**Технически университет – София**
**Факултет по приложна математика и информатика**

Учебна дисциплина:
“Софтуерни технологии”

# Архитектурен проект на софтуерна система

на

Андон Георгиев Макенджиев, фак. №: 471222047,
Николай Венциславов Сугарев, фак. №: 471222078,
Божидар Валентинов Михайлов, фак. №: 471222053,
Християн Георгиев Братов, фак. №: 471222049,
Александър Сашов Младенов, фак. №: 471222059

## Тема:

“Онлайн резервационна система за хотели“

**02.11.2024г.**

**София**

**СЪДЪРЖАНИЕ:**

# Секция № 1

## 1. Въведение

Този документ ще ви запознае с начина на изграждане и разработване на „Онлайн резервационна система за хотели“. Включени са както нефункционалните, така и техническите изисквания, чрез които тази платформа ще функционира. Представени са и диаграми, подпомагащи за цялостното разбиране на проекта и неговата реализация.

## 1.1 Крайна цел на проекта

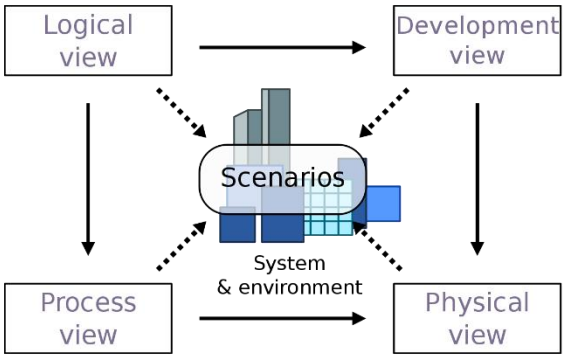
Разработеният проект ще бъде на тема „Онлайн резервационна система за хотели“, което означава, че финалният ни продукт ще е под формата на уеб приложение. Целта е да се достигне крайна, напълно функционираща и максимално безпроблемна платформа, която потребителите ще могат да използват за съответното предназначение. Заложили сме на технически спецификации, доказали се в практиката, които ще осигурят постигането на нефункционалните изисквания на системата.

## 1.2 Предназначение на документа

Документът служи като основен ресурс за всички членове на екипа, осигурявайки необходимата информация за разработка, внедряване и тестване на системата. Той ще предостави ясни указания за архитектурните решения и спецификациите, необходими за създаването на ефективна и безопасна платформа. Основната идея е да се гарантира, че всички участници в проекта са наясно с целите, изискванията и желания резултат.

## 1.3 Структура на документа

Системата е базирана на модела за софтуерна архитектура на Филип Крухтен (Philippe Kruchten) – „4+1“, включващ четири изгледа (logical view, physical view, development view, process view), както и един use case view. Този модел предлага цялостен подход, като помага на заинтересованите страни да разберат различни аспекти на системата и улеснява сътрудничеството през целия жизнен цикъл на разработка.




**Фиг. 1: „4+1“ модел за СА**

## 1.4 Дефиниции & Абревиатури

- **СА –** софтуерна архитектура
- **IDE –** integrated development environment (интегрирана среда за разработка)
- **QA –** quality assurance (инженери по осигуряване на качеството)
- **UI/UX –** user interface / user experience
- **ASP .NET –** уеб платформа на Microsoft
- **MVC –** module-view-controller
- **SQL –** structured query language
- **DDoS –** Distributed Denial-of-Service
- **API –** Application Programming Interface
- **CSS –** Cascading Style Sheets

## 1.5 Ограничения и насоки за разработване на проекта

1. **Интегрирана среда за разработка (IDE):** избраната среда за разработка на този проект е **Visual Studio**. Всички членове на екипа използват съответното IDE за писане на код, имплементиране на нови функционалности, отстраняване на грешки и тестване.

2. **Структура на екипа:** всеки член на екипа има определени отговорности.

   - **Заинтересована страна (Stakeholder)**
     **Отговорен за:** осигурява финансиране и цялостна подкрепа спрямо разработвания проект.

   - **Мениджър на проекта**
     **Отговорен за:** наблюдава етапите на проекта, възлага задачи и гарантира спазването на крайните срокове на проекта.

   - **Бизнес анализатор**
     **Отговорен за:** събира и разтълкува изискванията след проведен разговор с крайния потребител, гарантира те да бъдат ясни и постижими.
   - **Софтуерен архитект**
     **Отговорен за:** дефинира архитектурата на проекта, взема решения относно технологичния стек и осигурява постигането на нефункционалните изисквания на системата.

   - **Софтуерни разработчици**
     **Отговорни за:** писане на код, внедряване на нови функционалности и поддържането на качествен програмен код, съобразен с доказалите се в практиката принципи и шаблони.

- **Инженери по осигуряване на качеството (QA)**
  **Отговорни за:** тестване на системата, идентифициране и докладване на грешки и нередни неща по кода, както и осигуряване на надеждност на софтуера.

