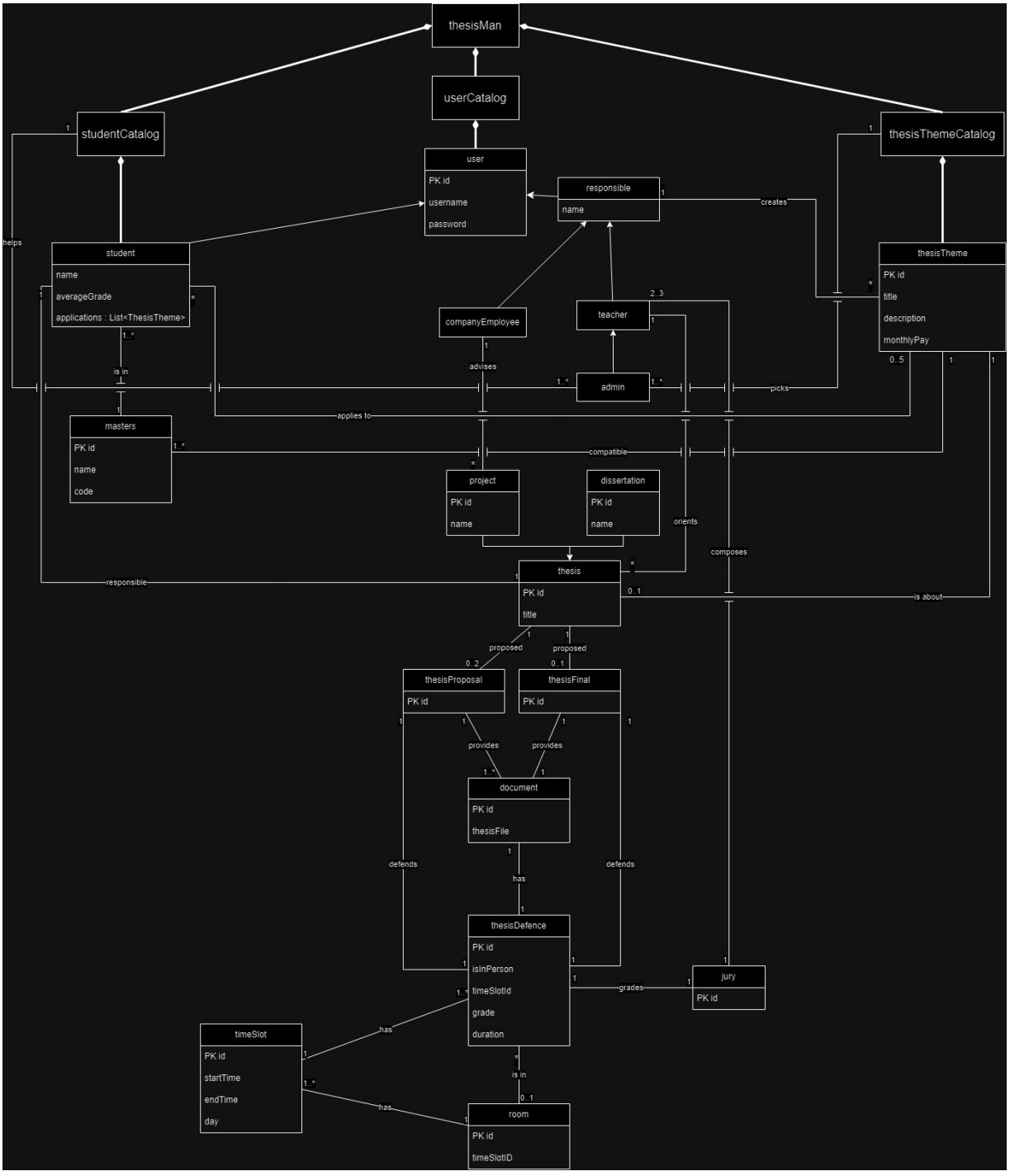
Relatório:

Domain Model:



