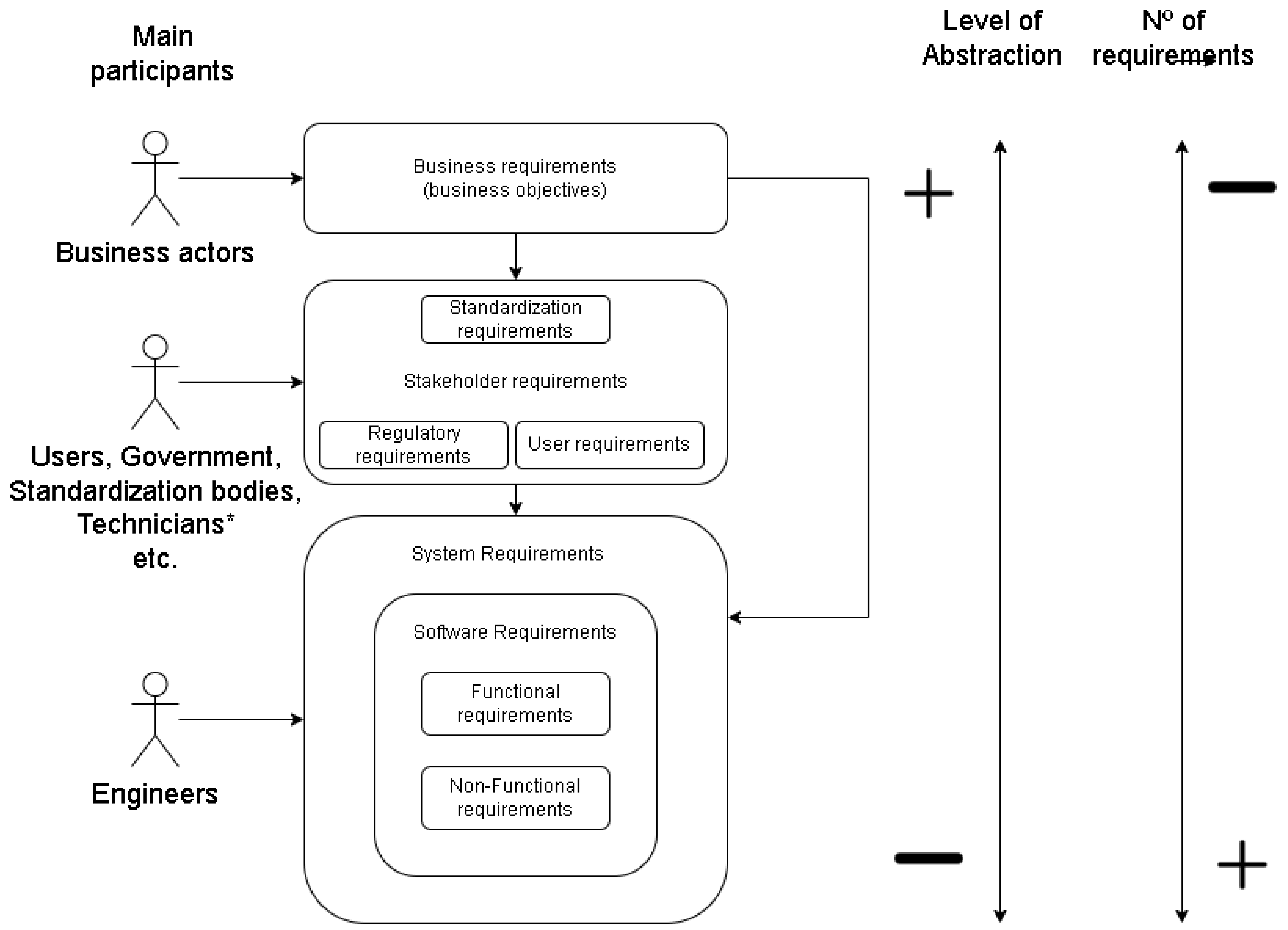
AirBnB

An example from business requirements to models

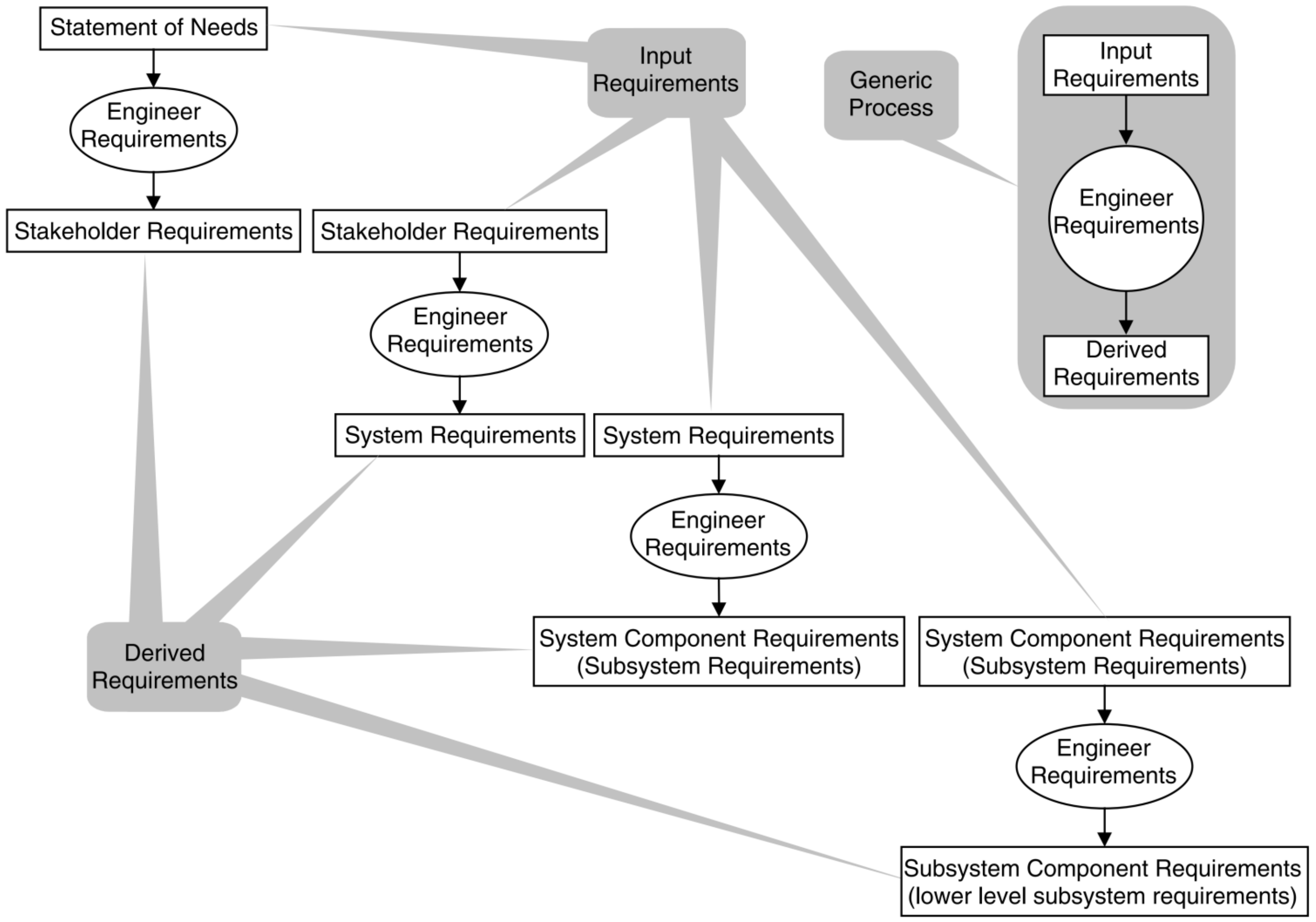
Jose María Alvarez Rodríguez

Objectives

- To state some business requirements
- To derive some software requirements
- To populate a conceptual
- To explore different architectural models (logical view)
- (Optional) To show other views


Main participants
Level of Abstraction
Nº of requirements
Business actors
Users, Government, Standardization bodies, Technicians* etc.
Engineers
Business requirements (business objectives)
Standardization requirements
Stakeholder requirements
Regulatory requirements
User requirements
System Requirements
Software Requirements
Functional requirements
Non-Functional requirements
+
−
−
+

Generic Requirements Engineering process



Requirements engineering process as a spiral view

- Everything starts with some business needs.
- Then user requirements are gathered.
- These requirements are landed into functionalities from a system perspective.
- Finally, requirements are verified and validated through different techniques.