- **UI/UX дизайнери**
  **Отговорни за:** проектиране и внедряване на потребителски интерфейси за максимално потребителско изживяване.

- **Администратори на бази данни**
  **Отговорни за:** управляване архитектурата на базата данни, производителността и целостта на данните.

3. **График на проекта:** изготвен е подробен график на проекта, който очертава ключови етапи, резултати и крайни срокове. Ще се поддържат редовни актуализации и комуникация, за да бъдат всички членове на екипа известени с напредъка на проекта.

4. **Контрол на версиите (Version Control):** като система за контрол на версиите ще се използва „Git“ за управление на промените по време на разработката на системата, като това улеснява сътрудничеството и проследяването на историята на проекта.

5. **Стандарти за писане на код:** наложени са последователни стандарти за писане на код и добри практики. Ще се извършват прегледи на кода, за да се поддържа качеството и последователността на код в проекта.

6. **Процедури за тестване:** множество процедури за тестване на проекта ще бъдат внедрени за откриване и отстраняване на всички нередности още в началото на работния цикъл.

7. **Мерки за сигурност:** ще се интегрират в дизайна и разработката на системата, с цел защита на чувствителни потребителски данни и намаляване на потенциални външни заплахи. Ще се провеждат редовни одити на сигурността и актуализации.

# Секция № 2

## 2 Нефункционални изисквания

Архитектурата на „Онлайн резервационната система за хотели“ осигурява изпълнението на следните архитектурни характеристики:

- **Достъпност**

От изключително важно значение е нашата система да е винаги на разположение за потребителите. Една от най-неприятните ситуации възможни би била : клиент да има желание за “last minute” резервация, а платформата да не работи. За да избегнем такива ситуации, системата трябва да поддържа висока степен на достъпност и бързо възстановяване при срив. Това ще се постигне чрез наличието на резервни сървъри и система за автоматично превключване.

- **Разширяемост**

Системата трябва да бъде гъвкава и лесно адаптивна за добавяне на нови функционалности в бъдеще, без да се нарушава съществуващата работа. Всяко имплементиране на нова логика трябва да бъде направено без много промени в основния код  за възможно най-кратко време.

- **Производителност**

Когато хората търсят хотели, те не искат да чакат твърде дълго за зареждане на резултатите. Забавянето на системата може да доведе до загуба на клиенти, което от своя страна води до финансови загуби, ниски оценки и лоши ревюта от крайните потребители. Платформата трябва да извършва обработка на потребителски заявки в нормално възприетото за това време (2 секунди), дори и при многократно нарастване на броя едновременно работещи с нея клиенти.

- **Сигурност**

Поверителността и защитата на данните са изключително важни, особено когато става въпрос за лични данни и плащания. Системата трябва да гарантира, че чувствителни данни няма да бъдат достъпни за трети лица, както и да има разработена защита срещу често срещани атаки като **DDoS** и **SQL Injection.**

- **Възможност за тестване**

Важно е да можем лесно да тестваме системата, за да сме сигурни, че всичко работи както трябва. Затова трябва да има интегрирани инструменти за тестване, които да позволяват да се проверяват различните функционалности, както и да се симулират различни натоварвания. Това ще помогне да се откриват и отстраняват проблеми преди те да стигнат до клиента.

- **Възможност за комуникация с външни системи**

За да може системата да комуникира с други услуги (например платформи за плащане и хотелиерски системи), тя трябва да поддържа стандартизирани формати на данни като JSON и XML. Това ще улесни обмена на информация и интеграцията с тези системи.

- **Използваемост**

Използването на платформа, която е сложна и объркваща , води до загуба на клиенти. Интерфейсът трябва да бъде интуитивен и приятен за ползване, като осигурява лесен достъп до всички основни функции. Освен това, системата трябва да работи добре както на настолни компютри, така и на мобилни устройства, за да предостави удобство на всички потребители.

- **Ефективност**

Системата преизползва определени ресурси, за които това е допустимо.

# Секция № 3

## 6. Архитектурно представяне

Моделът за софтуерна архитектура „4+1“, интегриращ вече познатите ви изгледи, е идеален подход за покриване на нефункционалните изисквания на разработваната система. Всеки изглед се привежда в съответствие със специфични цели: **логически** за производителност и ефективност чрез оптимизирана обработка, процес за скалируемост чрез улесняване на модулната реконфигурация; **физически** за сигурност чрез защита на чувствителни данни и „use case“ за устойчивост, подсигуряващ непрекъсната функционалност, дори в случай на повреда на сървъра. Този архитектурен шаблон предоставя цялостен подход, позволяващ дизайн и стратегии за внедряване, съобразени с всяко изискване, като по този начин подобрява цялостната устойчивост и адаптивност на крайния продукт.

## 3.1 Изгледи на модела „4+1“

**Use case view**

**Заинтересовани страни:** бизнес анализатори и ръководители на проекти, които се интересуват от разбирането на функционалните изисквания на системата и взаимодействията с потребителите.