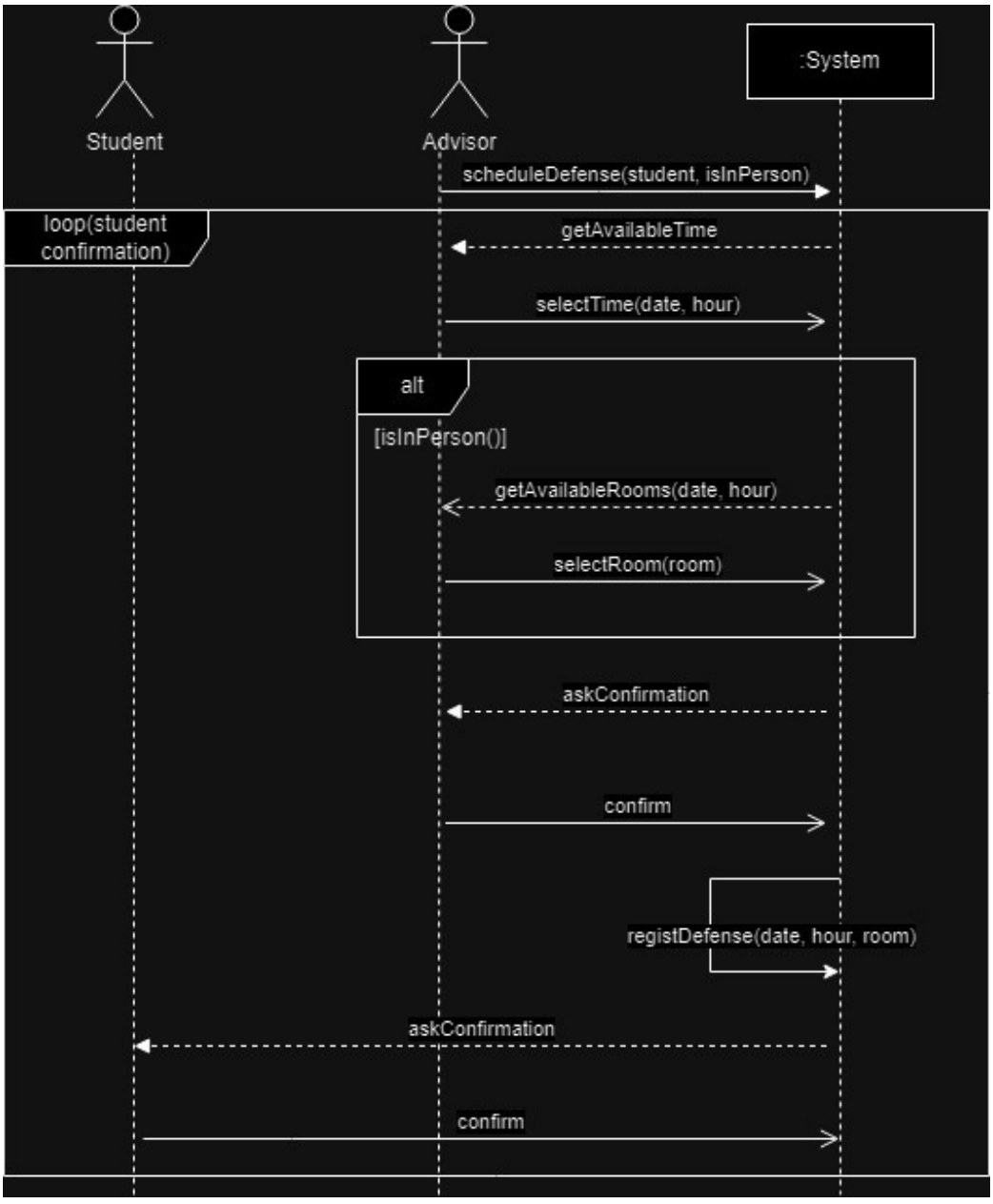
O sistema, denominado thesisMan, é o núcleo central de nossa aplicação. Ele conecta-se diretamente com o studentCatalog e o userCatalog, pois mesmo considerando que os estudantes são também usuários do sistema, considerámos benéfico ter uma tabela que os representa-se apenas a eles.

O admin está ligado a studentCatalog e a thesisThemeCatalog devido ao facto de o admin estar encarregado de dar a um estudante o tema da sua tese, no caso em que o aluno fica sem nenhuma.

Introduzimos a classe Responsible para melhor representar as ações tanto do companyEmployee quanto do teacher, pois ambos podem submeter temas de tese, estendendo, assim, essa classe para englobar suas funcionalidades, permitindo assim à ThesisTheme ter um responsible associado em vez de necessitar de associação às duas classes.

Utilizamos a herança de forma a não haver duplicação de código em exemplos como user com student e responsible pois, e responsible faz herança com teacher e companyEmployee e a thesis que faz herança com project e dissertation.

