABLE "thesis_defense" CONSTRAINT "fk92togxt6r2ncjbd5tmact6emi" FOREIGN KEY (fk_thesis_execution_id) REFERENCES thesis_execution(id)
```

Testes

Criamos 33 testes para validar a nossa implementação do diagrama de classes e das queries dos vários repositórios. Terão uma extensão e alcance maior quando existirem fluxos de informação e ação mais complexos. Neste momento testam apenas as interações básicas entre as várias tabelas e entidades.
Para executar os testes, basta correr *$bash test.sh*.

Limitações do trabalho

A classe “DissertationTopic” contém um ENUM “DissertationTopicType” que indica se é projeto ou dissertação. Esta informação pode ser verificada vendo que tipo de utilizador submeteu o tópico (getSubmitter()). Achamos que facilita a leitura caso seja preciso ir ver a base de dados.

Inicialmente quisemos usar a strategy Cascade.ALL, contudo percebemos que essa era a fonte de vários bugs e erros. Desta forma poderão existir alguns problemas com a persistência de certos dados na BD.

Apesar de ser possível ir buscar os dados através de uma FK, decidimos repetir certos atributos para que fosse mais fácil manusear os dados, evitando que as mesmas queries fossem repetidas constantemente.

Datas deveriam estar anotadas com @Column(columnDefinition=”DATE”).
Enum DissertationTopicType deveria estar anotada com @Enumerated(EnumType.STRING).
No final decidimos ficar com o tipo String para ser mais estável e mais fácil interpretar/manipular a tabela.