**Предназначение / Предимства:** този изглед улавя функционалните изисквания на системата от гледна точка на крайните потребители, описвайки как те взаимодействат със системата, за да изпълнят конкретни задачи и цели.

**Видове диаграми:** use case diagram

### Logical view

**Заинтересовани страни:** софтуерни разработчици, архитекти и лица, които се интересуват от разбирането на структурата и функционалността на системата.

**Предназначение / Предимства:** този изглед се фокусира върху дизайна на системата от високо ниво, като набляга на връзките и взаимодействията между различни компоненти/модули. Той помага на заинтересованите страни да разберат как различните части на системата работят заедно.

**Видове диаграми:** class diagram, package diagram, component diagram

### Physical view

**Заинтересовани страни:** системни администратори, мрежови инженери и лица, които се интересуват от разбирането на изискванията за внедряване и инфраструктура на системата.

**Предназначение / Предимства:** този изглед разглежда аспектите на физическата инфраструктура на системата. Той помага на заинтересованите страни да оценят характеристиките на мащабируемостта, наличността и производителността на системата.

**Видове диаграми:** deployment diagram

### Development view

**Заинтересовани страни:** софтуерни разработчици, строителни инженери и лица, които се интересуват от разбирането на процеса на разработка и организацията на системата.

**Предназначение / Предимства:** този изглед обръща особено внимание на процеса на разработка на софтуер, включително организацията на изходния код. Той помага на заинтересованите страни да разберат как системата се разработва, поддържа и развива с течение на времето.

**Видове диаграми:** component diagram

**Process view**

**Заинтересовани страни:** системни архитекти, софтуерни разработчици и лица, които се интересуват от разбирането на динамичното поведение на системата и взаимодействието между различните процеси.

**Предназначение / Предимства:** този изглед се фокусира върху поведението на системата по време на изпълнение (синхронизация на процеси, разпределение на ресурси). Той помага на заинтересованите страни да разберат как системата се държи при различни условия.

**Видове диаграми:** activity diagram, sequence diagram, state diagram

# Секция № 4

## 4. Технически спецификации / Технологичен стек

„Онлайн резервационната система за хотели“ е стабилна платформа, която е предназначена да улесни бързото резервиране на определен хотел по света в сигурна и удобна за потребителите среда. Използвайки модерни и доказали се технологии, системата осигурява висока производителност, надеждност и мащабируемост, за да покрие изискванията на една динамична среда за правене на резервации.

## Технологичен стек:

- **Back-end Framework:** ASP .NET MVC

  **Определение:** ASP.NET MVC е фреймуърк за уеб приложения, разработен от Microsoft и имплементира шаблона модел-изглед-контролер (model-view-controller).

  **Причини за използване:**
  - избран е заради своята устойчивост, мащабируемост и широка поддръжка в рамките на .NET екосистемата.
  - предлага чисто разделяне на проблемите, улесняващо модулно развитие и по-лесна поддръжка.
  - използва силата на .NET екосистемата, предоставяйки фреймуърк с множество функции за изграждане на уеб приложения.

- **Front-end:** HTML, CSS & JavaScript

  Визуалната част на системата ни е изградена на основата на HTML, CSS и JavaScript, които са технологии за изграждане на потребителски интерфейс, като всяка от тях си има отделно предназначение (за структура, стилизиране, интерактивност). Ще бъде използван и вградения „Razor“ синтаксис на ASP .NET Core за интеграцията на фронтенд с бекенд и ефективното създаване на различните изгледи.

- **Database Management System:** MS SQL Server

**Определение:** MS SQL Server е собствена система за управление на релационни бази данни, разработена от Microsoft.

**Причини за използване:**

- предлага висока производителност, надеждност и стабилни функции за сигурност, които са от решаващо значение при обработката на чувствителни данни.
- безпроблемно се интегрира с .NET екосистемата, като предлага вградена поддръжка на **Entity Framework**, както и други технологии на Microsoft.

- **Integration & APIs:** RESTful APIs & WebSocket API

**Определение за RESTful APIs:** RESTful APIs са набор от архитектурни принципи за създаване на уеб услуги, които използват стандартни **HTTP** методи и следват клиент-сървър модела.

**Определение за WebSocket API:** това е комуникационен протокол, който позволява цялостна комуникация между клиент и сървър в реално време през единична дълготрайна връзка.

**Причини за използване:**

- улеснява се интеграцията с услуги на трети страни, като също се позволява и достъп до търговски данни, анализи и други основни функции.

- **Payment Gateway:** „Stripe“

  **Определение:** „Stripe“ е популярна онлайн платформа за обработка на плащания, която позволява да се приемат плащания от клиенти чрез дебитна или кредитна карта, както и други методи в различни валути. Предоставя сигурно и удобно за разработчиците API за интегриране на платежни системи в уебсайтове и приложения.

