

**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – ВАРНА**

**ФАКУЛТЕТ ПО ИЗЧИСЛИТЕЛНА ТЕХНИКА И АВТОМАТИЗАЦИЯ**

**КАТЕДРА „СОФТУЕРНИ И ИНТЕРНЕТ ТЕХНОЛОГИИ“**

**Специалност „Софтуерни и интернет технологии“**

**Проект по Обектно – ориентирано програмиране част 2**

**Тема 2**

**Хотелска информационна система**

**Изготвена от :**

Зерин Мехмедова и Йордан Симеонов

(СИТ, гр. 1А,4А, Фак. №: 22621632,22621710)

**2025г.**

Съдържание

[Условие на проекта 2](#_Toc187081022)

[Анализ на проблема 3](#_Toc187081023)

[Работа с потребители 4](#_Toc187081024)

[Навигация 4](#_Toc187081025)

[Управление на системата според потребител 5](#_Toc187081026)

[Администратор 5](#_Toc187081027)

[Мениджър 9](#_Toc187081028)

[Собственик 13](#_Toc187081029)

[Рецепционист 18](#_Toc187081030)

[Структура на проекта 22](#_Toc187081031)

[Проектиране на системата 25](#_Toc187081032)

[База от данни 25](#_Toc187081033)

[Модел на Чен 31](#_Toc187081034)

[Use-case Диаграма 32](#_Toc187081035)

[Class Диаграми 32](#_Toc187081036)

[Entity & Repository 32](#_Toc187081037)

[Converters 33](#_Toc187081038)

[Modules 33](#_Toc187081039)

[Controllers 34](#_Toc187081040)

# Условие на проекта

II. Хотел

Да се разработи информационна система – Хотел. Програмата съхранява и обработва данни за хотелски услуги (резервации и допълнителни услуги). Системата позволява множествен достъп.

Системата поддържа два вида потребители администратор и клиенти (рецепционист, мениджър, собственик) с различни роли за достъп до функционалностите в системата.

Операции за работа с потребители:

* Създаване на собственици на хотел(и) от администратор;
* Създаване на хотел с мениджър от собственик
* Създаване на рецепционисти от мениджър

Системата поддържа операции за работа с резервации:

* Създаване на клиенти;
* Създаване на резервация от рецепционист (Номер на резервация, Тип на резервация, Тип на прекратяване на резервацията, категории стаи, ...);
* Създаване и предоставяне на допълнителни услуги, съобразени със сезона (отчетност на тип услуга и брой ползвания);
* Рейтинговане на клиенти.

Системата поддържа справки по произволен период за:

* Категория клиенти:  
  o Информация за клиенти (лични данни);

o Използване на хотел и хотелски услуги;  
  
o Рейтинг на клиенти.

* Рецепционисти (създадени резервации, данните на рецепциониста)
* Създадени регистрации (дата, статус, хотел, съдържание на формуляра);
* Стаи (Рейтинг на стаите за ползваемост);

Мениджър на хотел достъпва справки само за хотел, за който е отговорен.   
Собственика достъпва справки за всички притежаващи хотели. Рецепциониста има право на справки за заетостта на стаите.

Системата поддържа Известия за събития:

* Изтичаща резервация;
* Известия за рисков клиент (при създаване на нова резервация).

# Анализ на проблема

Системата ще работи с потребители на четири различни нива на достъп-администратори,собственици,мениджъри и рецепционисти като всяка една от тях ще представлява уникални нива на достъп и права.

Потребител с роля „Администратор“ има достъп до следните права в системата:

* Добавяне на нов собственик в системата
* Търсене на съществуващи собственици по име и имейл
* Проверка на цялата информация за даден собственик
* Търсене на съществуващи мениджъри по име,имейл и телефон
* Справка за всички хотели в системата
* Търсене на хотел по име,локация и брой звезди
* Добавяне на нова услуга
* Търсене на съществуващи услуги по име и цена

Потребител с роля „Мениджър“ има достъп до следните права в системата:

* Добавяне на нов рецепционист в системата
* Справка за хотелите, за които отговаря всеки един мениджър
* Търсене на хотели по име ,локация и брой звезди
* Проверка на цялата информация за даден хотел
* Добавяне на рецепционисти за хотелите,в които работи определения мениджър
* Справка за използваемостта на стаите за определен период от време в хотел по наш избор
* Справка за определен период от време,в което са направени резервации за определен хотел
* Справка за използваемостта на услугите на определен хотел за период от време

Потребител с роля „Собственик“ има достъп до следните права в системата:

* Справка със всички хотели,за които отговаря всеки собственик
* Търсене на хотели по име,локация и звезди
* Проверка на цялата информация за избран хотел
* Добавяне на мениджъри към хотел
* Премахване на мениджъри от хотел
* Добавяне на услуги към хотел
* Премахване на услуги от определен хотел
* Добавяне на стая към избран от нас хотел
* Визуализация на стаите в определен хотел
* Добавяне на нов мениджър
* Добавяне на нов хотел
* Справка за използваемостта на стаите за определен период от време в хотел по наш избор
* Справка за определен период от време,в които са направени резервации за определен хотел
* Справка за използваемостта на услугите на определен хотел за период от време