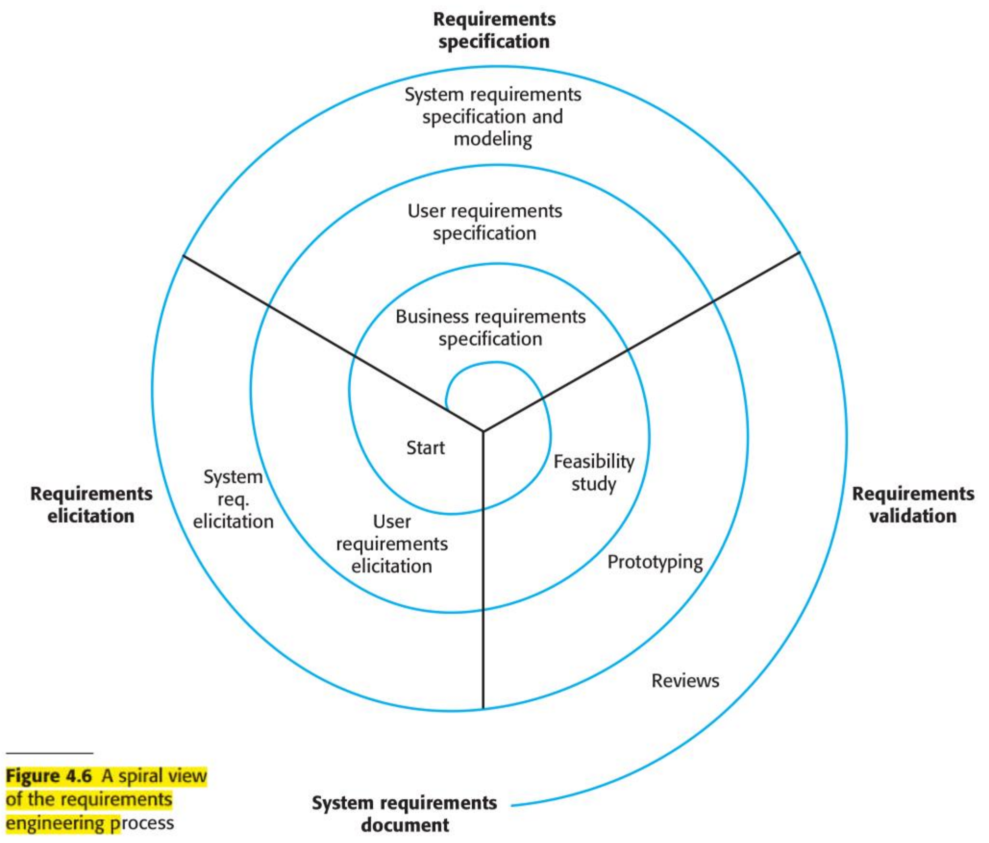

Requirements specification
System requirements specification and modeling
User requirements specification
Business requirements specification
Start
Feasibility study
Requirements elicitation
System req. elicitation
User requirements elicitation
Prototyping
Requirements validation
Reviews
System requirements document

Figure 4.6 A spiral view of the requirements engineering process

Business requirements

Id	Text
BR1	The software platform shall offer users the location of the closest apartments.
BR2	The software platform shall connect to the most common social network platforms.
BR3	The software platform shall allow users to make bookings.
BR4	The software platform shall be secure.
BR5	The software platform shall be available in all possible channels, at least web and mobile.

https://es.slideshare.net/PitchDeckCoach/airbnb-first-pitch-deck-editable

User requirements

Id	Text	Derived from
UR1	The user shall be able to show the closest apartments in a map.	BR1
UR2	The user shall be able to manage bookings.	BR3

- User requirements:
  - Gathering needs from the **user perspective**

Software requirements

- Specification of functionalities from the **system perspective**
  - System, subsystem or component in charge of delivering some functionalities under certain conditions
- Non-functional requirements
  - Cross-cutting concerns to ALL functionalities, e.g. security, availability, accessibility
    - -ilities of the system

Software requirements

Id	Text	Derived from	Type
FR1	The AirBnB recommendation subsystem shall show a map of the apartments available in a range of 500 meters in regards to the current GPS position.	BR1, UR1	Functional
FR2	The AirBnB authentication subsystem shall allow users to log in with their existing credentials for Google or Facebook.	BR2	Functional
FR3	The AirBnB booking subsystem shall allow registered users to make a booking establishing a check-in and check-out date covering a máximum of 15 days.	BR3, UR2	Functional
FR4	The AirBnB booking subsystem shall allow registered users to check out a booking making a payment through Paypal, Debit or Credit Card.	BR3	Functional
…			
NFR1	The AirBnB platform shall provide any communication under the SSL protocol.	BR4	Non-functional
NFR2	The AirBnB platform shall be deployed in the following platforms: IoS 10 or higher and Android 4 or higher.	BR5	Non-functional

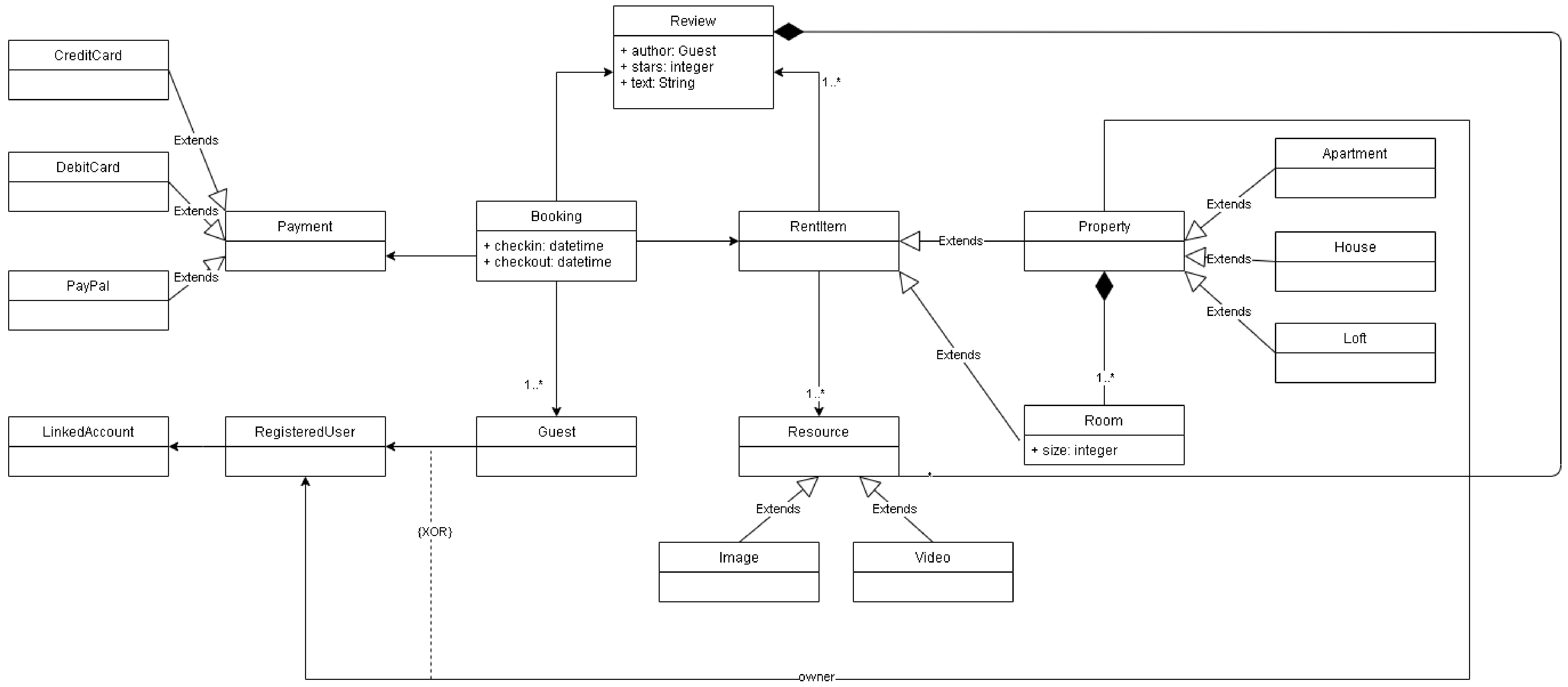
Conceptual model

- Graphical view of the requirements
- Domain vocabulary: entities and relationships