  **Причини за използване:**
  - осигурява сигурна обработка на плащанията, като това помага за изграждане на доверие с потребителите и гарантира съответствие с местните разпоредби.

- **Entity Framework**

  **Определение:** Entity Framework е технология за достъп до данни, която позволява на софтуерните разработчици да работят с релационни данни, използвайки обекти, специфични за домейна.

  **Причини за използване:**
  - опростява взаимодействията с базата данни със своето интуитивно API и инструменти, като така се намалява времето и усилията за разработка.
  - предлага интеграция с ASP.NET MVC, което позволява на разработчиците да се фокусират върху бизнес логиката, а не върху начините за достъп до данни на ниско ниво.

- **Testing:** NUnit / xUnit

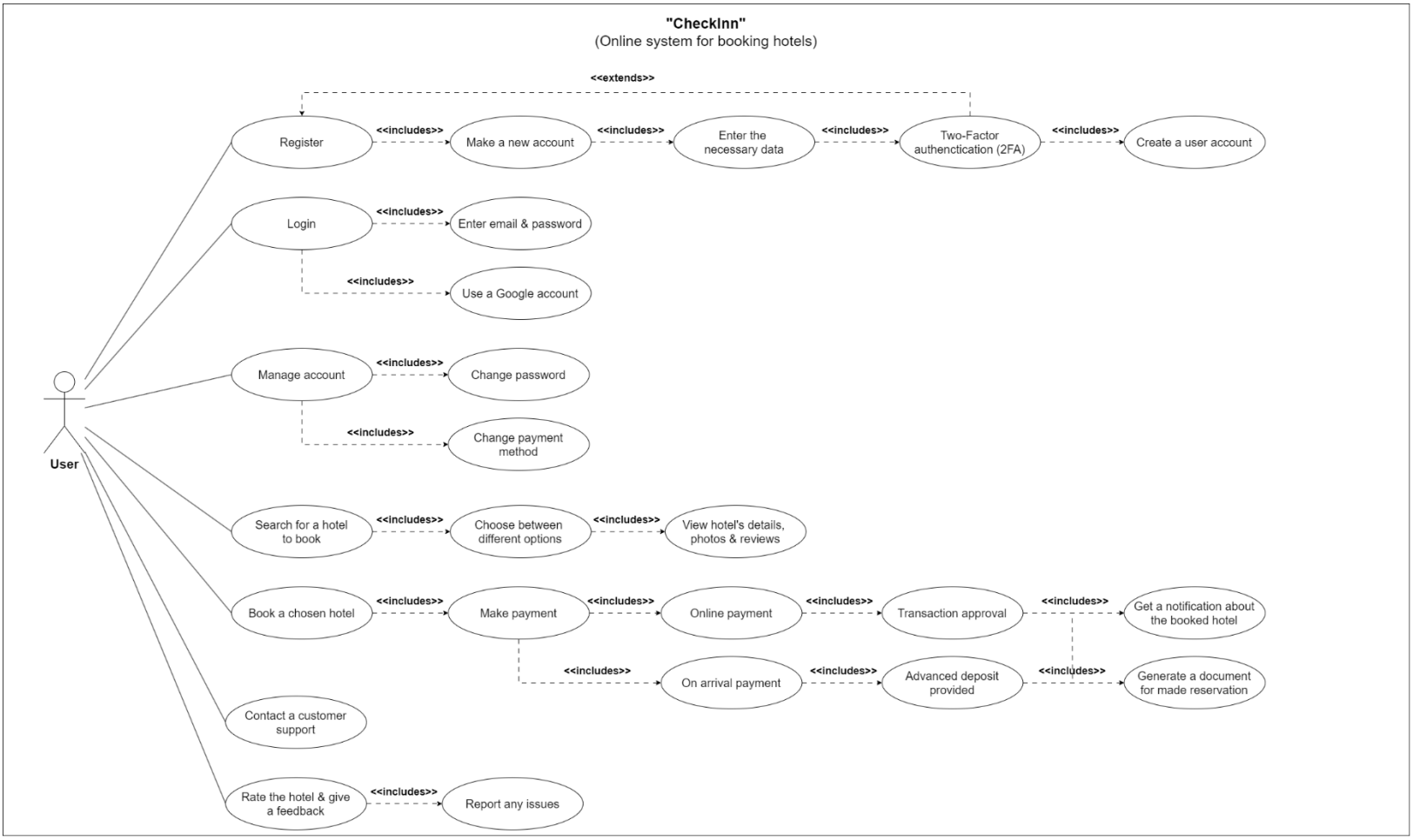
  „Unit testing“ е начин за тестване на софтуер, който оценява отделни компоненти на кода, като функции или методи, за да се гарантира, че работят оптимално и правилно. **NUnit** и **xUnit** са популярни библиотеки за .NET приложения, предоставящи инструменти и анотации за създаване, управление и изпълнение на модулни тестове.

  **NUnit** е известен със своя утвърден набор от функции и лекота на използване, докато **xUnit** се фокусира върху разширяемостта и производителността за нуждите на съвременните тестове.