Потребител с роля „Рецепционист“ има достъп до следните права в системата:

* Добавяне на клиент
* Създаване на резервация за определен хотел,в които работи рецепционистта
* Списък с клиенти
* Изобразяване на допълнително информация за клиентите в системата
* Редактиране на информацията на клиентите
* Справка за използването на стаите за определен период от време
* Справка за резервациите за хотела,в които работи рецепциониста
* Справка за направерените резервации
* Показване на информацията за резервации и дали тя е одобрена или не
* Търсене на резервации
* Търсене на клиенти

# Работа с потребители

A screenshot of a bedroom

Description automatically generated

Фиг.1 -Начален екран

При опит за вход в системата,както е показано на Фиг.1 се извършва проверка за коректност на входните данни.Ако те не съвпадат с тези в базата от данни,на екрана се показва съобщение за отказан достъп.При успешен достъп в системата на потребителя му се позволява да използва системата.

# Навигация

След успешно влизане в системата,програмата следи като какъв потребител сме се вписали и я изписва в горния ляв ъгъл на екрана. Потребителят е пренасочен към главния панел на системата. В лявата му част са посочени бутони,чрез които можем да достъпим определени места от системата. В средата имаме поле за нотификации,които ще ни следи какво правим в системата-дали добавяме или актуализираме данни. Най-отдолу се намира бутонът за излизане от системата.

# Управление на системата според потребител

Всеки потребител си има свое уникално меню,което само той може да достъпва.В менюто са описани всичките му права на достъп в системата.

## Администратор

A screenshot of a computer

Description automatically generated

Фиг.2 – Прозорец на потребител с роля „Администратор“

В средата на екрана на Фиг.2 са визуализирани известията на системата-при първият ни опит да се впишем потребителското ни име беше сгрешено,а при втория паролата. При трети успешен опит влизаме в системата.

* Първият ни избор в системата са собствениците:

A screenshot of a computer

Description automatically generated

Фиг.3. – Собственици

При натискане на бутона собственици отиваме в друга сцена,където са изобразени всички възможни собственици в системата.Те са изобразени в таблица с първо име,фамилия и имейл. Освен това имаме и търсачка,чрез която можем да намерим определен собственик в системата и бутон за изчистване,ако случайно сбъркаме и искаме да изтрием всичко.

В долния десен ъгъл имаме възможност за връщане с едан стъпа назад към главното меню на администратора.

В горния десен ъгъл имаме възможност за добавяне на нов собственик.

A screenshot of a computer

Description automatically generated

Фиг.4 – Създаване на нов собственик

След натискане на бутона имаме форма за създаване на собственици на екрана. Можем да въведем две имена,потребителско име,с което ще влиза в системата, парола и потвърждение на паролата,имейл и телефонен номер. Всяко едно поле си има своя собствена валидация в системата и ако поне едно е сгрешено или не е в определения диапазон от системата, при натискане на бутона за добавяне излиза въобщение за грешка на екрана и нямаме право да създадаваме,докато данните не са правилни.

При успешно въвеждане на данни в системата се изкарва съобщение на екрана за успешно създаден собственик. След това,чрез бутона за връщане назад можем да се върнем във второстепенното меню на таблицата със собственици.

* Втория ни избор в системата са мениджърите

A screenshot of a computer

Description automatically generated

Фиг.5 – Мениджъри

При натискане на бутона мениджъри системата ни води до втора сцена на екрана,където имаме таблица с всички мениджъри в системата. Те са изобразени чрез уникалният си ключ,две имена,имейл и телефонни номера. Отново имаме търсачка,която търси мениджърите по определените им данни на екранна. Бутона „Х“ изтрива всичко написано в полето за търсене.

В долния десен ъгъл имаме бутон,който ни отвежда към главното меню.

* Третия ни избор в системата са хотелите

A screenshot of a computer

Description automatically generated

Фиг.6 – Хотели

При натискане на бутона хотели системата ни изкарва следващата сцена ,където се намира таблица с най-важните данни за един хотел- име,локация и брой звезди. Имаме поле,в което можем да търсим хотел по наше желание,според неговото име,локация и брои звезди. До полето се намира и бутон за изчистване на данните,въведени в него.

В долния десен ъгъл имаме бутон за връщане към главното меню.

* Четвъртия ни избор в системата услугите

A screenshot of a computer

Description automatically generated

Фиг.7 – Хотелски услуги

След натискане на бутона услуги.на екрана се изобразява таблица с имената и цените на всяка една услуга,която може да притежава един хотел. Всяка една услуга може да бъде търсена по име и цена в определеното за нея поле за търсене. Също така с бутона „Х“ ,то ще може да бъде и изтрито.

В горния десен ъгъл имаме бутон,чрез който можем да добавяме нова хотелска услуга в системата.

