## *Архитектурен проект*

## 

## *Тема:*

## ***Система за управление и резервация на***

## ***зали/кабинети***

## 

## 

## 

## *Група: 77*

## 

## 

## ***Автори:***

## *Алекс Ласкин, фак. Номер: 471219012*

## *Ангелина Дачкинова, фак. Номер: 471219050*

## *Аспарух Дамянов, фак. Номер: 471219059*

## *Десислава Милушева, фак. Номер: 471219007*

## *Николай Томов, фак. Номер: 471219074*

## *Станислав Рангелов , фак. Номер: 471219029*

## *30.10.2021*

Съдържание

[*Въведение* 3](#_Toc86607487)

[*Предназначение* 3](#_Toc86607488)

[*Обхват* 3](#_Toc86607489)

[*Актьори* 3](#_Toc86607490)

[*Архитектурен обзор* 4](#_Toc86607491)

[Use-case изглед 4](#_Toc86607492)

[Логически изглед 6](#_Toc86607493)

[Процесен изглед 6](#_Toc86607494)

[Изглед на данните 7](#_Toc86607495)

[Изглед на внедряването 7](#_Toc86607496)

[Изглед на имплементацията 8](#_Toc86607497)

[*Нефункционални изисквания* 9](#_Toc86607498)

[Достъпност 9](#_Toc86607499)

[Разширяемост 10](#_Toc86607500)

[Производителност 10](#_Toc86607501)

[Сигурност 10](#_Toc86607502)

[Възможност за тестване 11](#_Toc86607503)

[Интероперабилност 11](#_Toc86607504)

[Използваемост 11](#_Toc86607505)

# *Въведение*

Този документ има за цел да представи основните характеристики както и архитектурата на системата за резервиране на стаи. Разглеждат се основните сценарии и изгледи както и нефункционалните изисквания на системата.

# *Предназначение*

## *Обхват*

## 

Проектът има за цел да спомогне организационната дейност в големи организации, като висши учебни заведения, училища, така както и офиси, разполагащи с конферентни зали или зали за осъществяване на срещи. Проектът може да бъде приложен в различни структури, в зависимост от приложимостта и необходимостта на продукта. Ползите от осъществяване на проекта ще бъдат устойчиви. Целта на проекта е да допринесе на съответните организации, използващи продукта по-добра организация в използването на стаи и зали, предназначени за общо ползване.

Документът обхваща втори етап от жизнения цикъл на проекта - “Проектиране на на софтуер”. Създава се детайлен проект на системата и се определя софтуерната архитектура. Софтуерната архитектура представя процеса за изработка на софтуера, при които се избират: технологиите, които ще се използват; стандарт за писане на код; инструменти и платформа. Сложните задачи се разделят на по-прости и лесни за изпълнение. Разделят се компонентите и се описва на тяхната функционалност.

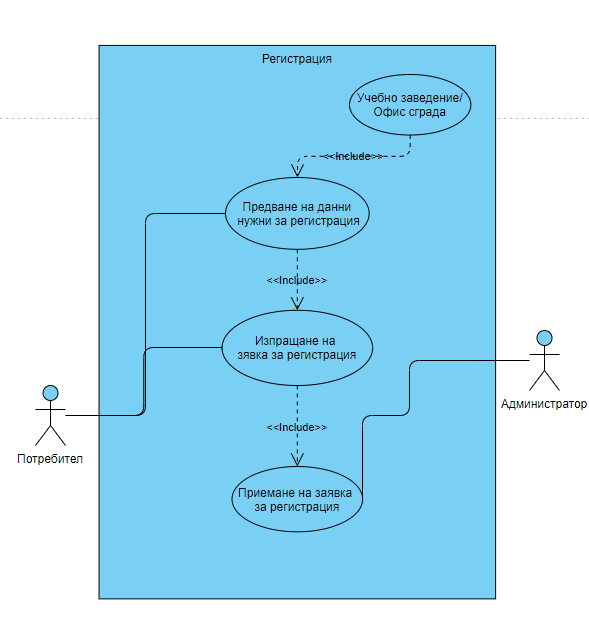
## *Актьори*

Основните участници заинтересовани от документа са Администратор и Поръчител(Възложител). Интересът на поръчителя в този документ е насочен към Нефункционалните изисквания, докато администратора го засяга Архитектурния образ.Възможно е и някои ексклузивен потребител да изяви желание за достъп до документа , но предоставянето на такъв не е желателно.