# Секция № 5

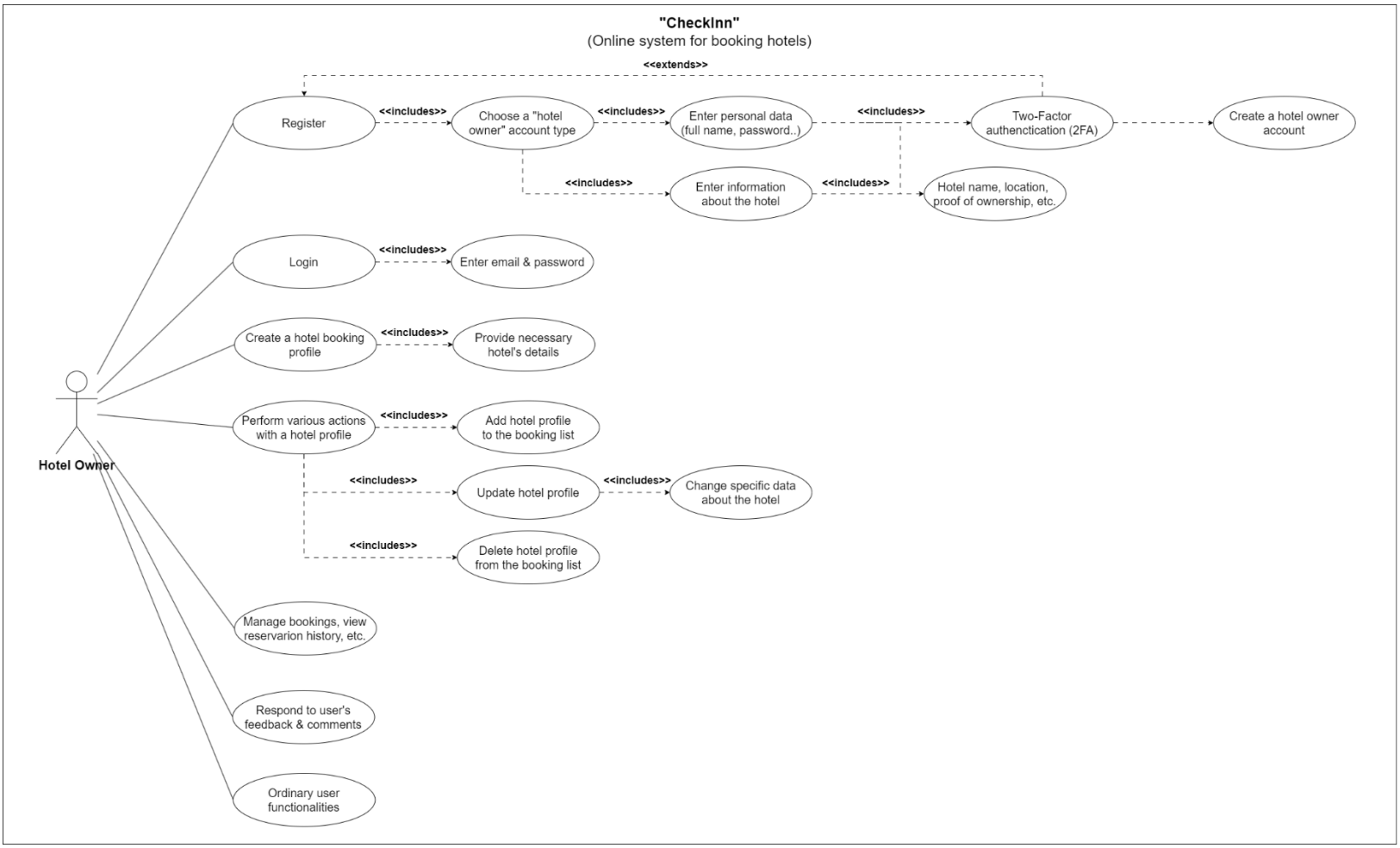
## 5. UML диаграми

## 5.1 Use Case View:



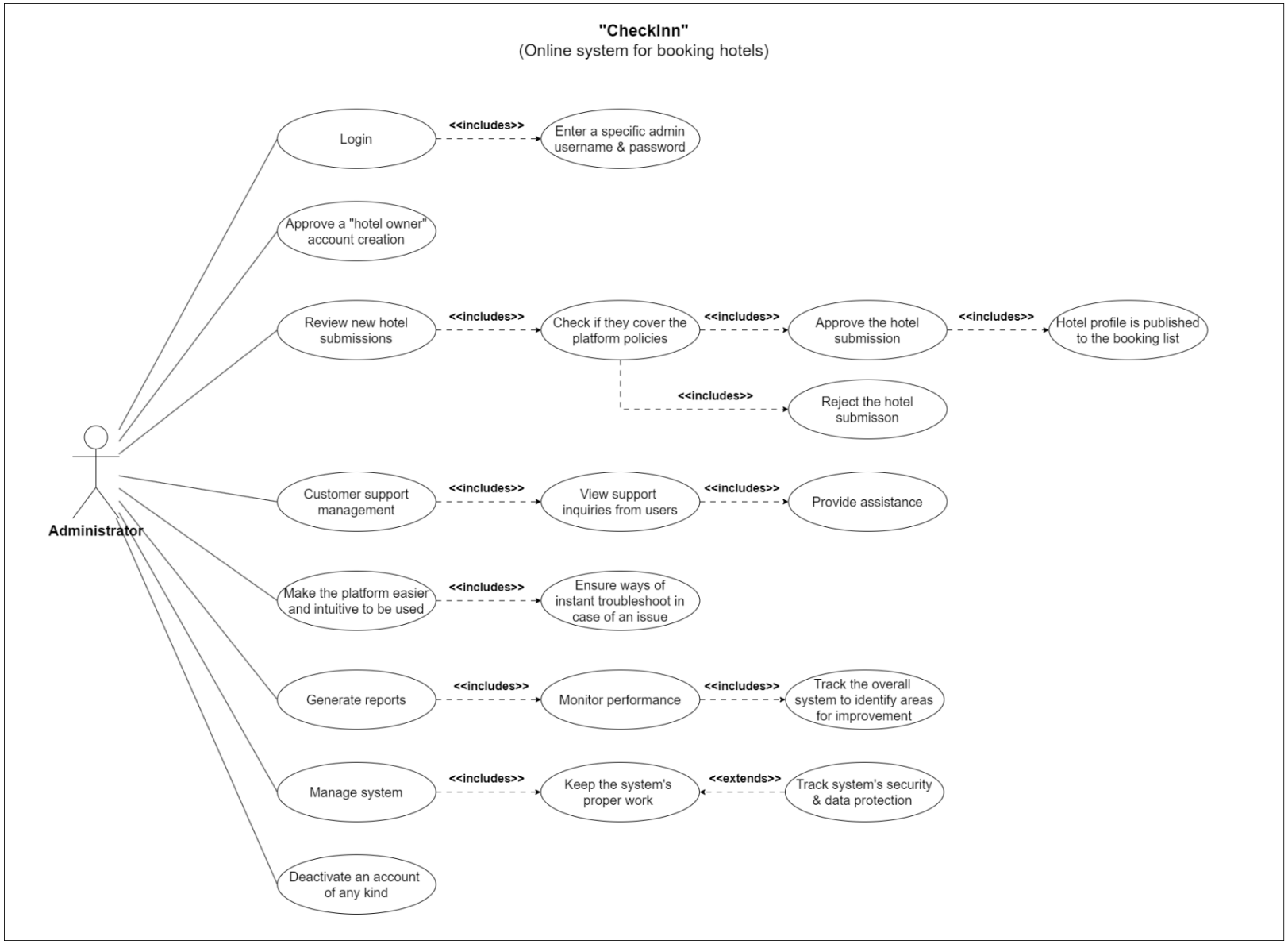