A screenshot of a computer

Description automatically generated

Фиг.8 – Форма за добавяне на хотелска услуга

Формата за добавяне се състои от име и цена. При успешно добавяне след натискане на бутона „Add” излиза съобщение на екрана за успешно създаване,но ако поне едно от двете полете не изпълнява изискванията на системата ,се изкарва съобщение,че няма как да се създаде такава хотелска услуга.

## Мениджър

A screenshot of a computer

Description automatically generated

Фиг.9-Прозорец на потребител с роля „Мениджър“

Имаме няколко избора за работа като мениджъри в системата – добавяне на рецепционисти за хотели,за които отговаряме, хотели,справки за заетостта на стаите и резервациите.

* Първия ни избор в системата е добавянето на рецепционист

A screenshot of a computer screen

Description automatically generated

Фиг.10 – Форма за добавяне на рецепционист

След натискане на бутона имаме форма за създаване на рецепционисти на екрана. Можем да въведем две имена,потребителско име,с което ще влиза в системата, парола и потвърждение на паролата,имейл и телефонен номер. Всяко едно поле си има своя собствена валидация в системата и ако поне едно е сгрешено или не е в определения диапазон от системата, при натискане на бутона за добавяне излиза въобщение за грешка на екрана и нямаме право да създадаваме докато данните не са правилни.

При успешно въвеждане на данни в системата се изкарва съобщение на екрана за успешно създаден рецепционист. След това,чрез бутона за връщане назад „Back” можем да се върнем във главното меню на мениджъра.

* Втория ни избор в системата са хотелите

A screenshot of a computer

Description automatically generated

Фиг.11 – Таблица с хотели

След натискане на бутона за хотелите,за които отговаря мениджъра,се визуализират всички хотели. Можем да ги търсим спрямо име,локация и звезди в определеното поле за търсене и също така при грешка да можем и да изтриваме от това поле. При двойно натискане с мишката на определен хотел излиза допълнителна информация за натиснатия хотел.

A screenshot of a computer

Description automatically generated

Фиг.11 – Информация за хотел

Освен информация за натиснатия хотел,чрез падащо меню можем да си избираме кои рецепционисти да добавяме или премахнем от хотела.

* Третия ни избор в системата е справка за заетостта на стаите

A screenshot of a computer

Description automatically generated

Фиг.12 – Форма за заетостта

При натискане на бутона за справка,можем да изберем начална дата,крайна дата и хотела ,за който искаме да направим справка. При неуспешно въведени дати излиза съобщение за грешка ,за да сложим правилни данни.

A screenshot of a computer

Description automatically generated

Фиг.13- Справка за заетостта

На фигура 13 виждаме справка със заетите стаи за определен период от време за избран от нас хотел.

На екрана се изобразява таблица с данни за номера на стаята,цената,която трябва да се заплати,вида на стаята и колко пъти е използвана.

Имаме и търсачка ако искаме да намерим определена от нас стая,а също така и бутон за изтриване на полето за търсене.

* Четвъртия ни избор в системата е справка за резервации

Отново ползваме същата форма както и при справката за заетостта на стаите с избор на начална дата,крайна дата и възможност за избор на хотел.При натискане на бутона „Get” получаваме исканата от нас справка.

A screenshot of a computer

Description automatically generated

Фиг.14 – Спраква за създадени резервации

При успех на фигура 14 е показан резултата – получаваме таблица с името на рецепционистта,който е създал резервацията и уникалният и ключ в базата от данни. Имаме и търсачка ако искаме да намерим определена от нас резервация,а също така и бутон за изтриване на полето за търсене.

* Петия ни избор в системата е справка за регистрациите

Процедурата е същата като при последните две справки, но при натискане за вземане на справката резултата е различен.

A screenshot of a computer

Description automatically generated

Фиг.15 – Справка за регистрации

При успешно взимане на всички възможни регистрации в зададения от нас период,те се визуализират в таблицата показана на фирура 15.

В таблицата те могат да бъдат търсени по уникалния ключ на резервацията, цена,вид,начална и крайна дата,номер на стая,име на клиент и дали тяхната резервация е приета или отказана.

## Собственик

A screenshot of a computer

Description automatically generated

Фиг.16 – Меню на собственик

Имаме няколко избора за работа като собственик в системата – право да виждаме информацията на хотелите,техните детайли,да добавяме мениджъри,също така имаме справки за резервациите,заетостта на стаите,използваемостта на услугите и регистрациите в системата.

* Първия ни избор е информация за хотела/ите

A screenshot of a computer

Description automatically generated

Фиг.17- Информация за хотел

След натискане на първия бутон,на екрана се изобразява таблица с данните на хотелите,търсачка спрямо име,локация,звезди и бутон за изтриване на информацията в търсачката.

A screenshot of a computer

Description automatically generated

Фиг.18-Допълнителна информация за хотел

При двойно натискане на искан от наш хотел се отваря втори прозорец,където виждаме информацията за хотела,услугите които предлага и мениджърите на хотела.При натискане на мениджърите или хотелските услуги можем да добавяме и премахваме мениджъри,както така и услуги.

