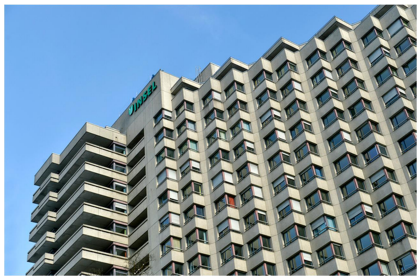
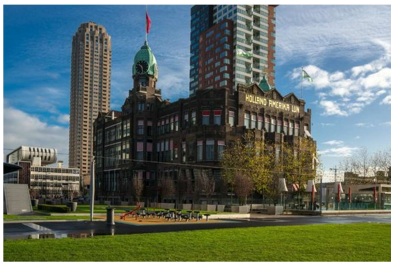
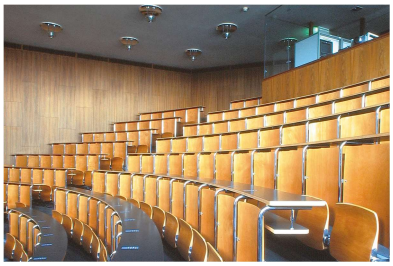
Documen-tation

Groupe Pink







Content

SRS-Project

Introduction

The goal of this project is to learn how to deal with development tools like EGit, Maven, ... and with development processes such as UML, Scrum, ... For this we have to design a *Smart Reservation System (SRS)*.
This program must be suitable for different types of enterprises. Be it sports clubs that rent out halls or hotels, which rent out meeting or overnight rooms, or a university, which wants to book teaching rooms for courses. The system shall allow its users to easily book any type of rooms.
The program has to run on different applications / devices. Users shall access the system via various kinds of applications such as standard Web interfaces and apps running on smartphones or tablets.

Requirements of the SR-System

- Reservations must be made easily
- Reservations can be simple time ranges or periodic events
- Time grids shall be made available in order to align reservations with time boundaries of, for example, given schedules.

UML

Information bez. StarUML und generell zu den Class-/ Object-Diagrammen, Domain Model

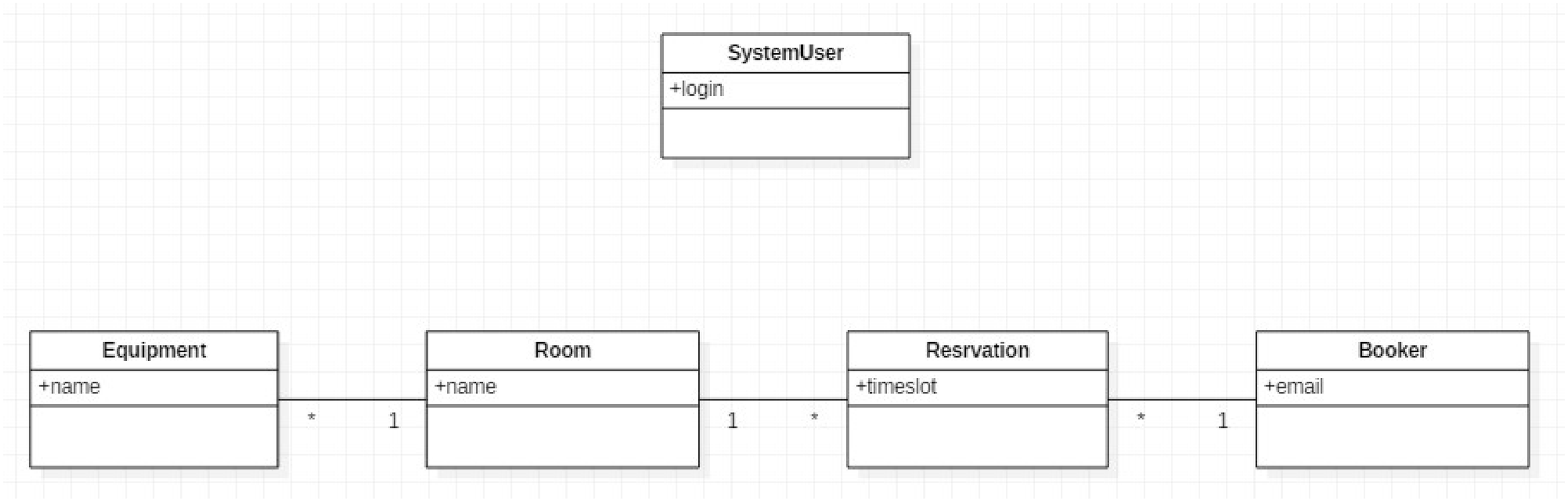
- Domain Model:
  - In unserem SRS kann pro Zeiteinheit nur eine Reservation pro Raum getätigt werden.
  - Zu einer Reservation gehört genau ein Booker und genau ein Raum