**Диаграма 1:**
Use Case Diagram - User

# II



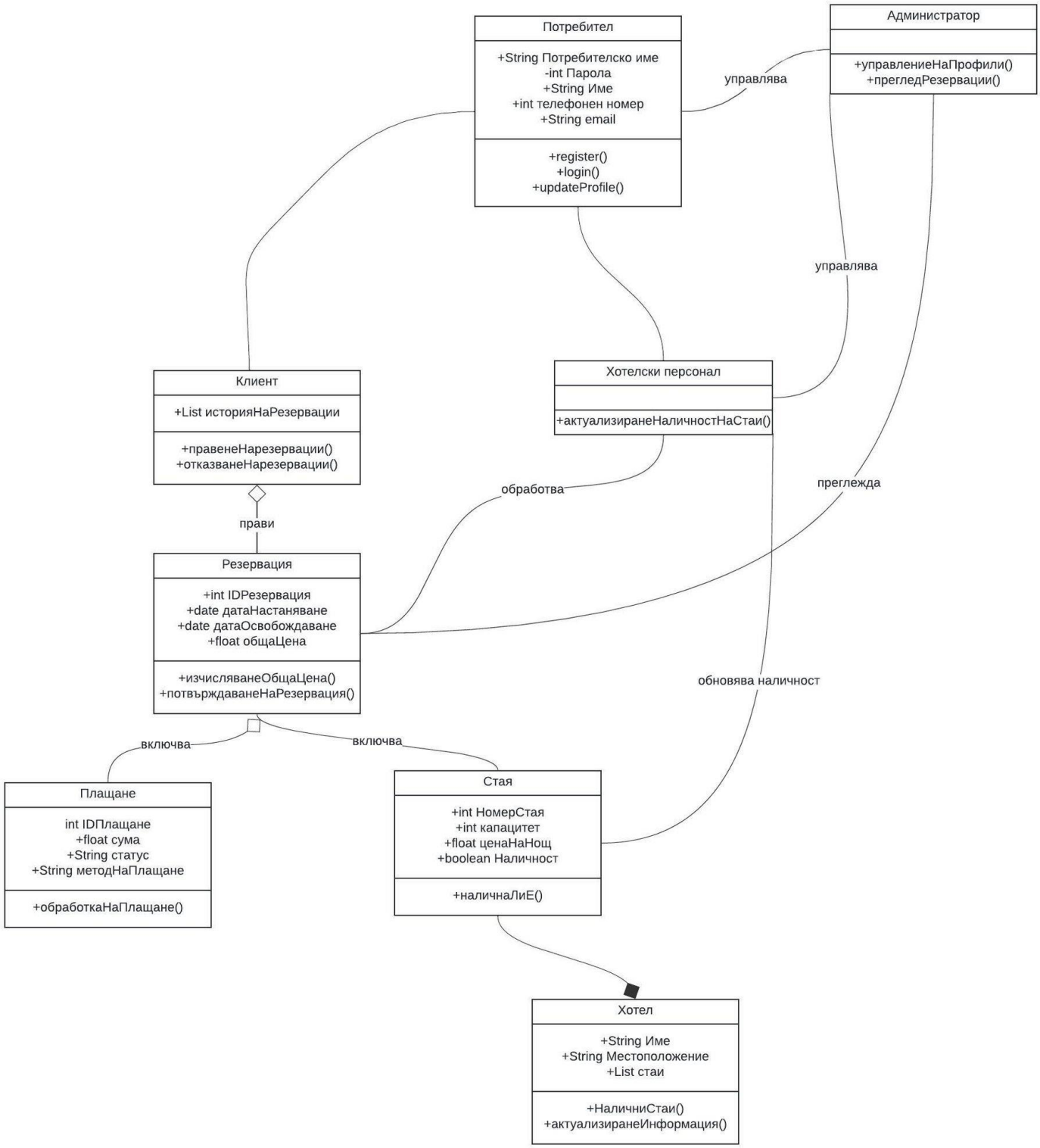