* Втория ни избор е детайли за хотел

A screenshot of a computer

Description automatically generated

Фиг.19 – Детайли за хотел

Логиката е същата както и при фигура 17,но при двойно кликане с мишката върху избран от наш хотел,можем да видим неговите детайли,включващи съществуващите вече стаи и форма за въвеждане на нова стая.

При липса на данни,в което и да е поле ще се хвърли грешка,а при успешно въвеждане при натискане на бутона за добавяне,ще получим съобщение на екрана за одобрено добавяне на стая.

* Третия ни избор е добавяне на мениджър

A screenshot of a computer

Description automatically generated

Фиг.20 -Форма за създаване на мениджъри

След натискане на бутона имаме форма за създаване на мениджъри на екрана. Можем да въведем две имена,потребителско име,с което ще влиза в системата, парола и потвърждение на паролата,имейл и телефонен номер.При грешно въведени данни и при натискане на бутона за добавяне излиза въобщение за грешка на екрана и нямаме право да създадаваме,докато данните не са правилни.

* Четвъртия ни избор е справка за резервациите и кои ги е създал

A screenshot of a computer

Description automatically generated

Фиг.21-Форма за период от време

Тази форма е уникална и за останалите три справки,които вижда всеки един собственик в системата. Въвеждаме начална и крайна дата,за която искаме да искараме справка и определения хотел,за които искаме да видим информацията.

A screenshot of a computer

Description automatically generated

Фиг.22 – Резултат от справка

Справката се състои от уникалния ключ на резервацията и рецепциониста ,които я е направил. Имаме и търсачка,по която можем да търсим имената и клювочете на резервацията.

* Петия ни избор е справка за услугите

При натискане на екрана отново ще видим фигура 21.

A screenshot of a computer

Description automatically generated

Фиг.23-Резултат от справка

На екрана се изобразява резултата от справката като таблица,която се състои от име на хотелската услуга,стойността и приходите от нея.Имаме търсачка,в която можем да търсим информация в справката по три отделни критерия,а ако направим грешка да изтрием цялото поле с бутон „Х“.

* Шестия ни избор е справка за заетостта на стаите

Отново започва с фигура 21,но този път е за заетостта на стаите за период от време.

A screenshot of a computer

Description automatically generated

Фиг.24-Резултат от справка

Като резултат получаваме таблица с номера на стаята,нейната цена,типа на стаята и колок пъти е била използвана.Отново имаме търсачка и бутон за чистене.

* Седмия ни избор е спраква зa резервациите в системата

Фигура 21 е използвана за последен път в последната справка за резервациите,където се въвежат отново начална и крайна дата и хотел.

A screenshot of a computer

Description automatically generated

Фиг.25 - Резултат от справка

На екрана се визуализира таблица с информация за регистрацията-уникален ключ,цена,тип на резервацията,дата за начало и край,номер на стая,име на клиент и статус на резервацията.Имаме търсачка и бутон за триене на информацията в нейното поле.

## Рецепционист

A screenshot of a computer

Description automatically generated

Фиг.26 – Меню на рецепционист

Като рецепционист/работник в системата имаме правото да създаваме клиенти,да виждаме списък с всичките клиенти от системата, да създаваме резервации.Освен това имаме право на справки за резервациите,регистрациите и използваемостта на стаите,но всичко това само ,за хотелите в които работи определения потребител.

* Първия ни избор е създаване на клиент

A screenshot of a computer

Description automatically generated

Фиг.27 – Форма за създаване на клиент

Формата се състои от полета за две имена,телефонен номер,ЕГН, адрес,имейл,рожденна дата,дата на издаване, дата на изтичане и поле за мястото,където са издадени.При правилно въвеждане на клиент на екрана се изкарва съобщение за успешно създаден клиент.

* Втория ни избор е създаване не резервация

A screenshot of a computer

Description automatically generated

Фиг.28 – Форма за създаване на резервация

Визуализира се таблица с хотелите, в които работи логнатия в системата потребител,с роля рецепционист.При двойно кликане на избрания от нас хотел, се отваря сцена с форма за създаване на резервация. Тя се състои от вид на резервацията,начална дата,крайна дата, клиент,който е създал резервацията,номер на стая и услугите,които ще предлага резервацията. При искане за създаване на резервация от клиент,който е маркиран като не толкова добър клиент за хотела,се изкарва предупреждение на екрана.

* Третия ни избор е списък с клиенти

A screenshot of a computer

Description automatically generated

Фиг.29 – Таблица с клиенти

Всеки един клиент може да бъде намерен в таблицата по имена,имейл,номер,адрес,ЕГН и рейтинга му. При двойно натискане с мишката на определен клиент се отваря нов прозорец с всичките данни на клиента и имаме право да променяме неговия рейтинг от следните възможности – normal,bad,concerning.

* Четвъртия ни избор е справка за резервациите

A screenshot of a computer

Description automatically generated

Фиг.30 – Справка на всички успешни резервации

Формата от фигура 21 излиза на екрана при натискане на бутона за справка.След това се отваря резултата от фиг.30.