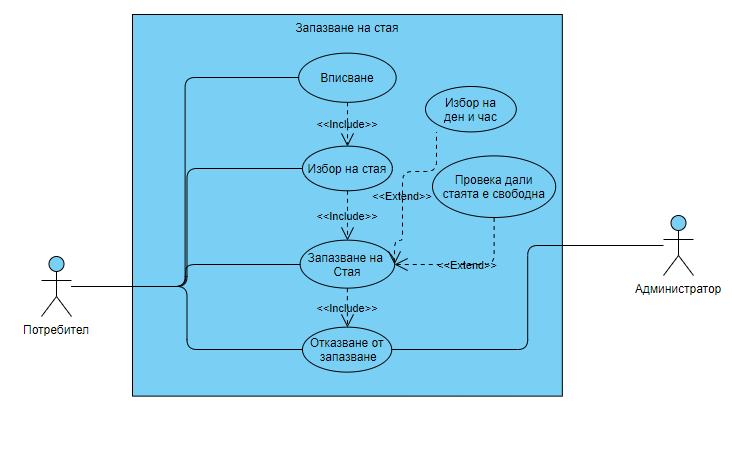
# *Архитектурен обзор*

### 1. Use-case изглед

Чрез use-case изглед се представят основните сценарии за използване на системата от гледна точка на потребителят.

******

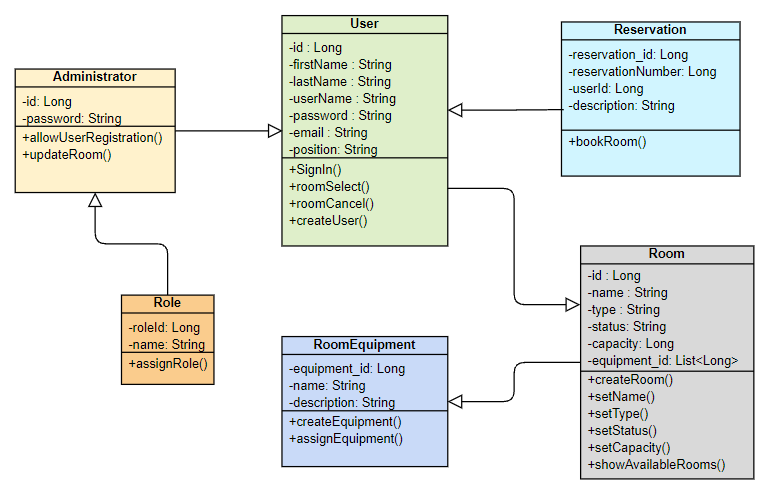
Тази диаграма показва как ще работи функционалността за регистрация на потребител, който ще има право да запазва стаи в дадено учебно заведение/офис сграда.



Тази диаграма показва как ще работи функционалността за запазване на дадена стая. И функционалността да се откажем от дадена стая или при необходимост администратор да ни откажа правото за тази стая в даденият ден и час.

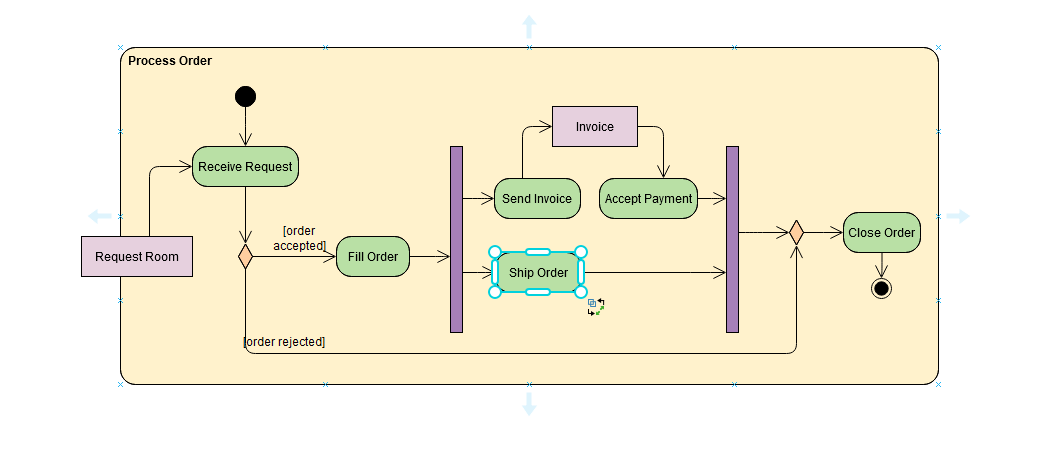
### 2. Логически изглед

Логическия изглед разделя системата на основни компоненти и описва най-важните класове и тяхната организация в пакети и подсистеми. Той е предназначен както за проектантите на системата така и за нейните разработчици.