Domain vocabulary (partially)

Id	Text	Derived from	Type
FR1	The AirBnB recommendation subsystem shall show a map of the apartments available in a range of 500 meters in regards to the current GPS position.	BR1	Functional
FR2	The AirBnB authentication subsystem shall allow users to log in with one of their existing credentials for Google or Facebook.	BR2	Functional
FR3	The AirBnB booking subsystem shall allow registered users to make a booking establishing a check-in and checkout date covering a max. of 15 days.	BR3	Functional
FR4	The AirBnB booking subsystem shall allow registered users to check out a booking making a payment through Paypal, Debit or Credit Card.	BR3	Functional
…			
NFR1	The AirBnB platform shall provide any communication under the SSL protocol.	BR4	Non-functional
NFR2	The AirBnB platform shall be deployed in the following platforms: IoS 10 or higher and Android 4 or higher.	BR5	Non-functional

Conceptual model



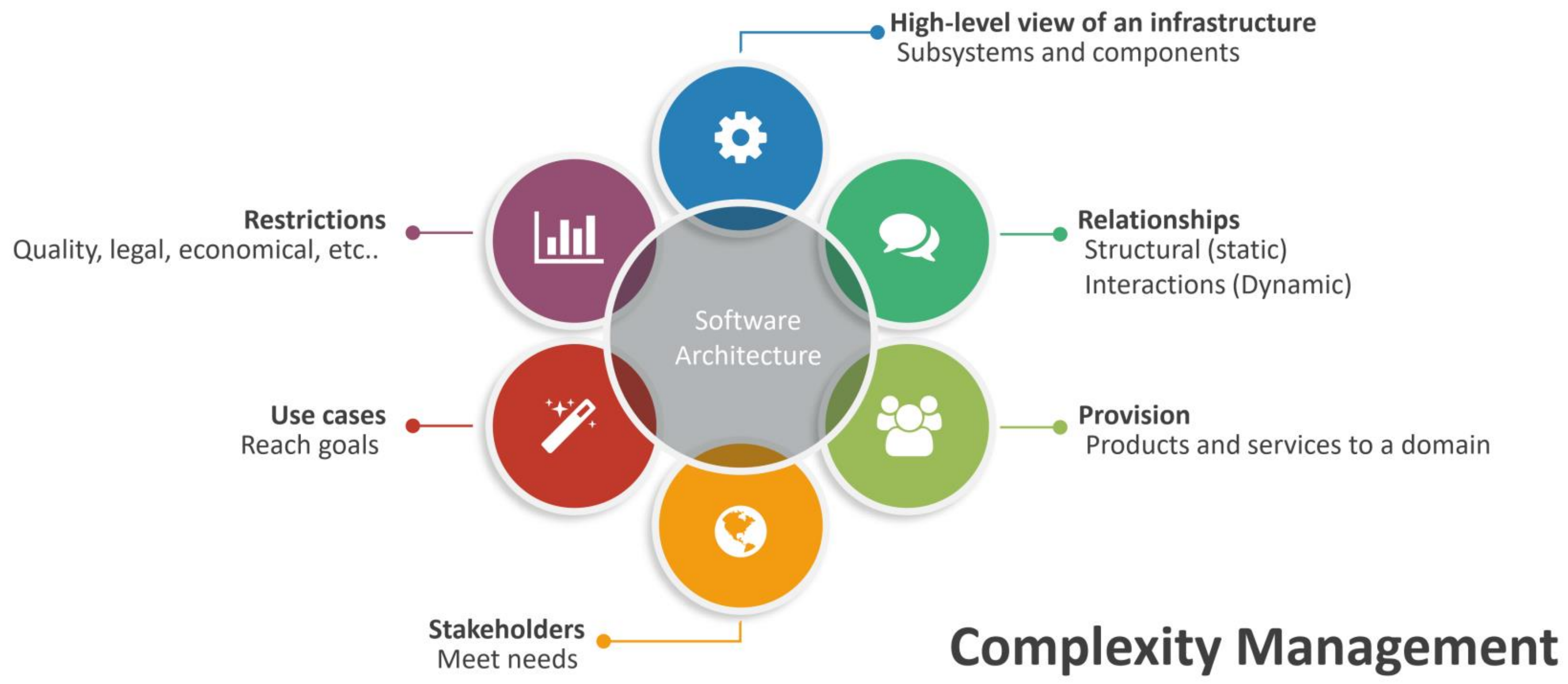
Conceptual model remarks

- It assumes several descriptions of some entities: apartment or booking.
  - More requirements will fit into this conceptualization.
- It includes two attributes in booking because they are very clear but it is not establishing any data type for them.
  - We are specifying not making a detailed design
- **It helps us to represent the information that will be managed by the platform.**
- It does not include user interface elements.

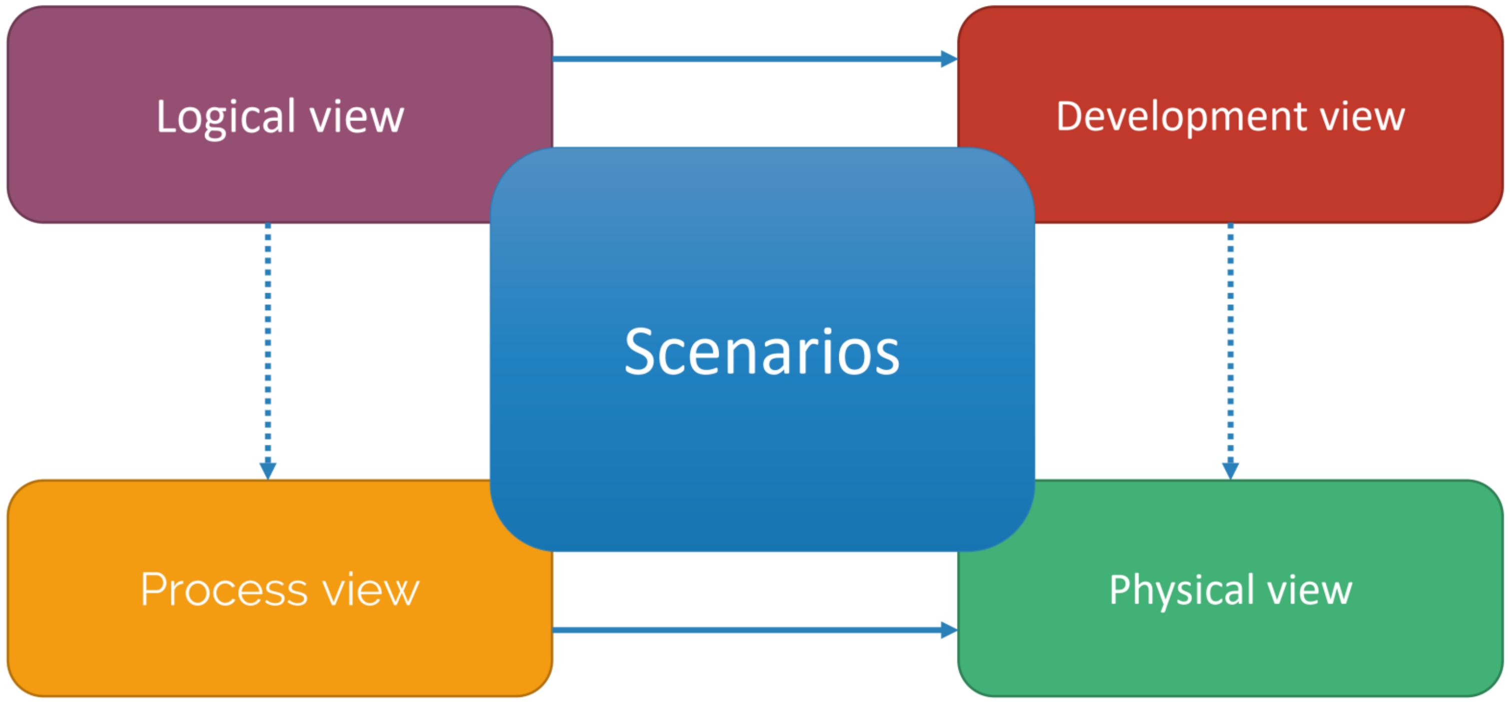
Architectural model: principles

*"<system> **fundamental concepts** or **properties** of a system in **its environment** embodied in its **elements**, **relationships**, and in the **principles** of its **design** and **evolution**."*