- Class-Diagram:
  - Bezüglich +, -, #, ~ vor den Attributen: über die Software StarUML ist es nicht möglich keine Informationen bez. der Visibility zu liefern, man kann nicht Nichts auswählen

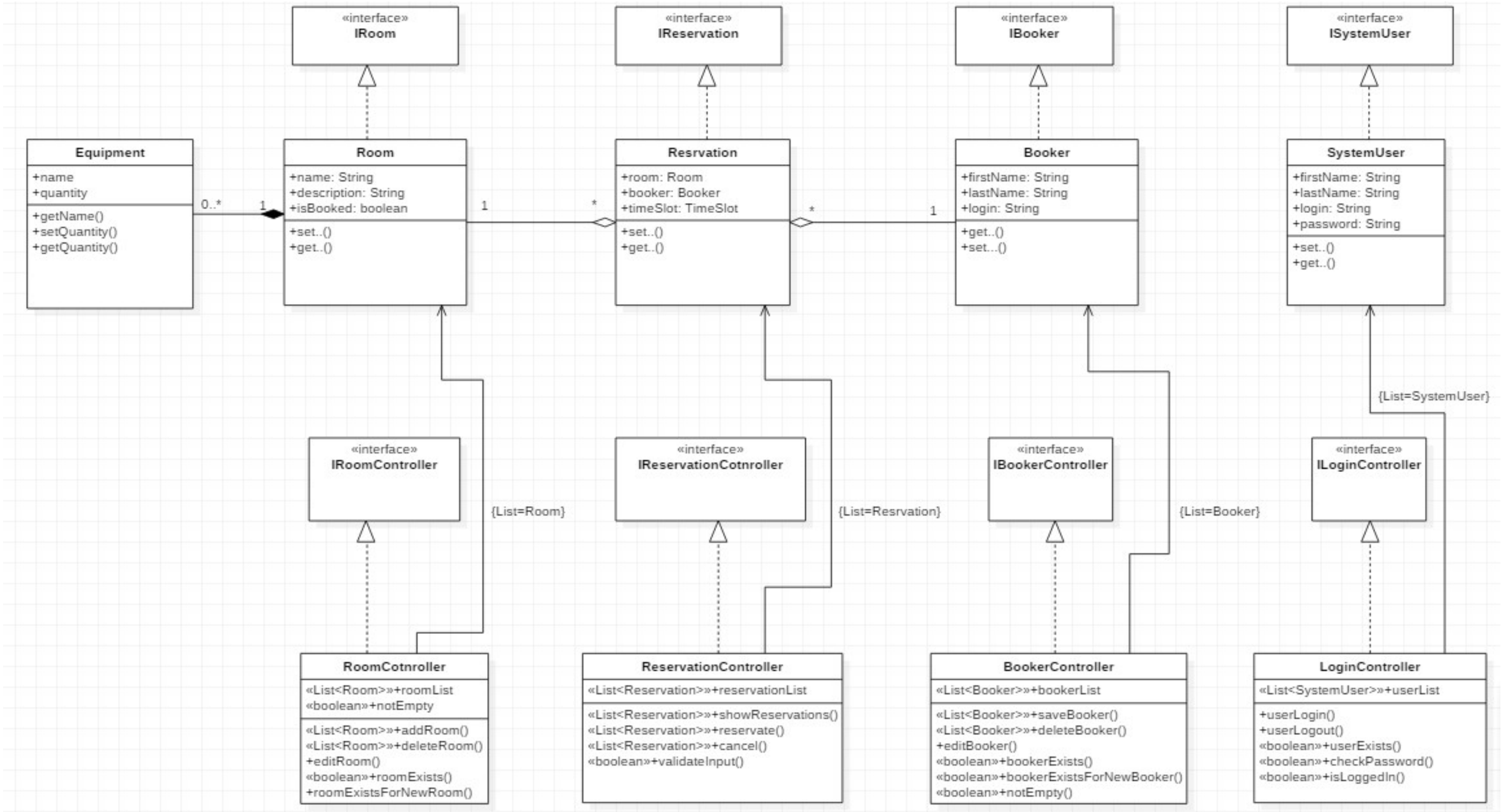
- Object-Diagram:
  - Vor den jeweiligen Objekten kann man nicht ein Doppelpunkt eingeben, diese Eingabe wird nicht akzeptiert