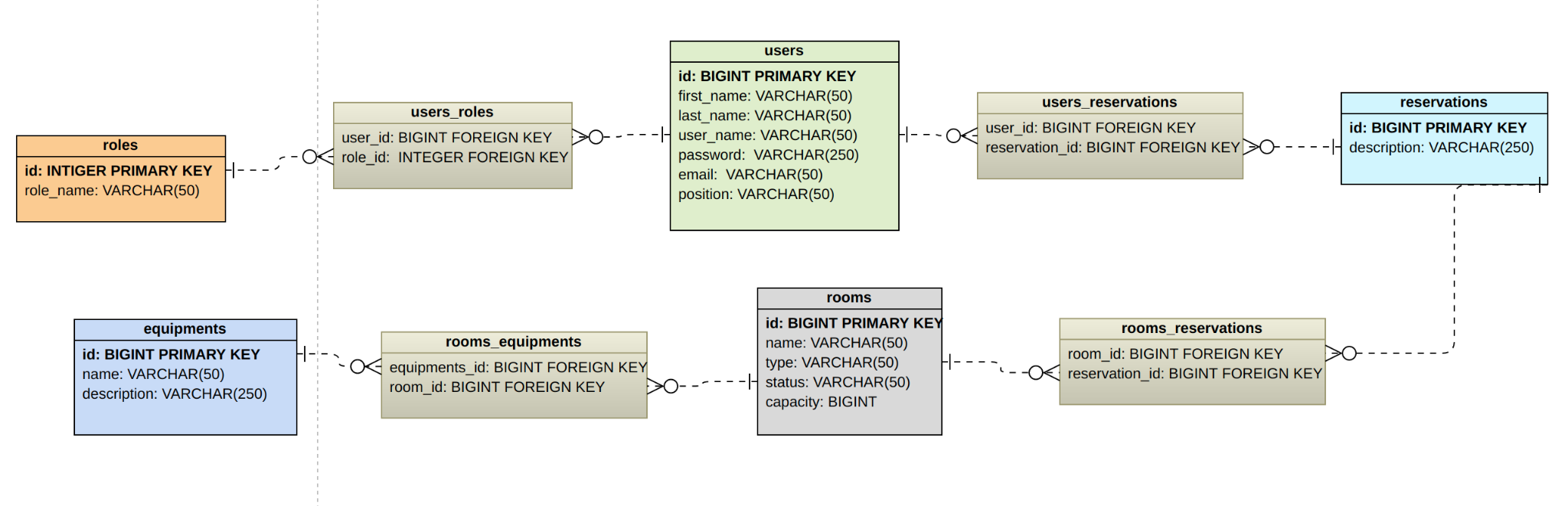
### 3. Процесен изглед

Потребителят отправя заявка за стая.

Заявката се обработва спрямо наличността на стаята. Ако критериите се изпълнят процесът продължава, ако не заявката се затваря.При обработката на заявката клиентът получава сметка, която заплаща и така се финализира резервацията.

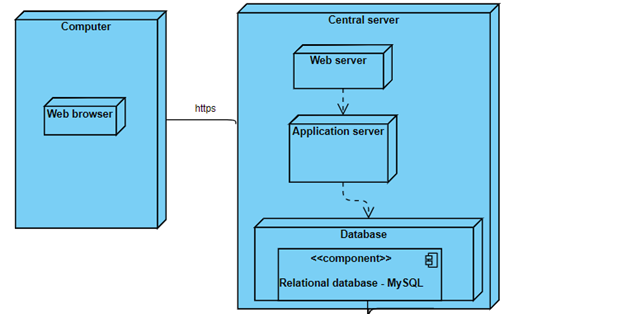
### 4. Изглед на данните

Този изглед представлява модел на обработваните данни, включващ логическите връзки между тях с помощта на графична схема на базата от данни, включваща типовете данни и релации.



### 5. Изглед на внедряването

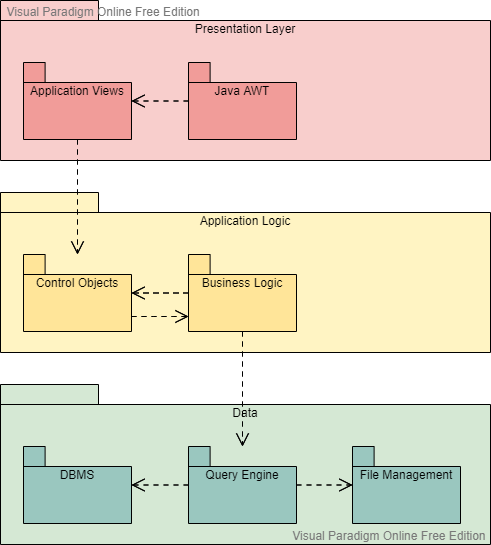
Диаграмата за внедряване на UML е диаграма, която показва конфигурацията на възлите за обработка на времето за изпълнение и компонентите, които живеят върху тях. Диаграмите за внедряване са вид структурна диаграма, използвана при моделиране на физическите аспекти на обектно-ориентирана система. Те често се използват за моделиране на статичният изглед за внедряване на система.