Изобразяват се всички потвърдени резервации в системата за определен период от време,който могат да бъдат търсени по техния уникален ключ,цена,тип на резервация,дати на пристигане и напускане,номер на стая ,име на клиент и статус на резервацията.

* Петия ни избор е справка с регистрациите

При натискане на бутона за справка излиза таблица с имената на хотелите,в които работи нашия потребител.При двойно кликане на екрана се показва резултата от фиг.31.

A screenshot of a computer

Description automatically generated

Фиг.31-Резултат от справка

При двойно кликане с мишката на избрана от нас резервация можем да видим всички детайли на резервацията,нейните услуги.Имаме право да сменим статуса на резервацията от одобрена на премахната и обратното.

* Шестия ни избор е спраква за използваемостта на стаите

Отново използваме формата от фиг. 21.

A screenshot of a computer

Description automatically generated

Фиг.32 – Резултат от справка

Резултата от справката е изкаран на екрана като таблица,която се състои от полета с номера на стаята,цената на стаята,типа на стаята и използваемостта и в даден период от време. В търсачката може да бъдат търсени стаи по съответните четири колони. Таблицата се актуализира при писане на определени символи.

# Структура на проекта

За най-цялостно изпълнение на заданието  се използва многомодулна архитектура,която е разделена на 5 модула. Целта на архитектурата е да има ясно разграничение между модулите, добра визуализация и лесна имплементация на нови функционалности в системата.

**Структура на проекта и избор на архитектурен модел (MVC)**

Проектът използва архитектурния модел MVC (Model-View-Controller), който разделя приложението на три основни компонента с различни функции:

1. **Модел (Model):** Представлява данните на приложението и бизнес логиката за тяхната обработка. Съдържа структури от данни и методи за манипулиране на информацията.
2. **Изглед (View):** Отговаря за визуализацията на данните пред потребителя. Включва всички графични елементи, форми и таблици. При използване на JavaFX, изгледът ще бъде графичният интерфейс.
3. **Контролер (Controller):** Служи като посредник между модела и изгледа. Приема потребителски вход, обработва го и съответно обновява модела или изгледа. В JavaFX, контролерът обработва събития от потребителския интерфейс (например кликвания на бутони) и обновява данните в модела и изгледа.

**Взаимодействие и поток на данни:**

* Контролерът получава вход от потребителя и извиква методи на модела за обработка.
* Моделът може да уведомява контролера за промени (например чрез Observer pattern).
* Контролерът обновява изгледа, за да отрази промените в модела.
* Потокът на данни е двупосочен:
  + Потребителски вход: Изглед -> Контролер -> Модел
  + Актуализация на изгледа: Модел -> Контролер -> Изглед

**Обосновка за избора на MVC:**

Изборът на MVC се основава на следните предимства:

1. **Разделяне на отговорностите:** Разделянето на приложението на модел, изглед и контролер води до:  
     
   * **По-чист код:** Всеки компонент има ясна функционалност и кодът е по-лесен за четене, поддръжка и промяна.
   * **Ефективна екипна работа:** Разработчиците могат да работят едновременно върху различни компоненти.
2. **Лесна промяна на изгледа:** MVC позволява промяна на потребителския интерфейс без да се засяга бизнес логиката в модела, което осигурява гъвкавост при дизайна.
3. **Повторна употреба на компоненти:** Моделът и контролерът могат да бъдат използвани повторно в други части на приложението или в бъдещи проекти, което спестява време и усилия.
4. **По-лесно тестване:** Всеки компонент може да бъде тестван независимо, което улеснява откриването и отстраняването на грешки.

Проектът може да следва архитектура MVC, но все пак е разделен ан 5 подмодула, с цел да се абстрахират функционалностите на приложението. Те са следните:

* **API Module:** Модулът е разделен на две главни директории – main и test. В главната директория се намират няколко поддиректории на главната.Първата от тях е основната база,чрез която функционират процесите,във втората се намират енумерациите,които използваме за различните типове клиенти,резервации в хотелската система.В третата поддиректория се намират грешките,които хвърля системата,а в последната се намират моделите.  В моделите се намират основните обекти,класове за създаване,визуализация, влизане в системата,справки и актуализации ,които са основата на нашите процеси,заради които работи хотелската система. Втората директория е за тестове,където се тества всяка една функционалност на модела.

* **CORE Module:** Модулът е изграден от две главни директории - main, където се реализира API module и test. Там се намира билдването от към ентитита към основните обекти на системата,всички процеси. Също така имаме и изключения,които се обработват в реално време с помощта на логъри и проследяват поведението на приложението,всички инициализации на всеки един клас-процес,конвъртър и репозитори.

* **DOMAIN Module:** В домейн модула се намират кодирането на паролите на потребителите в системата и Singleton анотации, които дават гаранция, че от всеки един клас и процес ще има само една единствена инициализация

* **PERSISTANCE Module:** чрез него си комуникираме с базата данни.Там се намират връзките с Hibernate, с всяко едно същесъвуващо репозитори и неговите реализации,създаване на админ и хотелски услуги,енумерации и ентититата- обектите от базата данни.