SSD:

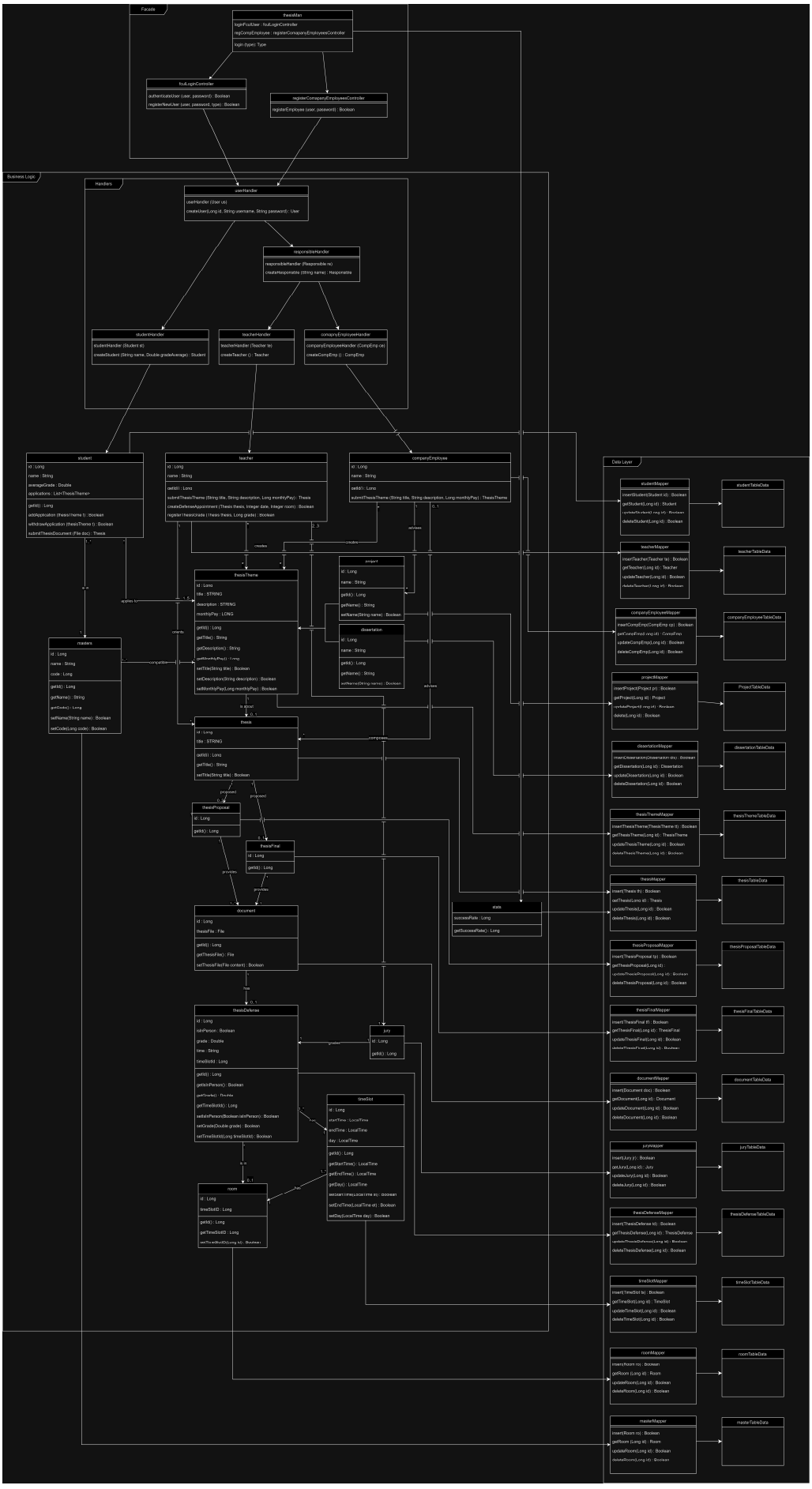


Cenário de sucesso

1. Orientador faz um pedido de marcação de defesa
2. Sistema mostra horarios disponiveis
3. Orientador escolhe um horario, no caso da defesa ser remota passa para o passo 6
4. Sistema mostra as salas disponiveis nesse horario
5. Orientador seleciona uma sala
6. Sistema pede confirmação da marcação
7. Orientador confirma
8. Sistema regista a data, horário e sala da defesa
9. Sistema notifica o aluno, da data, horário e sala de defesa
10. Aluno confirma a sua disponibilidade

O ssd foi feito baseado nesta interpretação dos passos do caso de uso. Foram usados getAvailableTime e getAvailableRoom, para impedir o utilizador de selecionar uma sala ou hora indisponível, pois apenas mostra as disponíveis. Alt foi usado para representar o passo 3, especificamente este aspeto "no caso da defesa ser remota passa para o passo 6", na situação de isInPerson() ser true, então ele mostra as salas disponíveis, se for false simplesmente passa à frente. Loop (student confirmation) foi usado para, o caso em que o estudante não demonstra disponibilidade, para o horário escolhido pelo orientador, logo repete o processo de escolha de sala e hora, até que ambos fiquem satisfeitos com o horário selecionado.