Source: ISO/IEC/IEEE 42010:2011 Systems and software engineering —Architecture description



Architectural model: 4+1 model



Architectural model: 4+1 model summary

	Logical (conceptual)	Process (runtime)	Development (implementation)	Physical (deployment)
Concern	Information model	Concurrency, synchronization Interaction	Software organization in the development environment	Mapping software-hardware
Stakeholders	End-users	System integrators	Developers	IT engineers
Requirements	Functional	Performance Availability Reliability Concurrency Distribution Security	Software management Reuse Portability Maintainability Platform and technological restrictions	Performance Availability Reliability Scalability Topology Communications
Notation	Class and/or component diagram in UML	Sequence diagram in UML	Component and/or package diagram in UML	Deployment diagram in UML

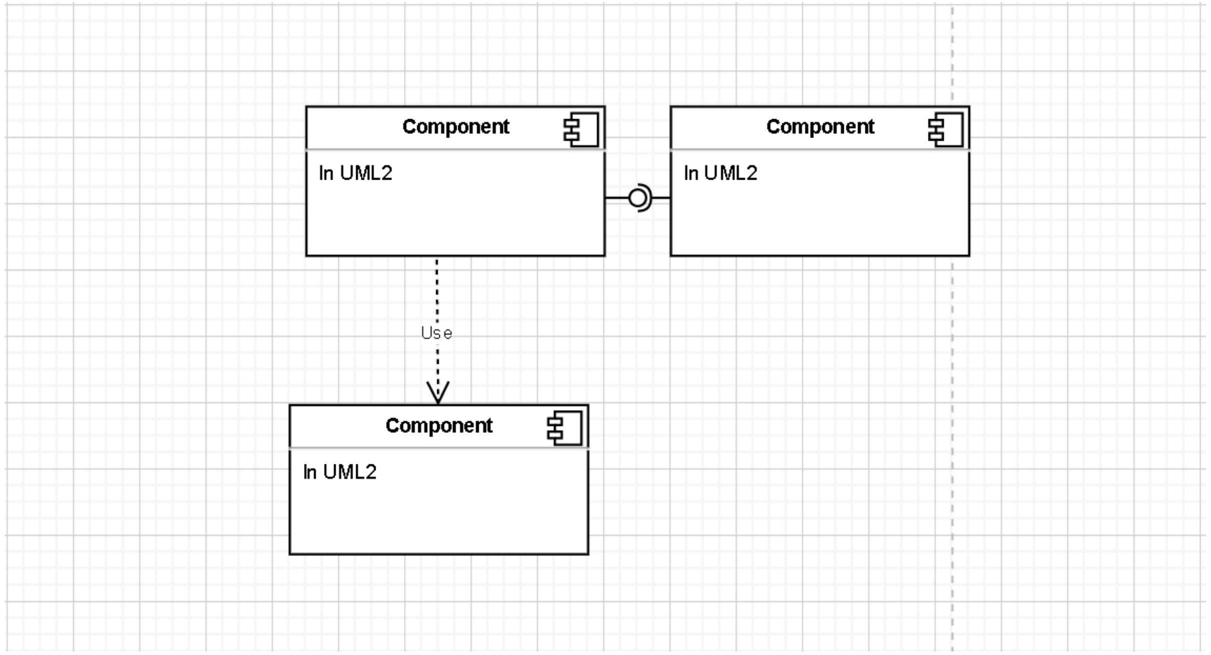
Architectural model: logical view

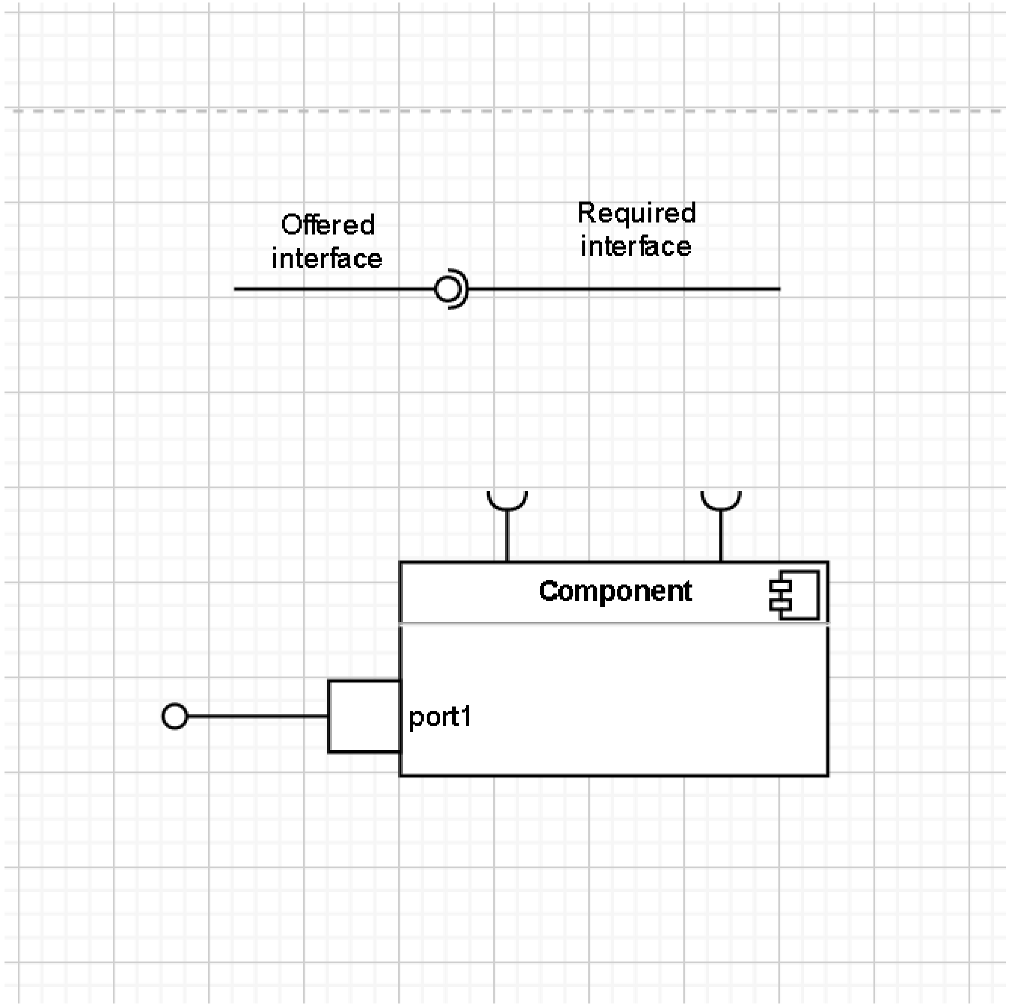
- Information model→Class diagram (entities and relationships)
- Operations→Component diagram
- What is a subsystem and a component?
  - A subsystem is a part of a system.
    - **An integrated set of elements, subsystems or assemblies that accomplish a defined objective**. These elements include products (hardware, software, firmware), processes, people, information, techniques, facilities, services and other support elements. (INCOSE Handbook V4.0, 2015)
    - A subsystem can be divided into different subsystems and components
  - A component is a set of related functionalities offering a coherent interface
    - A component can NOT be divided into more elements

Architectural model: logical view notation

- In UML (1.x, 2.x) we have a **component diagram** as a notation/syntax to produce a logical view of the system.
- A component in a component diagram can represent a subsystem or a component.
    - We can add annotations.