**Диаграма 2:**
Use Case Diagram – Hotel Owner

# III



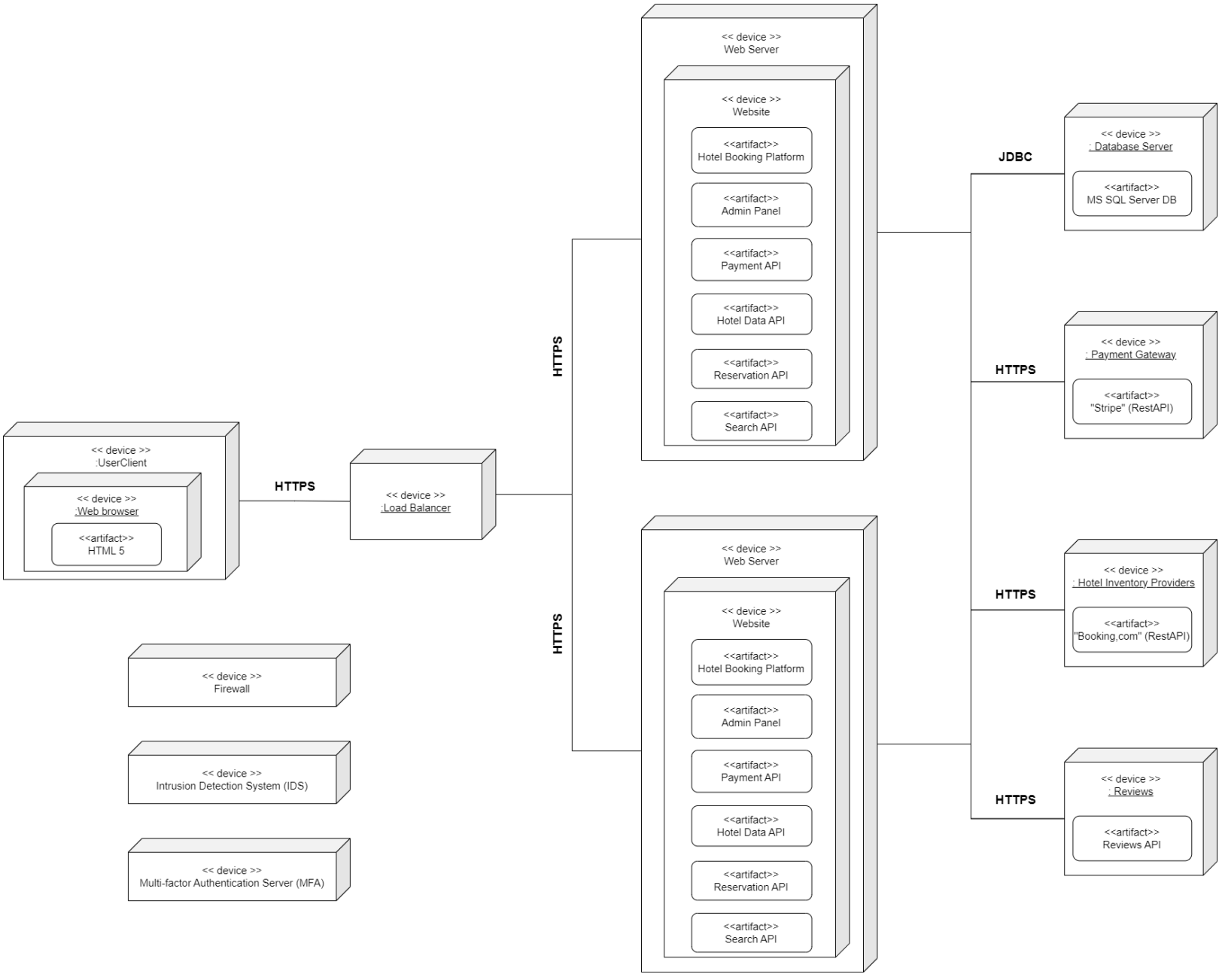

**Диаграма 3:**
Use Case Diagram – Administrator