Diagram Model:


Facade
thesisMan
loginYouUser : YouLoginController
regCompEmployee : registerComapanyEmployeesController
login (type): Type
youLoginController
authenticateUser (user, password) : Boolean
registerNewUser (user, password, type) : Boolean
registerComapanyEmployeesController
registerEmployee (user, password) : Boolean
Business Logic
Handlers
userHandler
userHandler ((User us)
createUser(Long id , String username, String password): User
responsibleHandler
responsibleHandler (Responsible re)
createResponsible (String name) : Responsible
studentHandler
studentHandler (Student st)
createStudent (String name, Double gradeAverage) : Student
teacherHandler
teacherHandler (Teacher te)
createTeacher () : Teacher
companyEmployeeHandler
companyEmployeeHandler (CompEmp ce)
createCompEmp () : CompEmp
student
id : Long
name : String
averageGrade : Double
applications : List<ThesisTheme>
getId() : Long
addApplication (thesisTheme t) : Boolean
withdrawApplication (thesisTheme t) : Boolean
submitThesisDocument (File doc) : Thesis
teacher
id : Long
name : String
getId() : Long
submitThesisTheme (String title, String description, Long monthlyPay) : ThesisTheme
createDefenseAppointment (Thesis thesis, Integer date, Integer room) : Boolean
registerThesisGrade (Thesis thesis, Long grade) : Boolean
companyEmployee
id : Long
name : String
getId() : Long
submitThesisTheme (String title, String description, Long monthlyPay) : ThesisTheme
project
id : Long
name : String
getId() : Long
getName() : String
setName(String name) : Boolean
dissertation
id : Long
name : String
getId() : Long
getName() : String
setName(String name) : Boolean
thesisTheme
id : Long
title : STRING
description : STRING
monthlyPay : LONG
getId() : Long
getTitle() : String
getDescription() : String
getMonthlyPay() : Long
setTitle(String title) : Boolean
setDescription(String description) : Boolean
setMonthlyPay(Long monthlyPay) : Boolean
masters
id : Long
name : String
code : Long
getId() : Long
getName() : String
getCode() : Long
setName(String name) : Boolean
setCode(Long code) : Boolean
thesis
id : Long
title : STRING
getId() : Long
getTitle() : String
setTitle(String title) : Boolean
thesisProposal
id : Long
getId() : Long
thesisFinal
id : Long
getId() : Long
document
id : Long
thesisFile : File
getId() : Long
getThesisFile() : File
setThesisFile(File content) : Boolean
thesisDefense
id : Long
isInPerson : Boolean
grade : Double
time : String
timeSlotId : Long
getId() : Long
getIsInPerson() : Boolean
getGrade() : Double
getTime() : String
getTimeSlotId() : Long
setIsInPerson(Boolean isInPerson) : Boolean
setGrade(Double grade) : Boolean
setTimeSlotId(Long timeSlotId) : Boolean
jury
id : Long
getId() : Long
timeSlot
id : Long
startTime : LocalTime
endTime : LocalTime
day : LocalTime
getId() : Long
getStartTime() : LocalTime
getEndTime() : LocalTime
getDay() : LocalTime
setStartTime(LocalTime st) : Boolean
setEndTime(LocalTime et) : Boolean
setDay(LocalTime day) : Boolean
room
id : Long
timeSlotID : Long
getId() : Long
getTimeSlotID : Long
setTimeAndSlot(Long id) : Boolean
stats
successRate : Long
getSuccessRate() : Long
Data Layer
studentMapper
insertStudent(Student st) : Boolean
getStudent(Long id) : Student
updateStudent(Long id) : Boolean
deleteStudent(Long id) : Boolean
studentTableData
teacherMapper
insertTeacher(Teacher te) : Boolean
getTeacher(Long id) : Teacher
updateTeacher(Long id) : Boolean
deleteTeacher() : Boolean
teacherTableData
companyEmployeeMapper
insertCompEmp(CompEmp ce) : Boolean
getCompEmp(Long id) : CompEmp
updateCompEmp(Long id) : Boolean
deleteCompEmp(Long id) : Boolean
companyEmployeeTableData
projectMapper
insertProject(Project pr) : Boolean
getProject(Long id) : Project
deleteProject(Long id) : Boolean
ProjectTableData
dissertationMapper
insertDissertation(Dissertation dis) : Boolean
getDissertation(Long id) : Dissertation
updateDissertation(Long id) : Boolean
deleteDissertation(Long id) : Boolean
dissertationTableData
thesisThemeMapper
insertThesisTheme(ThesisTheme t) : Boolean
getThesisTheme(Long id) : ThesisTheme
updateThesisTheme(Long id) : Boolean
deleteThesisTheme(Long id) : Boolean
thesisThemeTableData
thesisMapper
insert(Thesis th) : Boolean
getThesis(Long id) : Thesis
updateThesis(Long id) : Boolean
deleteThesis(Long id) : Boolean
thesisTableData
thesisProposalMapper
insert(ThesisProposal tp) : Boolean
getThesisProposal(Long id) : updateThesisProposal(Long id) : Boolean
deleteThesisProposal(Long id) : Boolean
thesisProposalTableData
thesisFinalMapper
insert(ThesisFinal tf) : Boolean
getThesisFinal(Long id) : ThesisFinal
updateThesisFinal(Long id) : Boolean
deleteThesisFinal(Long id) : Boolean
thesisFinalTableData
documentMapper
insert(Document doc) : Boolean
getDocument(Long id) : Document
updateDocument(Long id) : Boolean
deleteDocument(Long id) : Boolean
documentTableData
juryMapper
insert(Jury j) : Boolean
getJury(Long id) : Jury
updateJury(Long id) : Boolean
deleteJury(Long id) : Boolean
juryTableData
thesisDefenseMapper
insert(ThesisDefense td) : Boolean
getThesisDefense(Long id) : ThesisDefense
updateThesisDefense(Long id) : Boolean
deleteThesisDefense(Long id) : Boolean
thesisDefenseTableData
timeSlotMapper
insert(TimeSlot ts) : Boolean
getTimeSlot(Long id) : TimeSlot
updateTimeSlot(Long id) : Boolean
deleteTimeSlot(Long id) : Boolean
timeSlotTableData
roomMapper
insert(Room ro) : Boolean
getRoom (Long id) : Room
updateRoom(Long id) : Boolean
deleteRoom(Long id) : Boolean
roomTableData
masterMapper
insert(Room ro) : Boolean
getRoom (Long id) : Room
updateRoom(Long id) : Boolean
deleteRoom(Long id) : Boolean
masterTableData