Component diagram

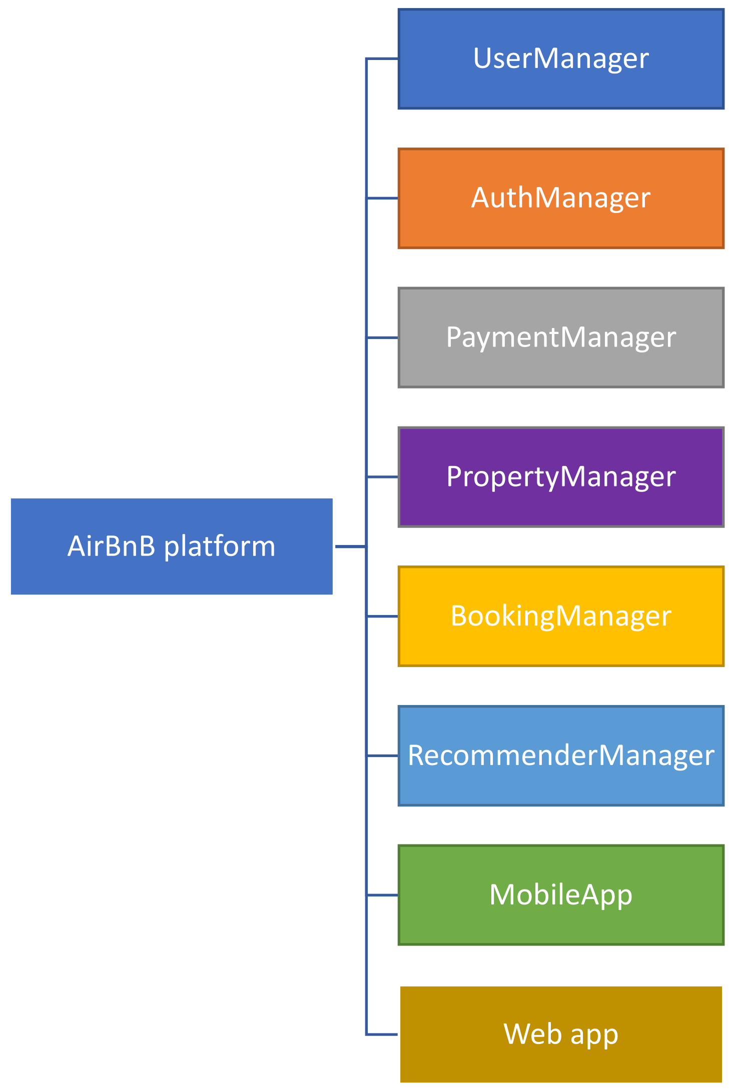




Component diagram: how to start?

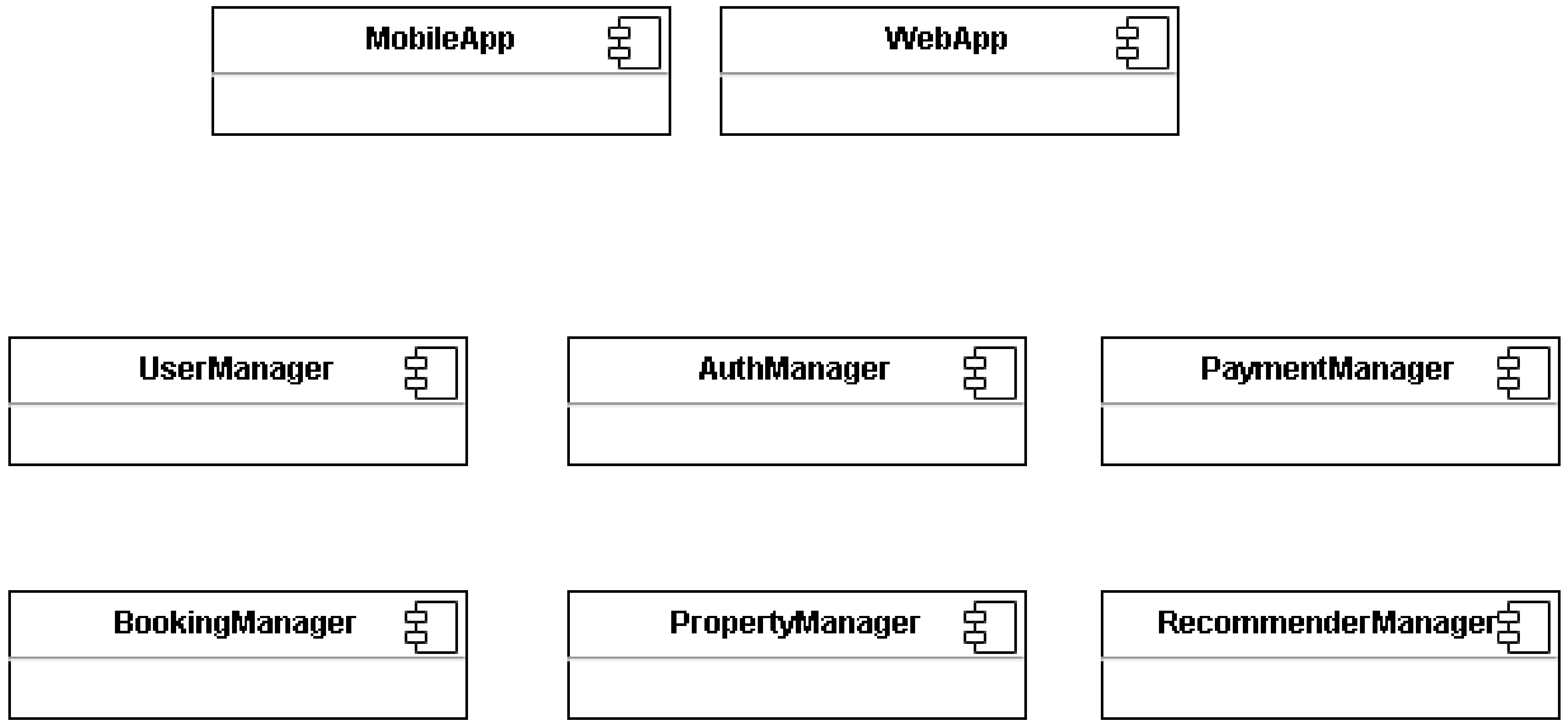
- **Functional decomposition**: grouping functionalities (requirements)
  - We can also use other type of decomposition: use cases, tasks, etc.
- The question:
  - Which are the **major responsibilities** of my system?
  - How they can be connected?
- Approach:
  - We put together things that may change together
- Objectives:
  - Cohesion & Coupling
  - Non-functional aspects: extensibility, simplicity, scalability, etc.