- Package-Diagram:
  - Die Klassen / Interfaces die sich in einem Package befinden, werden nicht in den jeweiligen Packages angezeigt. Die Zuteilung hat sich gegenüber Sprint03 so verändert, dass ein neues Package hinzugefügt worden ist, welches als Interface (im Sinne von Schnittstelle des ganzen Systems) fungiert, so können die Klassen auf dieses Package zugreifen, wenn eine Referenz benötigt wird. Dies um Zyklen zu verhindern und die Stabilität des Systems zu gewährleisten.

Domain Model

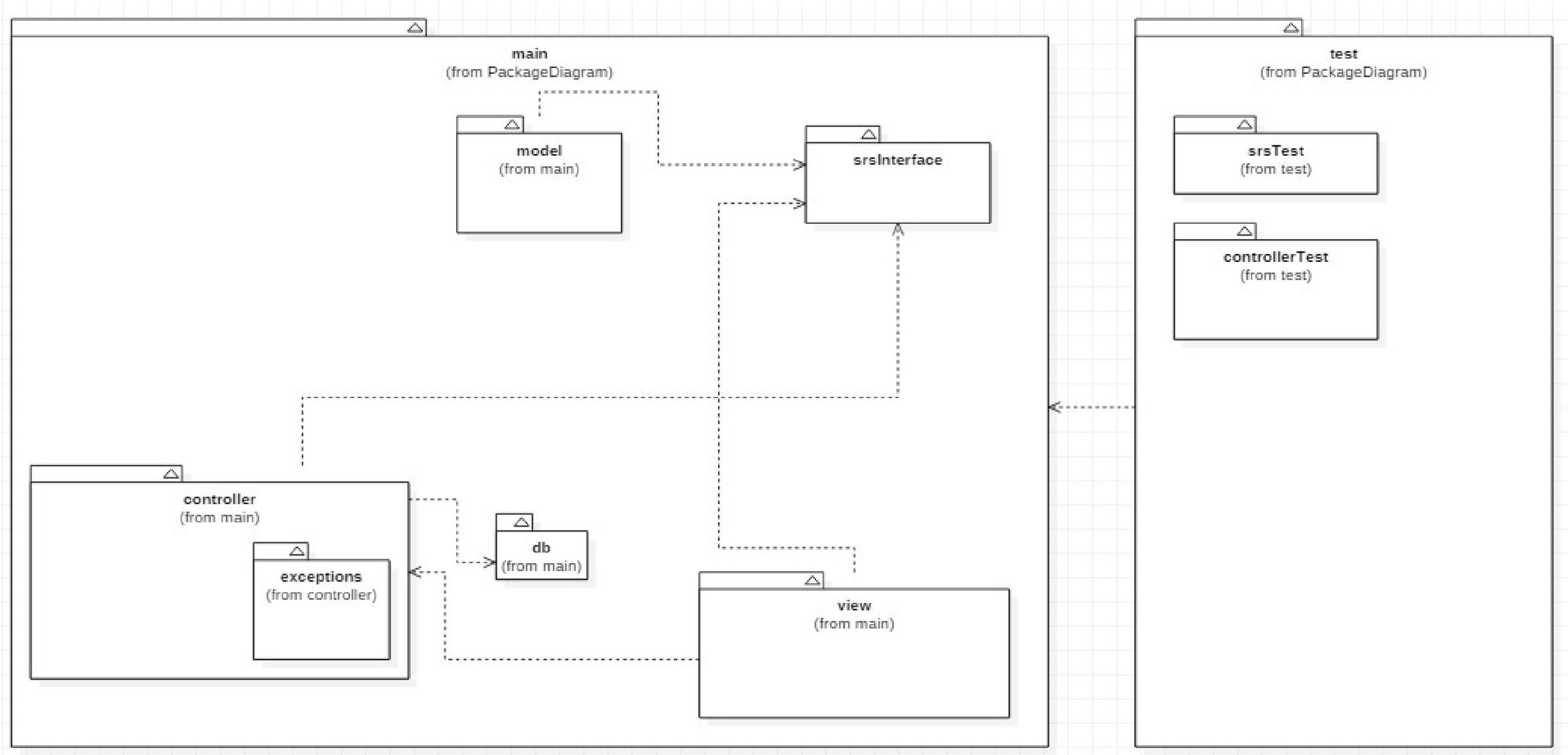


Class diagram



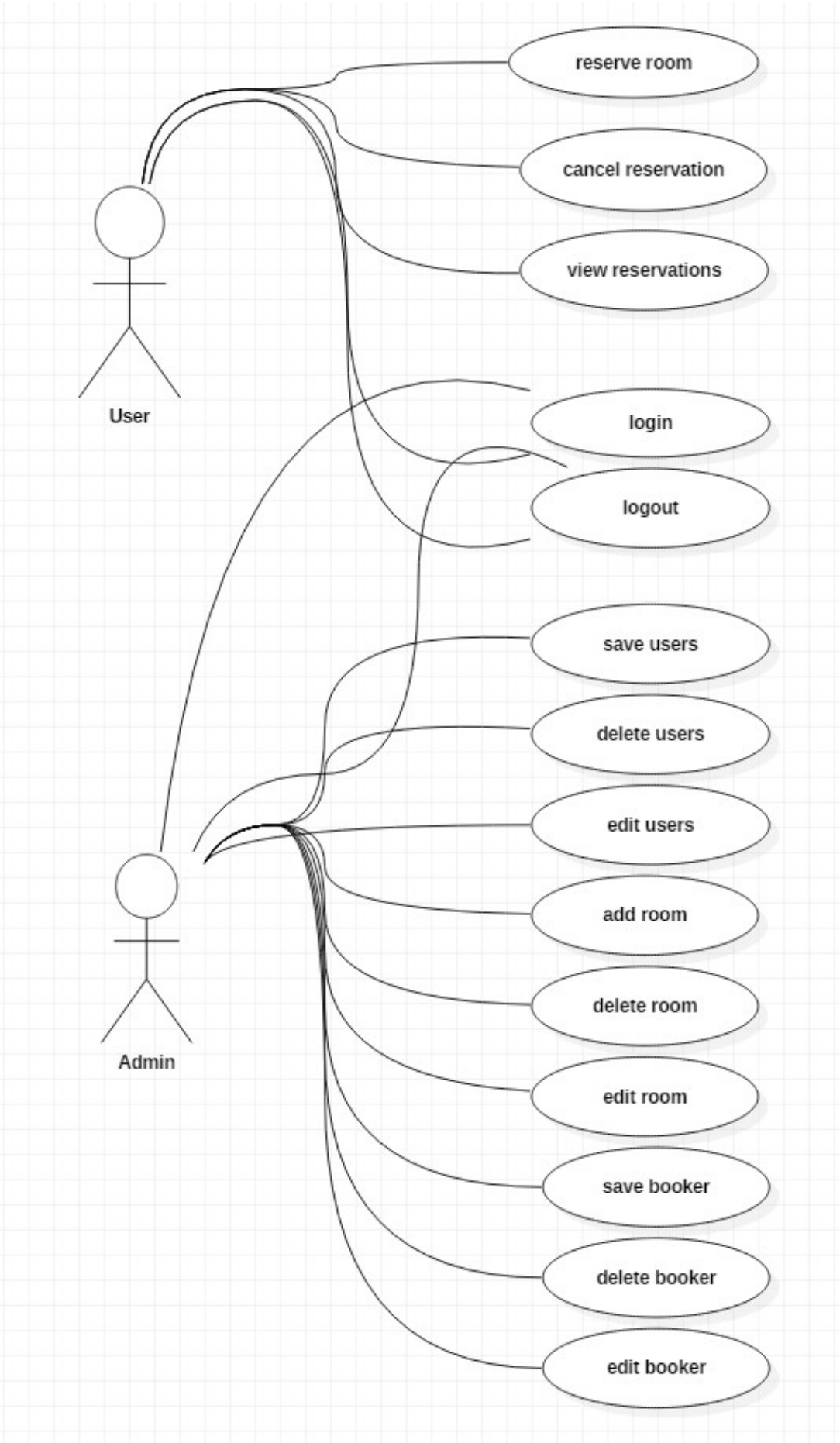
Object diagram

Package diagram



Use Case Diagram

Use Cases



User Stories

User Stories

Title	Description	Success	Fail
Register user	As an administrator, I want to register new users, so they can reserve rooms.	Users can only book rooms, if they are registered.	The reason of error will be shown.
Delete user	As an administrator, I want to delete a user, so he can't reserve rooms anymore.	Users, who don't have the permit to book rooms, must be deleted.	The reason of error will be shown.
Edit user	As an administrator, I want to edit a user, so the data are up to date.	Data from the users have to be up to date.	The reason of error will be shown.
Insert room	As an administrator, I want to insert a room, so users can reserve it.	Rooms can only be reserved if they are inserted in memory.	The reason of error will be shown.