* **REST Module:** в него се намират всички контролери на базата с помощта на JavaFx и всеки един потребител в системата си има уникални контролери.

A diagram of a model

Description automatically generated

Фиг.33 – Многомодулна архитектура

# Проектиране на системата

## База от данни

Базата от данни е реализирана чрез Hibernate ORM. Благодарение на него, ще можем успешно да реализираме връзката от приложението към базата, позволявайки ни да извършваме различни операции по базата. Hibernate от своя страна под капака извършва няколко жизнено важни неща за една база, а именно физическата връзка към нея, създаването, манипулирането на данни, така и обработването на заявки. Също така Чрез Session Factory, Hibernate позволява да осъществим така наречения Mapping на членовете в базата към Класовете Entity в нашето приложение. Благодарение на това, приложението може лесно да извършва промени върху данните, без да трябва да отваря нови сесии. В базата съществуват няколко члена:

* UserEntity - реализира таблицата Потребители. Заети полета:
  + id - Идентификатор на Потребител, използва UUID (

Universally unique identifier), където той бива генерериан автоматично от Hibernate, чрез стратегия от Анотацията GeneratedValue

* firstName - име на потребителя
* lastName - фамилия на потребителя
* phone - телефон
* username - потребителско име, тип низ
* role - роля на потребителя, тип Енумерация от вече зададени 4 роли ( администратор, собственик, мениджър, рецепционист или още служител )
* email - имейл
* hashedPassword - хеширана парола в тип низ, Използва се за да се подобри сигурността на системата, и да не се разкриват паролите на потребителите при достъпване
* hotelList - списък от хотели, в които потребителя е назначен и/или е собсвеник. Връзката е Много:Много, където типа извличане (Fetching) е нетърпелив ( Eager ). При такова извличане трябва да се вземе под внимание, не само за тази но и за други възки в базата, сериализирането на обектите. За улеснения е използван Lombok в проекта и е желателно, за да не стават безкрайни сериализации на обекти поради циклично обръщение на потребител към хотел и обратно, да се изключи ToString() за полета с нетърпеливо извличане.
* ServiceEntity - Реализира таблицата Услуги с полета:
  + id - Идентификатор на услуга
  + serviceName - име на услугата
  + price - цена на услугата, тип BigDecimal
* ClientEntity - Реализира таблицата Клиенти с полета:
  + id - Индентификатор на клиент
  + firstName - Име на клиента, тип низ
  + lastName - Фамилия на клиента, тип низ
  + phone - Телефон на клиента
  + ucn - абревиатура на Unique Civil Number, или още ЕГН
  + address - адрес на регистрация на клиент
  + email - имейл адрес на клиент
  + birthDate - дата на раждане
  + issueDate - дата на издаване на документа за самоличност
  + expireDate - дата на изтичане на документа за самоличност
  + issuedBy - институцията от която документа е издаден
  + rating - Рейтинг на клиента, използвана е Енумерация с видовете рейтинги от системата.
* Hotel Entity - Реализира таблицата Хотели с полета:
  + id - Идентификатор на хотел, използва UUID, генериран автоматично от ORM
  + name - име на хотел, тип низ
  + location - локация на хотел, тип низ
  + rating - рейтинг или още в някои литератури оценка на хотел в “звезди”
  + serviceList - списък от всички услуги които хотела предлага
  + userList - списък от всички лица обвързани с хотела ( рецепционисти, мениджъри, собственици)
* Room Entity - Реализира таблица Стаи, полета:
  + id - Идентификатор на стаята
  + floor - етаж на стаята
  + roomNumber - Номер на стаята, тип Integer
  + price - цена, тип BigDecimal
  + type - тип на стая, отново дефиниран в Енумерация
  + hotel - хотел към който отговаря стаята
* Reservation Entity  - Реализира таблицата Резервации:
  + id - Идентификатор на резервацията
  + type - Тип на резервацията, дефиниран от Енумерация
  + startDate - начална дата на резервацията
  + endDate - крайна дата на резервацията
  + price - Цена на резервацията
  + room - стаята която резервацията ще заеме
  + client - Клиентът който е наел стаята
  + createdBy - От кого е създадена резервацията
  + status - Статус на резервацията, Енумерация от моментни два типа “Потвърдена” и “Отменена”
  + services - списък от услуги които ще бъдат причислени към финалната цена на резервацията

A screenshot of a computer

Description automatically generated

Фиг.34 – Модел на базата

Тези Обекти-членове или Entities имат съответно хранилище, към което отговарят. Съществуват 7 хранилища, за да се избегне написването на определени методи като sav(), delete(), getAll(), findById(), се използва Общо хранилище - Base Repository, което отговаря за тези операции. Класа изпозлва генерици ( Generics ), като позволява да връща Обекти от тип Entity Marker, който от своя страна бива имплементиран от всяко едно Entity