Functional breakdown structure



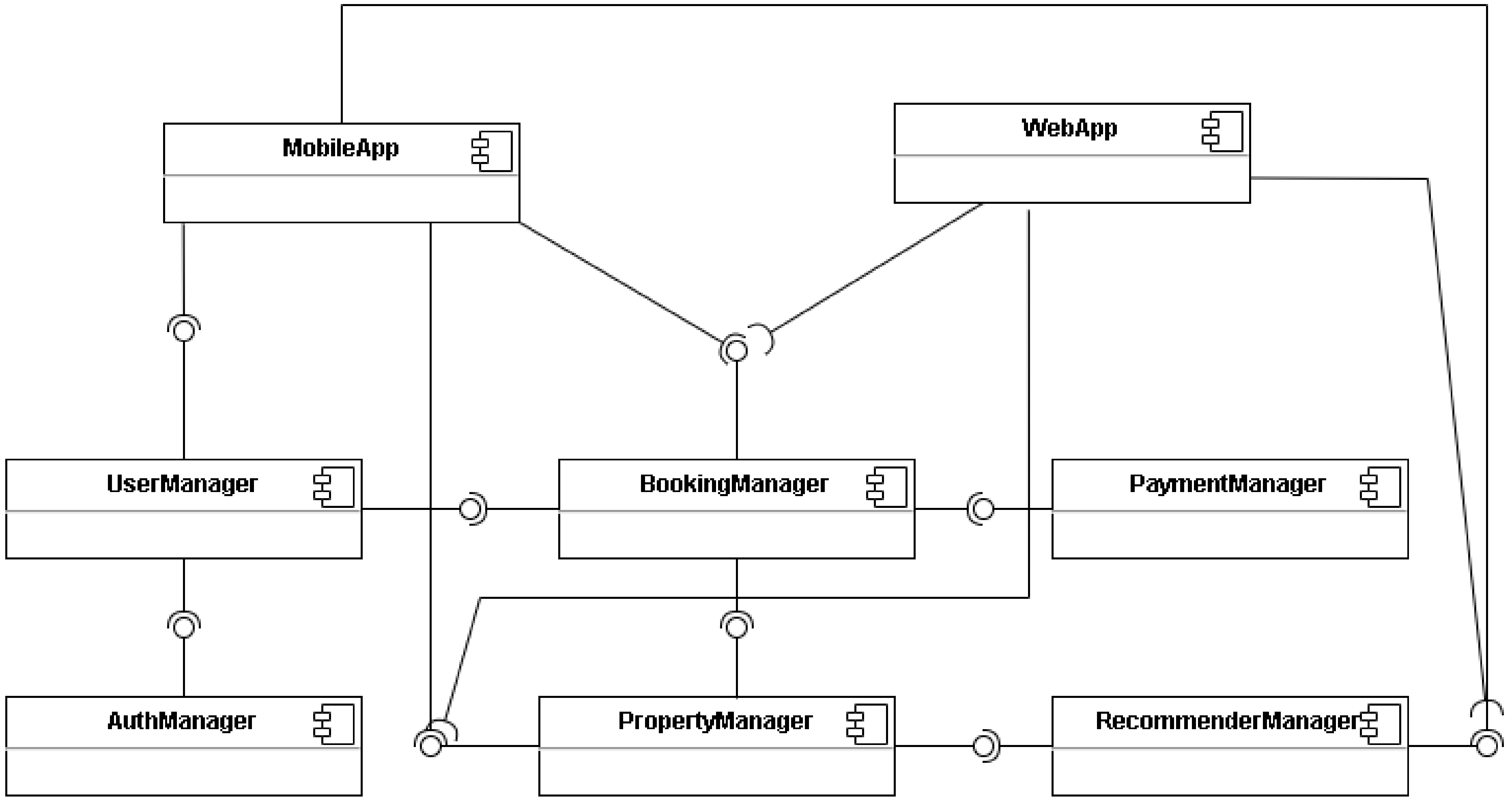
- Let’s draw using our notation→component diagram

Component diagram: components



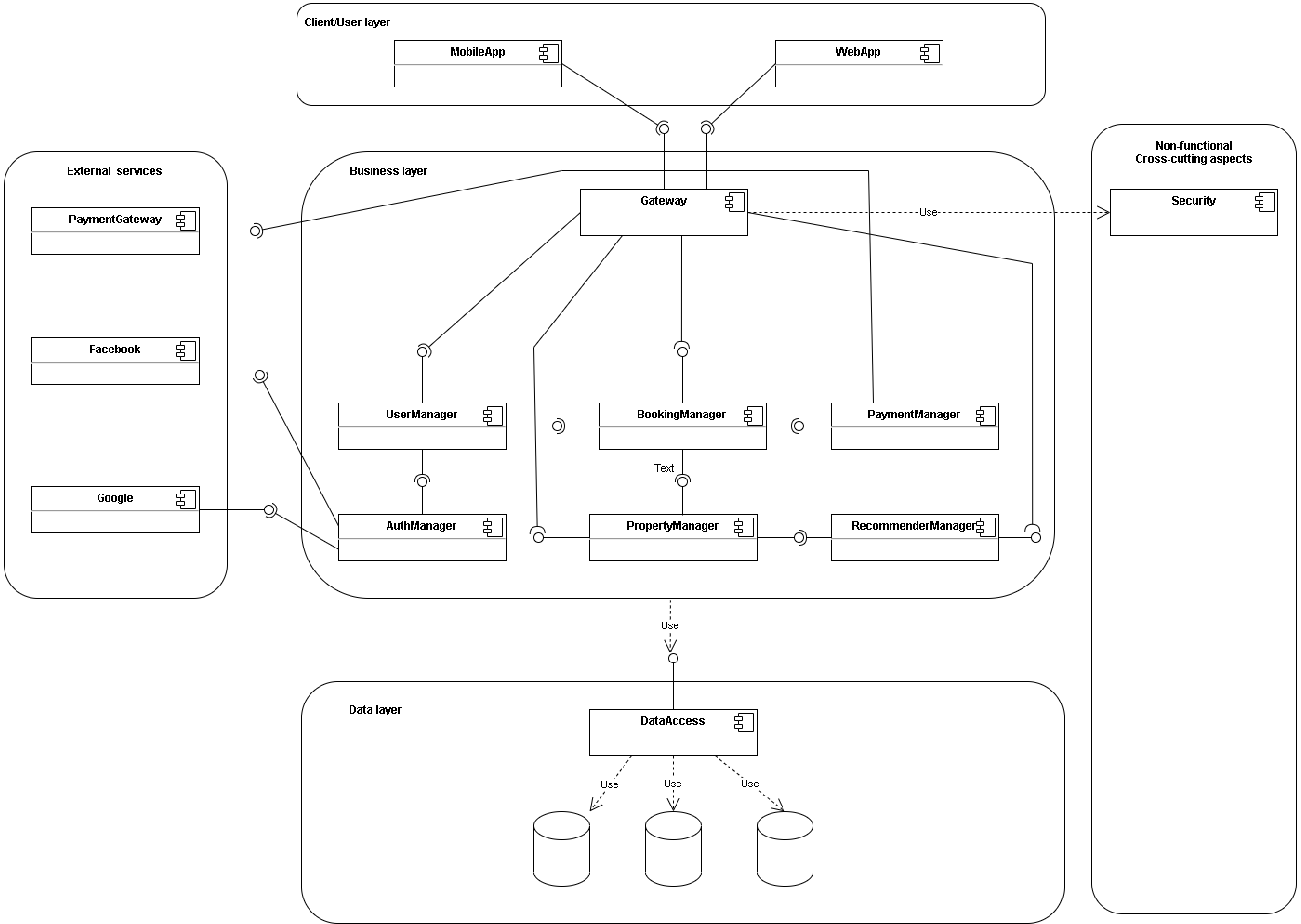
Component diagram: basic connections

- The Webapp also requires the UserManager interface.



Questions?

- How we represent the connection to data sources?
- Where we represent non-functional requirements?
- Where we represent external services?


Client/User layer
MobileApp
WebApp
External services
PaymentGateway
Facebook
Google
Business layer
Gateway
UserManager
BookingManager
PaymentManager
AuthManager
PropertyManager
RecommenderManager
Text
Use
Non-functional
Cross-cutting aspects
Security
Use
Data layer
DataAccess
Use
Use
Use