******

С тази диаграма се вижда разпределението на ресурсите в системата, както и комуникацията между тях. Връзката се осъществява онлайн чрез https, което е защитена комуникация.

### 6. Изглед на имплементацията

Изгледа илюстрира системата от гледна точка на програмиста и се занимава с управлението на софтуера а диаграмата показва по опростено класовете в приложението.



На диаграмата за имплементация се показва разделението на 3 слоя спрямо MVC архитектурата. Тя също показва разликата на достъп на функции между администраторите и потребителите. Както и ограничението за промяна на данните за потребители само от администратор.

# *Нефункционални изисквания*

### 1. Достъпност

За анализиране и идентифициране на проблеми довели до спиране на системата могат да бъдат използвани логовете предоставени от Spring Boot и React. За възстановяване на работата на приложението може да бъде рестартиран сървърът на проблемният фрагмент (API сървърът или React сървърът).

### 2. Разширяемост

В бъдещ етап проектът може да има подобрения, без да се нарушават съществуващите системни функции. Към готовия продукт може да бъде имплементиран изглед с помощта на календар за преглед на стаите и резервациите, направени в съответното време и възможност за резервация чрез него. Може да бъде добавена опция за разплащателен метод, ако наемането на стаята изисква заплащане. Също така могат да бъдат добавени допълнителни функционалности, спрямо желанието на клиента. За целта може да бъде използван механизмът White-Box. Извършва се при условие, че е наличен source (изходния) код на системата с или без неговата модификация.

### 

### 3. Производителност

Използването на MVC архитектурата дава значителни плюсове на производителността на проекта. Чрез нея проблемите със latency и security са сравнително по-малко в сравнение с microservices.Изисква се по-малък ресурс за поддръжка,development и deployment.Друг значим фактор е процесът на разработка - MVC поддържа бързо и паралелно развитие. Примерен сценарий за което е :

Когато MVC моделът се използва за разработване на някое конкретно уеб приложение,е възможно един програмист да работи върху изгледа, докато другият може да работи върху контролера, за да създаде бизнес логиката на уеб приложението.Това допринася и за лесното прилагане на компоненти като рамки, шаблони или скриптове.Както и улеснено тестване.

### 4. Сигурност

Засилване на сигурността на определени компоненти, би било възможно, когато това е необходимо. Самите данни обвързани с потребителската лична информация биха били по-критични и съответно ролята на сигурността за компонентите опериращи с тези данни ще бъде по-голяма.

### 5. Възможност за тестване

Когато разделим кода на тези 3 компонента (модел, изгледи и контролери), моделът ще съдържа цялата бизнес логика. Огромен процент от вашата кода ще бъде в компонента на модела. Чрез мокване на инфраструктурния код (като комуникация с базата или извършване на други HTTP извиквания и т.н.), този компонент може да бъде напълно единично тестван.

Наличието на потребителски интерфейс би направило тестването на модули доста трудно, ако не и невъзможно. Също така тестването на аспектите на уеб сървъра на приложението, чрез модулно тестване също е малко трудно. Чрез разделянето на този код, в изгледи и контролери, ще направи модела лесен за тестване на единици код.

### 6. Интероперабилност

Данните ще трябва единствено да се изпращат между отделните модули на системата така че тази връзка трябва да се използва само от нея. Каналите ще се използват за автентикация на потребителите и изпращане и получаване на данни за стаите на различните организации.

### 7. Използваемост

Необходим е прост интерфейс за да може потребителите бързо и лесно да открият и запазят някоя стая също така и да не запазят такава която не е съобразена с нуждите им. При такава грешка ще се наруши целта на приложението която е по-лесно и организирано откриване и предварително запазване на множеството офиси и кабинети в някоя организация. Интерфейса ще се чете лесно за да могат потребителите бързо да се научат да използват приложението ефикасно. Използваемостта е важна да бъде по добра от на вече установените методи за изграждане на графици и планове които този проект иска да замести.