* Базово хранилище - в него се съдържат няколко метода за работа, които ще бъдат преизползвани като се наследи класа от определеното. Класа използва и типизация <T extends Entitymarker, E extends UUID>
  + save - запазва Обект от тип Т в базата
  + delete - итрива Обект от тип Т в базата
  + findByID - връща Optional от тип Т
  + getAll - връща всички Обекти от тип T в дадена таблица
  + еxistByID - връща булева стойност дали даден обект съществува в базата. Еквивалент на метода findByID(). Удобен за булеви проверки
* ClientRepository - Не съдържа специфични методи за таблицата клиент. Наследява Базовото Хранилище.
* HotelRepository - методи за работа с хотели и потребители към тях. Наследява Базово Хранилище
  + findHotelById(UUID id) - Намира хотел по уникален ключ
  + findAllByOwner(UserEntity owner) - намира всички хотели на собственик
  + findManagersWithHotels() - намира всички мениджъри на хотели
  + findAllHotelsWithUsers() - намира всички хотели с потребители
  + findAllByManager(UserEntity manager) - намира всички хотели от мениджър
  + findAllByUser(UserEntity user) - намира всички хотели по подаден потребител
* ReservationRepository
  + isRoomAvailable(RoomEntity room, LocalDate startDate, LocalDate endDate) - при подадени дати за начало и край, както и стая, проверява дали стаята е свободна.
  + countServiceUsage(UUID hotelId,LocalDate startDate, LocalDate endDate)- при подаден хотел и дати, връща списък от Потреблението на услуги, комплектован в DTO ( Data Transfer Object )
  + getReservationsByReceptionists(UUID hotelId, LocalDate startDate, LocalDate endDate) - при подадени параметри за начална и крайна дата, както и хотелски идентификатор, връща списък от Резервациите направени от рецепционисти, комплектовани в DTO.
  + getRoomUsageByHotelAndDate(UUID hotelId, LocalDate startDate, LocalDate endDate) - връща списък с използваността на стаи за подаден хотел, в даден времеви период.
  + getReservationsByHotelAndDate(UUID hotelId, LocalDate startDate, LocalDate endDate)- връща всички резервации за хотел в даден интервал от време, Върнатия резултат е Списък от резервации.
  + findAllByHotelId(UUID hotelId) - намира всички резервации присъщи на хотел.
* RoomRepository - наследява Базово Хранилище
  + findAllByHotel(HotelEntity hotelEntity) - намира всички стаи при подаден хотел
* ServiceRepository - Наследява само базовото хранилище
* UserRepository
  + findByUsername(String username) - намира потребител по подаден низ.
  + findAllOwners() - връща списък от всички собственици
  + findAllManagers() - връща списък от всички мениджъри
  + findAllReceptionist() - намира и връща всички рецепционисти на хотел

След като разгледахме как е реализирана базата е време да покажем, как се осъществява комуникацията между елементите в системата. В приложението използваме монади, като преходът изглежда както следва:

A diagram of a software development process

Description automatically generated

Фиг.35 – Процес на обработка на данните

Както се вижда, диаграмата започва с потребителско действие. Това действие ще бъде отчетено в .fxml файла отговарящ на формата в която потребителя работи. След това според избора и действието, тези ActionEvents ще бъдат интерпретирани от Контролер , съответстващ на сцената. Този контролер може да чака и слуша ( Listen and Wait ) . След действие, контролера комплектова данните в Input присъщ към модула API. След което данните биват приети от процес под формата на Input. Ако е нужно, входните данни ще бъдат обработени за валидиране. Целта на процеса е да вземе данните от Input и да ги обработи под формата на заявка. След което да изпрати резултат, към контролера. От своя страна конторлера освен, че изпраща резултат, той може и да приема, като използва fold функцията на монада. По този начин при правилен резултат, той бива изобразен на екрана, а при грешен се извежда Alert чрез класа AlertManager.  
  
Логерът в този проект се използва за мониторинг и проследяване на събития, грешки и важни операции в приложението. Той предоставя централизирана система за записване на информация, която може да се използва както за дебъгване по време на разработка, така и за анализ на поведението на приложението в продукционна среда. Чрез логера се записват критични съобщения като грешки при взаимодействие с базата данни, некоректни входни данни или успешни бизнес операции. Интеграцията му позволява да се задават различни нива на известия (например **INFO**, **ERROR**), което улеснява филтрирането и анализа на събитията. Логерът е имплементиран по начин, който минимално влияе на производителността, и е конфигуриран да записва логовете в консола, файлове или други външни системи според нуждите.  
  
Относно зависимостите в проекта, те могат да бъдат разгледани по следния начин:

* JavaFX и свързани библиотеки:
  + com.dlsc.formsfx, org.kordamp.bootstrapfx, com.github.almasb.fxgl, org.controlsfx: Използват се за създаване на модерни, интуитивни и функционални JavaFX графични интерфейси. FormsFX улеснява формите, BootstrapFX осигурява CSS стилове, а FXGL е за разработка на игри или сложни визуализации.