Mapping requirements to components

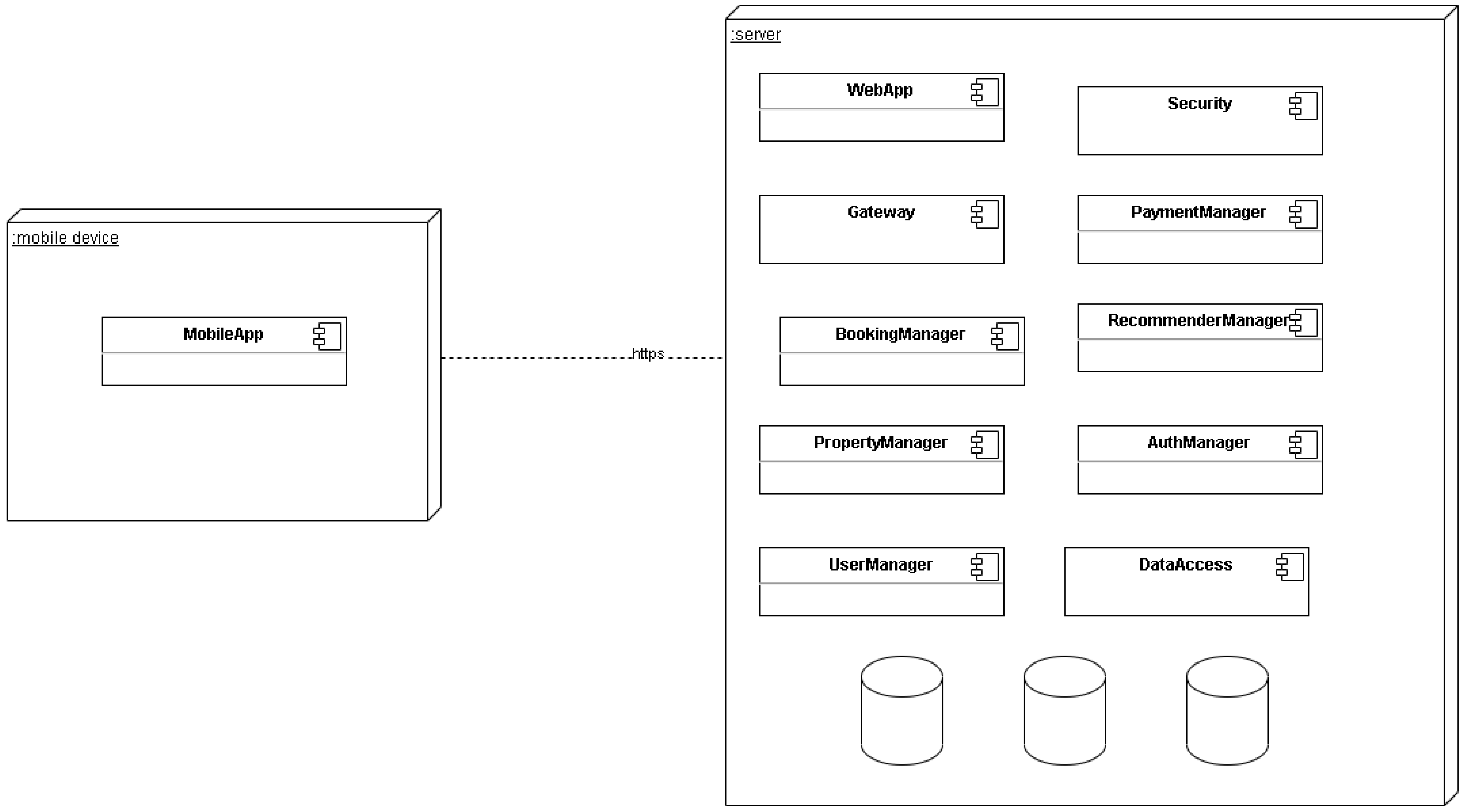
Requirement/ Component	MobileApp	WebApp	UserManager	BookingManager	PaymentManager	AuthManager	PropertyManager	RecommendationManager	Security
FR1	X	X					X	X	
FR2	X	X	X			X			
FR3	X	X	X	X					
FR4				X	X				
…									
NFR1									X
NFR2	X								

- We do not include here the gateway because it is just a dispatcher.
- We do not include here the database because it will be part of almost all requirements that need to govern data.

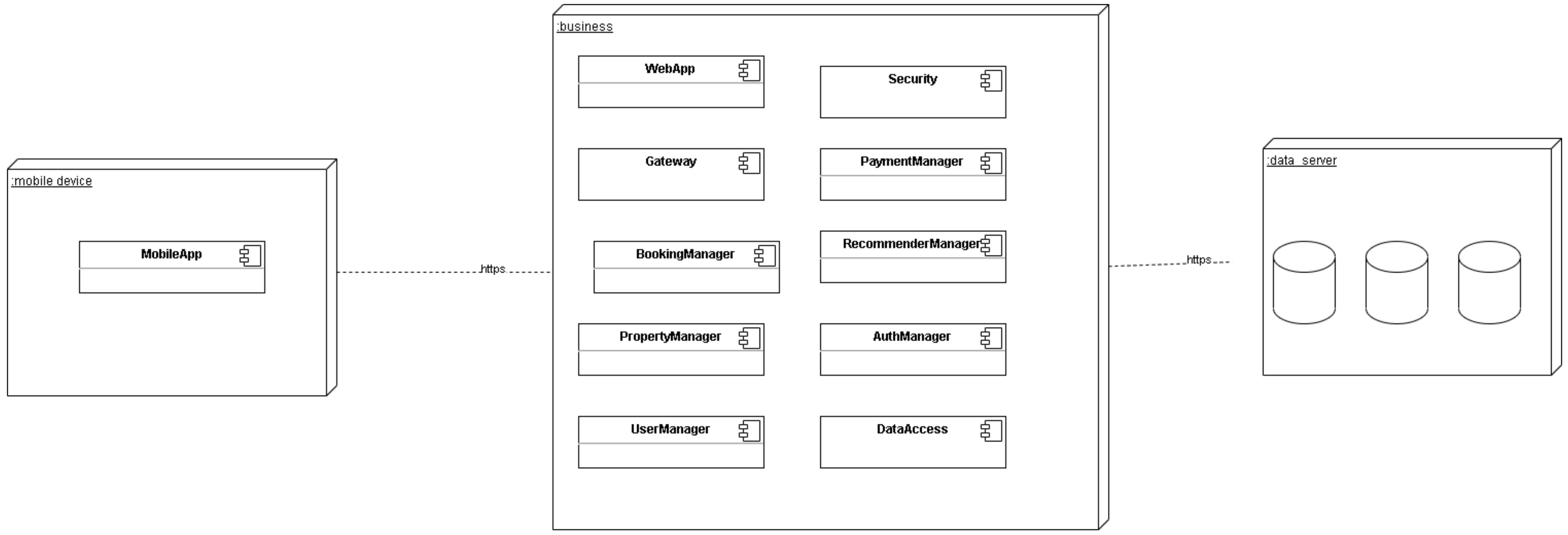
Can we use an architectural pattern?

- The previous one is a pattern: API Gateway
- Other simpler are: Client-Server or MVC
  - However, the internal composition of the server or the controller in MVC will be similar to our business layer.

Architectural model: physical (deployment diagram)