Após uma análise dos Diagram Models disponíveis, optamos por implementar Handlers para a criação dos users (e responsible, students, teachers e companyEmployees), identificando esta abordagem como a mais eficaz. Esses Handlers estão conectados às classes representadas no Domain Model, as quais, por sua vez, estão vinculadas a Mappers (que é a solução apropriada quando se usa Domain Model) para a obtenção precisa de informações necessárias, direcionando-se mais diretamente aos dados presentes nas respectivas TableData.

O stats está ligado ao thesisMapper para recolher a informação deste acerca de quantos alunos passaram, de forma ao thesisMan poder calcular a taxa de sucesso.

JPA:

Relatório sobre Implementação de Código com JPA

O desenvolvimento utilizando JPA (Java Persistence API) foi estruturado com base no modelo de domínio. A maioria das classes presentes nesse modelo foi replicada no contexto do JPA, uma vez que essas classes representam entidades que requerem persistência de dados e, portanto, é benéfico tê-las armazenadas no sistema.

A exceção notável é a classe Admin, ausente no contexto do JPA. Esta decisão foi tomada devido ao fato de que a classe Admin adiciona funcionalidades extras à classe Teacher, sem necessidade de armazenar informações próprias.

Os identificadores (IDs) de todas as entidades são gerados automaticamente pelo sistema. Além disso, identificou-se a necessidade de incluir a classe TimeSlot (que inicialmente era uma variável String) no modelo de domínio após a implementação do JPA, devido à necessidade de persistir informações relacionadas a intervalos de tempo em que as salas se encontram disponíveis.

É importante ressaltar que diversas classes JPA são inicializadas com listas vazias, as quais serão preenchidas posteriormente. Além disso, certos atributos não são inicializados no construtor, pois sua atribuição ocorre em etapas posteriores. Por exemplo, o atributo "grade" da classe ThesisDefense.

Uma nova classe, Responsible, foi introduzida posteriormente ao desenvolvimento inicial para permitir que ThesisTheme esteja associada exclusivamente a Responsible, sem a necessidade de associação direta com Teacher ou Company, dependendo de quem a criou.

No caso específico dos estudantes (Students), estes podem estar associados a várias instâncias de ThesisTheme, e a verificação de limites é realizada não diretamente na base de dados, mas sim na camada de businessLogic, no momento da adição (add).

Todas as associações do tipo ManyToOne, OneToMany, etc., foram estabelecidas com base nas relações observadas no modelo de domínio, garantindo assim a consistência e integridade do sistema.