Delete room	As an administrator, I want to delete a room, so users can't reserve it anymore.	Rooms which do not exist anymore have to be deleted, so they can't be booked by users.	The reason of error will be shown.
Edit room	As an administrator I want to edit a room, so users can either reserve again or don't reserve anymore.	Rooms are not always for booking, perhaps there is a renovation to do and then user cannot book these rooms.	The reason of error will be shown.
Reserve room	As a user, I want to reserve a room, so a booker can utilize it.	When bookers want to utilize a room, they have to be reserved in advance.	The reason of error will be shown.
Cancel room	As a user, I want to cancel a reservation, so it's available again.	When bookers have booked a room, they may be cancel the reservation and this have to be adapted in the reservation-memory.	The reason of error will be shown.
View room reservations	As a user, I want to see all reservations of a room, so I can see which rooms are occupied.	To see, which rooms are free for bookers, I need a view of all free / occupied rooms.	The reason of error will be shown.
View booker reservations	As a user, I want to see all reservations from a booker, so I can calculate which customers book often rooms.	To see, which bookers book often rooms, I need a view with the reservations form the past.	The reason of error will be shown.

Backlog

Product Backlog

Prio	Task	New	Active	Resolved	Closed	Description	Estimated time [h]
1	*Corrections*				X		*1*
1	Create first classes.				X	Create classes booker, room, reservation.	8
1	Reservation controller				X	Create the reservationController for reserving a room, for cancel a reservation and for view the reservations	1
1	Login controller				X	Create the loginController for admins/user to login.	1
1	User controller				X	Create the userController for register a user, delete a user, edit a user.	1
1	Room controller				X	Create the roomController for adding a room, delete a room, edit a room.	1
1	Booker controller				X	Create the booker controller for register a booker, delete a booker, edit a booker.	1
1	*Documentation*				X		*2*
2	Login				X	As an administrator/user, I want to login, so I can use the SRS.	6
2	Register User				X	As an administrator, I want to register new users, so they can reserve rooms.	6
2	Delete User				X	As an administrator, I want to delete a user, so he can't reserve rooms anymore.	6
2	Edit User				X	As an administrator, I want to edit a user, so the data are up to date.	6
2	Insert Room				X	As an administrator, I want to insert a room, so users can reserve it.	6
2	Delete Room				X	As an administrator, I want to delete a room, so users can't reserve it anymore.	6
2	Edit Room				X	As an administrator I want to edit a room, so users can either reserve again or don't reserve anymore.	6
2	Register Booker				X	As an user I want to register a booker so they can reserve rooms.	6