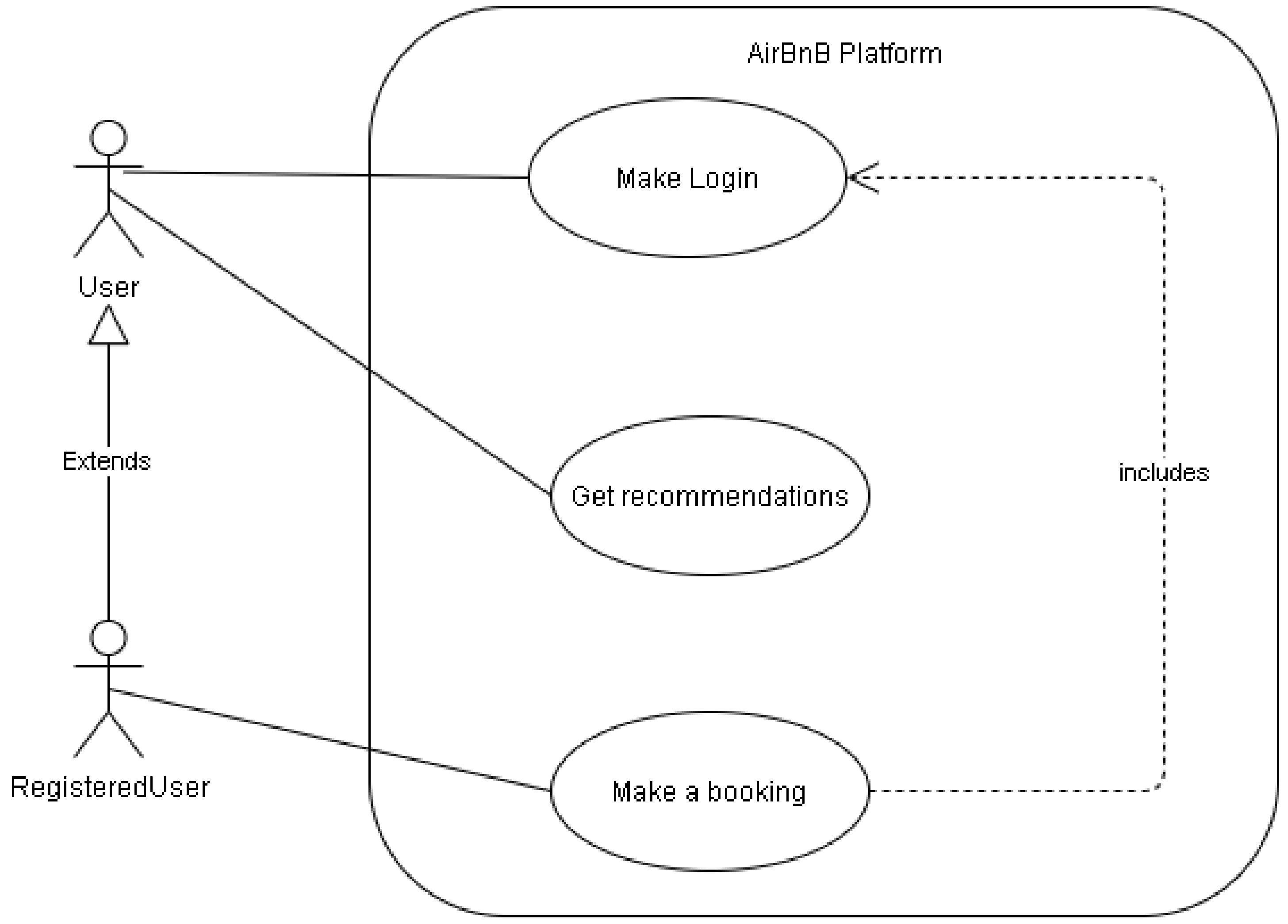
Architectural model: physical (deployment diagram II)



Architectural model: physical (deployment diagram III)

- Other topologies depending on non-functional aspects (cost, scalability, performance, technology stack, etc.)
- We can separate as much as we need.
  - One machine for the mobile app
  - One machine for the webapp
  - One machine for the Gateway
  - One machine for each component
  - One machine for each database
- Initially: 2 machines
- Finally: 14 machines

Architectural model: scenario (use case diagram)

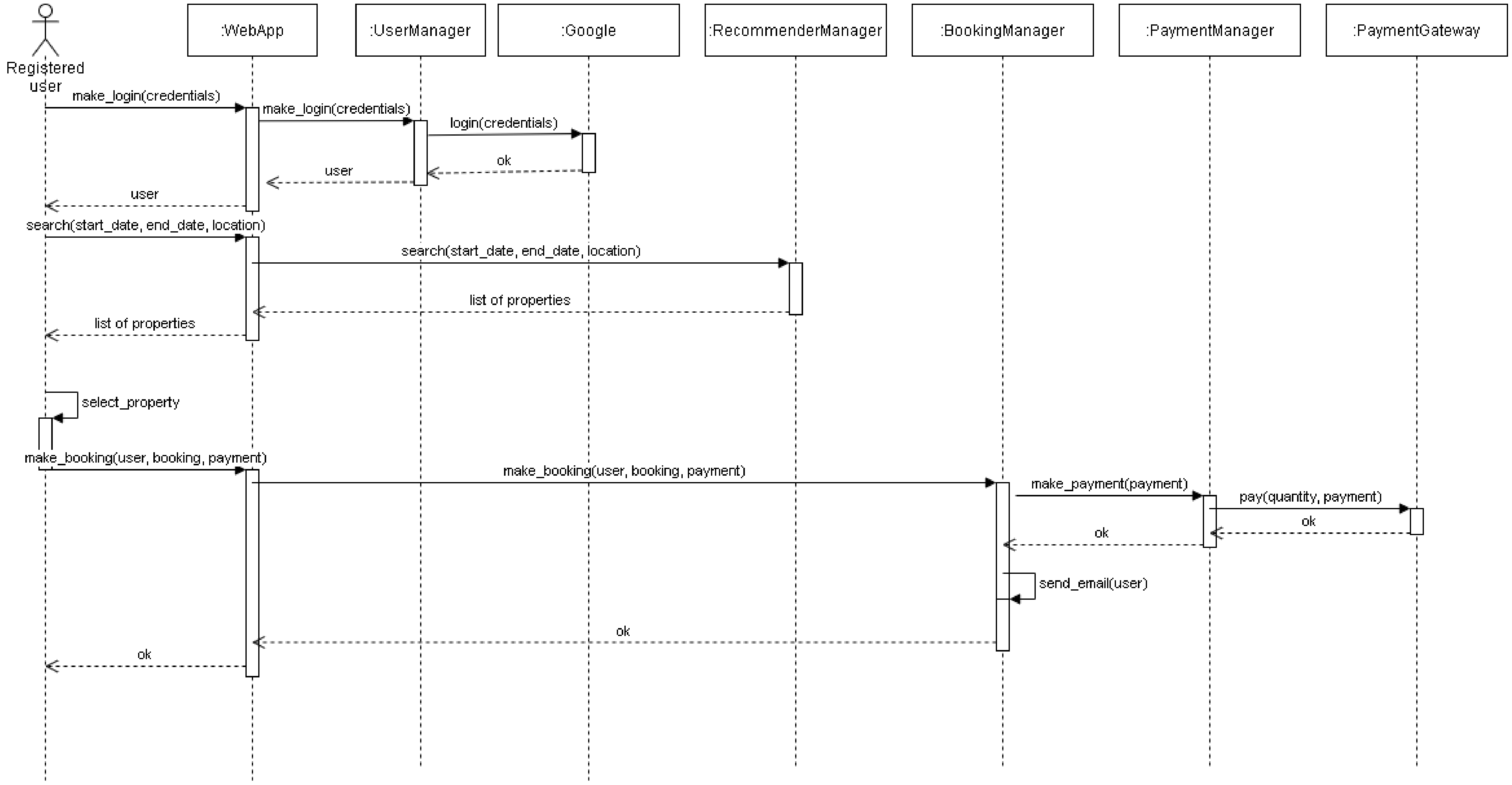


Scenario: make a booking

- Preconditions: the user shall be registered and logged in the system
- Postconditions: a booking is created and an email is sent.
- Basic scenario:
    - A registered user logs into the system
    - The registered user search for places indicating dates and location.
    - The registered user receives a set of recommendations (properties).
    - The registered user selects a property.
    - The registered user enters the payment information.
    - The payment is confirmed.
    - An email with the booking data is sent to the registered user.

Architectural model: process (sequence diagram)

- We use here the notation of a sequence diagram.
- We represent the interactions between components to implement a scenario.
- We omit the “Gateway” to simplify the diagram.


Registered user
:WebApp
:UserManager
:Google
:RecommenderManager
:BookingManager
:PaymentManager
:PaymentGateway
make_login(credentials)
make_login(credentials)
login(credentials)
ok
user
user
search(start_date, end_date, location)
search(start_date, end_date, location)
list of properties
list of properties
select_property
make_booking(user, booking, payment)
make_booking(user, booking, payment)
make_payment(payment)
pay(quantity, payment)
ok
ok
send_email(user)
ok
ok

Comments

- The information that is used as parameter or as a return value are the classes of our model.
- The operations are those offered by the components.

- →We can easily transform this diagram into source code.

Python source code…here we are making decisions on data types, data structures, etc. (solution domain not problem domain)

Conceptual model

```
class User:
   def __init__(self, name, id):
     self.name = name
     self.id = id
class Booking:
                        pass

class Payment:
                        pass

class Property:
                        pass
```

Components

```
class UserManager:
  def __init__(self):
     self.users = []
 def add_user(self, user):
     self.users.append(user)

class BookingManager:
  def __init__(self):
    pass
def make_booking(user, booking, payment):
   result =  self.payment_provider.make_payment(payment)
   if result == “OK”:
     self.send_confirmation(user, booking)
  return result
```