* Тестове:
  + org.junit.jupiter, org.mockito, org.testfx: Използват се за единични и интеграционни тестове. Mockito се грижи за мокване на обекти, а TestFX е за автоматизирано тестване на JavaFX интерфейси.

* Hibernate и база данни:
  + org.hibernate.orm, com.oracle.database.jdbc, jakarta.persistence-api: Тези зависимости управляват връзката с базата данни (Oracle) чрез ORM подход и предоставят инструменти за работа с обекти и трансакции.

* Логване и мониторинг:
  + org.log4j:Осигуряват логиране на събития и грешки в приложението. SLF4J предоставя общ интерфейс, а Log4j е за реалната имплементация.

* Валидиране и функционалност:
  + jakarta.validation, hibernate-validator, jakarta.el:Използват се за валидиране на входни данни, което гарантира правилността на информацията в приложението.

* Утилитарни библиотеки и Монади:
  + io.vavr: Подобряват функционалността чрез функционално програмиране (Vavr)Инструменти за продуктивност:
  + org.projectlombok:Използва се за автоматично генериране на шаблонен код (например getter-и и setter-и), което прави кода по-компактен и четим.

В приложението са интегрирани различни видове тестове, които гарантират правилното функциониране на системата, като се обхващат всички важни аспекти от работата ѝ. Те са организирани в три основни категории:

1. **Юнит тестове (Unit Tests)**
   * Използват **JUnit**, за да тестват отделните компоненти на приложението (например API слоеве).
   * Фокусът е върху изолирано тестване на логиката на малки методи и класове
   * Това позволява засичане на проблеми и гарантира стабилност на кода.
2. **Интеграционни тестове (Integration Tests)**
   * Използват **Mockito**, за да се тестват сложни процеси чрез мокване на зависимостите между компонентите.
   * Това осигурява безопасна среда за тестване на бизнес логиката и валидиране на коректната комуникация между различните слоеве
   * Мокването позволява лесно симулиране на специфични сценарии, които иначе биха били трудни за възпроизвеждане.
3. **Функционални тестове (Functional Tests)**
   * Реализирани чрез **TestFX**, тези тестове се фокусират върху JavaFX UI
   * Те симулират реални действия на потребителя като кликове, въвеждане на данни и превключване между сцени, за да гарантират, че интерфейсът е интуитивен и работи правилно.
   * Това включва проверка на визуалната част на приложението, както и на функционалността, свързана с навигацията и взаимодействието с потребителя.

## Модел на Чен

A diagram of a company

Description automatically generated

Фиг.26-Модел на Чен

## Use-case Диаграма

A diagram of a company

Description automatically generated

Фиг.37-Use- case Diagram

## Class Диаграми

### Entity & Repository

Както е описано по-горе, връзките помежду обекти и хранилища, може да видим чрез следната диаграма. Ясно е че много обекти използват референция към други. По този начин се изгражда добре затворена връзка между елементите на системата.

A screenshot of a computer screen

Description automatically generated

Фиг.38 – Клас диаграма на единиците ( Entities и Repository )

### Converters

Конвертерите в приложенията ще се използват за по-лесно обработване на данните от един тип в друг. Пример за това е обработването на обекти от базата в обекти за сцените. Целта е да се използват два отделни обекта да не се наруши структурата на проекта, но и да предпази отделните обекти в използване на модул , в който не отговарят.

A screenshot of a computer program

Description automatically generated

Фиг.39 – Клас диаграма на конвертерите

### Modules

### **Modules**

Както вече споменахме проекта е изцяло модулно базиран, реализиран чрез MAVEN. За да има правилна структура, модулите трябва да комуникират помежду си, без да се настъпва така нареченото “circular dependecy” в литературата на софтуерните архитектури.

A diagram of a computer

Description automatically generated

Фиг.40 – Диаграма на модулите

### Controllers

В едно добре структурирано приложение контролерите служат като скелет за управление на взаимодействието между потребителския интерфейс (UI) и основната бизнес логика. Причините са няколко

* Централизирано управление на взаимодействието: Контролерите действат като централен слой, който посредничи при въвеждането на потребители, обработва бизнес логиката и актуализира потребителския интерфейс. Това разделяне гарантира, че компонентите на потребителския интерфейс остават леки и фокусирани само върху рендиране на визуални елементи и улавяне на входове.
* Инкапсулиране на логиката: Контролерите капсулират логиката, необходима за обработка на конкретни изгледи или екрани в приложението.
* Обработване на въведени данни от потребители (напр. щраквания върху бутони или изпращане на формуляр).
* Извличане или изпращане на данни до процесите
* Актуализиране на съответните UI елементи въз основа на обработени данни. Това разделяне прави приложението по-лесно за поддръжка и разширяване.
* Интеграция с бизнес логиката: Контролерите взаимодействат с базовите услуги, процеси или репозиторита, за да извличат, създават, актуализират или изтриват обекти
* Полето за търсене във вашия изглед на услуги се управлява от контролер за филтриране и сортиране на данни от таблици динамично въз основа на въвеждане от потребителя.

A screenshot of a computer program

Description automatically generated

Фиг.41 – Контролерите