Prio	Task	New	Active	Resolved	Closed	Description	Estimated time [h]
2	Delete Booker				X	As an user I want to delete a booker so the database is up to date	6
2	Edit Booker				X	As an user I want to edit a booker so the database is up to date.	6
2	Reserve Room					As an administrator I want to edit a room, so users can either reserve again or don't reserve anymore.	6
2	Cancel Reservation				X	As a user, I want to cancel a reservation, so it's available again.	6
2	View room reservations				X	As a user, I want to see all reservations of a room, so I can see which rooms are occupied.	6
2	View booker reservations				X	As a user, I want to see all reservations from a booker, so I can calculate which customers book often rooms.	6
2	Database				X	Create a database and the necessery connection.	6
3	UI		X			Create the user interface.	20

Sprint Backlog

Sprint 2

Prio	Task	New	Active	Resolved	Closed	Description	Estimated time [h]
Code: Implementation							
1	Corrections from S1				X		1
1	Reservation Controller Interface				X	Create the interface for booking a room.	0.5
2	Reservation Controller Class				X	Implement the interface and implement the logic.	1
2	Test Reservation Controller				X	Test the reservation controller.	2
1	Login Controller Interface				X	Create the login controller for admins/user to login/logout the SRS.	0.5
2	Login Controller Class				X	Implement the interface and implement the logic.	1
2	Test Login Controller				X	Test the login controller.	2
2	Booking controller Interface				X	Create the interface for administrate the bookers, save/delete/edit a booker.	0.5
2	Booking controller Class				X	Implement the interface and implement the logic.	1
2	Test Booking Controller				X	Test the booking controller.	2
1	Booker controller Interface				X	Create the interface for reservating/cancel a room.	0.5
2	Booker controller Class				X	Implement the interface and implement the logic.	1
2	Test Booker Controller				X	Test the reservation controller.	2
Code: Correction / Fishing							
1	Check the coverage rate of tests.				X	Add missing tests	? De-pends…

Prio	Task	New	Active	Resolved	Closed	Description	Estimated time [h]
2	Check warinings/errors				X	Check if errors/warings exists.	0.1
2	Correct the warning/errors				X	Correct errors/warings if they exist.	? De-pends…
3	Javadoc				X	Create missing Javadoc	2
Documentation							
1	Adapt domain model				X	Adapt domain model	1
2	Fishing PBL / SBL				X	Fishing PBL / SBL	0.5
3	Create PDF				X	Create PDF	0.2
4	Links				X	Adapt / create links.	0.3
5	Upload				X	Upload on git.	0.2

Sprint 3

Prio	Task	New	Active	Resolved	Closed	Description	Estimated time [h]
Code: Implementation							
1	*Corrections from sprint02*					*Nothing received*	*???*
1	SOLID package design				X	Structure your code in packages by respecting the SOLID design principles	15
1	Consider the Single Response Principle				X		3
1	Consider the Open Closed Principle				X		3
1	Consider the Liskov's Substitution Principle				X		3
1	Consider the Interface Segregation Principle				X		3
1	Consider the Dependency Inversion Principle				X		3
1	Provide object persistence				X	Provide object persistence; the use of JPA is recommended	10
Code: Correction / Fishing							
1	Check the coverage rate of tests.				X	Add missing tests	0.5
2	Check warinings/errors				X	Check if errors/warings exists.	0.5
2	Correct the warning/errors				X	Correct errors/warings if they exist.	0.5
3	Javadoc				X	Create missing Javadoc	2

Prio	Task	New	Active	Resolved	Closed	Description	Estimated time [h]
Documentation							
1	Adapt domain model				X	Adapt domain model	0.3
1	Create package diagram				X	Create Package diagram	0.4
2	Fishing PBL / SBL				X	Fishing PBL / SBL	0.5
3	Create PDF				X	Create PDF	0.3
4	Links				X	Adapt / create links	0.3
5	Upload				X	Upload on git	0.1

Sprint 4

Prio	Task	New	Active	Resolved	Closed	Description	Estimated time [h]
Code: Implementation							
1	Corrections from sprint02, sprint03				X		2
1	SOLID package design				X	Structure your code in packages by respecting the SOLID design principles	2
1	Consider the Single Response Principle				X		0.5
1	Consider the Open Closed Principle				X		0.5
1	Consider the Liskov's Substitution Principle				X		0.5
1	Consider the Interface Segregation Principle				X		0.5
1	Consider the Dependency Inversion Principle				X		0.5
1	Provide object persistence (Equipment)				X	Provide object persistence; the use of JPA is recommended	1
1	Provide object persistence (Room)				X	Provide object persistence; the use of JPA is recommended	1
1	Provide object persistence (Booker)				X	Provide object persistence; the use of JPA is recommended	1
1	Provide object persistence (Reservation)				X	Provide object persistence; the use of JPA is recommended	1
1	Provide object persistence (SRS-Users)				X	Provide object persistence; the use of JPA is recommended	1
1	GUI		X			To deliver a solution for the user interface based on Vaadin	6

Prio	Task	New	Active	Resolved	Closed	Description	Estimated time [h]
Code: Correction / Fishing							
1	Check the coverage rate of tests.				X	Add missing tests	0.5
2	Check warinings/errors				X	Check if errors/warings exists.	0.1
2	Correct the warning/errors				X	Correct errors/warings if they exist.	0.5
3	Javadoc				X	Create missing Javadoc	0.5
Documentation							
1	Adapt domain model				X	Adapt domain model	1
1	Create package diagram				X	Create Package diagram	0.5
2	Fishing PBL / SBL				X	Fishing PBL / SBL	0.3
3	Create PDF				X	Create PDF	0.3
4	Links				X	Adapt / create links	0.2
5	Upload				X	Upload on git	0.1

Glossary

General

SRS	Smart Reservation System

Artefacts

Sprint Backlog	Items/stories from Product Backlog (only a few hours)
Release Backlog	A number of items from the Product Backlog is selected for each release
Product Backlog	Product Owner presents Product Backlog with all relevant user stories. Also the priority is mentioned.
User Stories	A written description of a functionality valuable to either a user (or owner) that will use the software system in future.
Use Case	Use cases are written stories.

Ratio

C_e – *Efferent Coupling*	*Die Anzahl der anderen Pakete, die die Klassen im Paket abhängen, ist ein Indikator für die Unabhängigkeit des Pakets.*
C_a – *Afferent Coupling*	*Die Anzahl der anderen Pakete, die von Klassen in diesem Paket abhängen, ist ein Indikator für die Verantwortung des Pakets.*
I	*Das Verhältnis der efferenten Kopplung C_e zur Gesamtkupplung:* $$I = \frac{C_e}{C_e + C_a}$$ *I ist eine Kennzahl die einem Indikator für die Widerstandsfähigkeit des Pakets steht, wobei gilt: Die Instabilität liegt im Bereich von 0 bis 1, eine Instabilität von 0 weist auf ein maximal stabiles Modul hin, eine von 1 auf ein maximal instabiles Modul.* *Quelle: https://de.wikipedia.org/wiki/Prinzipien_objektorientierten_Designs#Packaging-Prinzipien*

Stakeholders

Development Team	The Development Team is responsible for developing functionality and for delivering shippable increments of the SRS at the end of each Sprint. Developer Teams are self-managing, self-organizing, and cross-functional and they are responsible for figuring out how to turn Product Backlog into an increment of functionality within iteration and managing their own work to do so.
Scrum Master	Person who builds the relationship between customer and development-team. Scrum Master participates on all Scrum-Meetings. The Scrum Master is responsible for the Scrum process, facilitate the Team and protect the team from outside noise.
Product Owner	The Product Owner represents the product's stakeholders and the voice of the customer and is accountable for ensuring that the team delivers value to the business. The Product Owner writes customer-centric items (user stories), prioritizes them based on importance and dependencies, and adds them to the Product Backlog. The Product Owner is responsible for maximizing the ROI. The Product Owner priories the Product Backlog

Customer	DUE1
Investor	DUE1

User groups

User	*Persons who will use the SRS and book rooms for the booker.*
Administrator	*Persons who has the responsibilities of all users and all rooms (register new rooms, edit rooms, delete rooms)*
Booker	*The person who will book a room.*