# 5.2 Logical View:



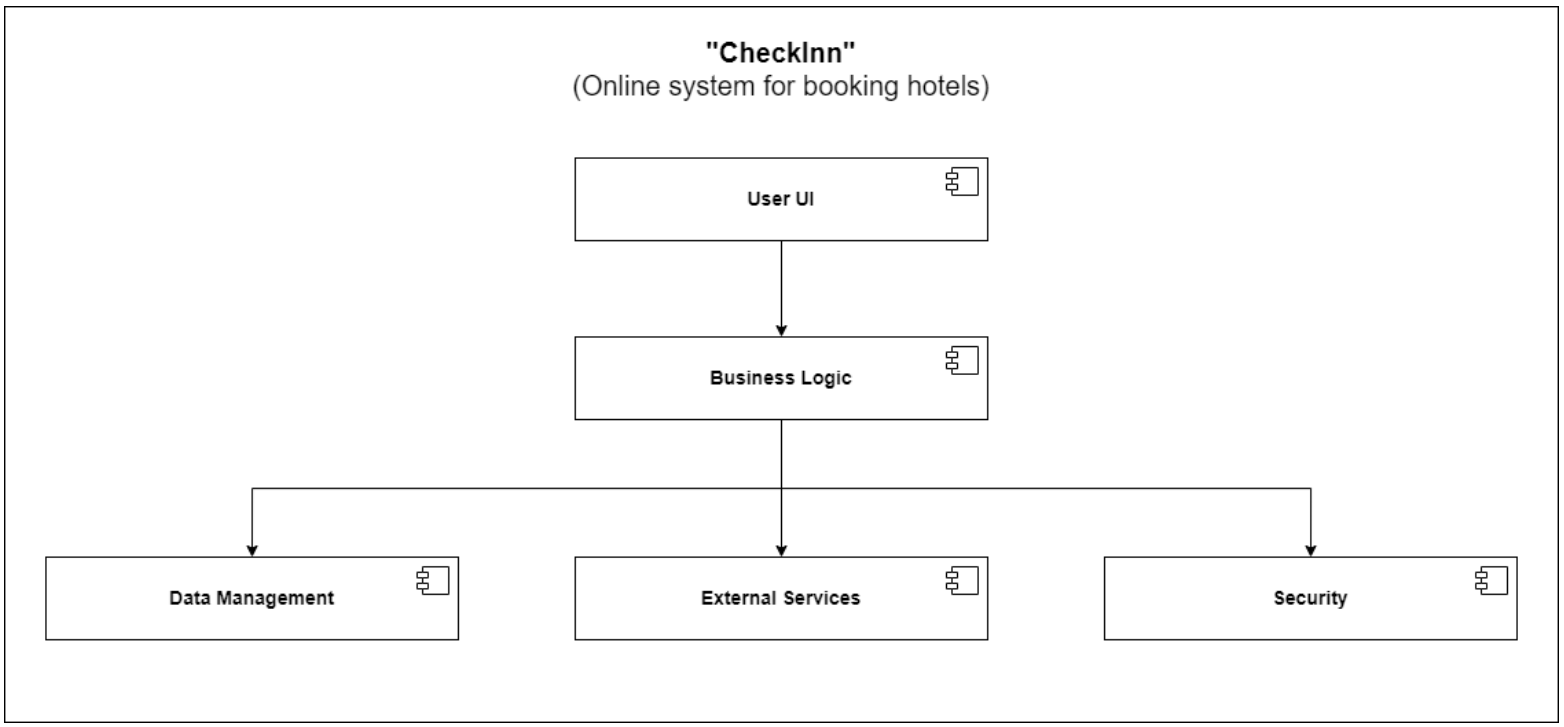

**Диаграма 4:**
Class Diagram

# 5.3 Physical View:

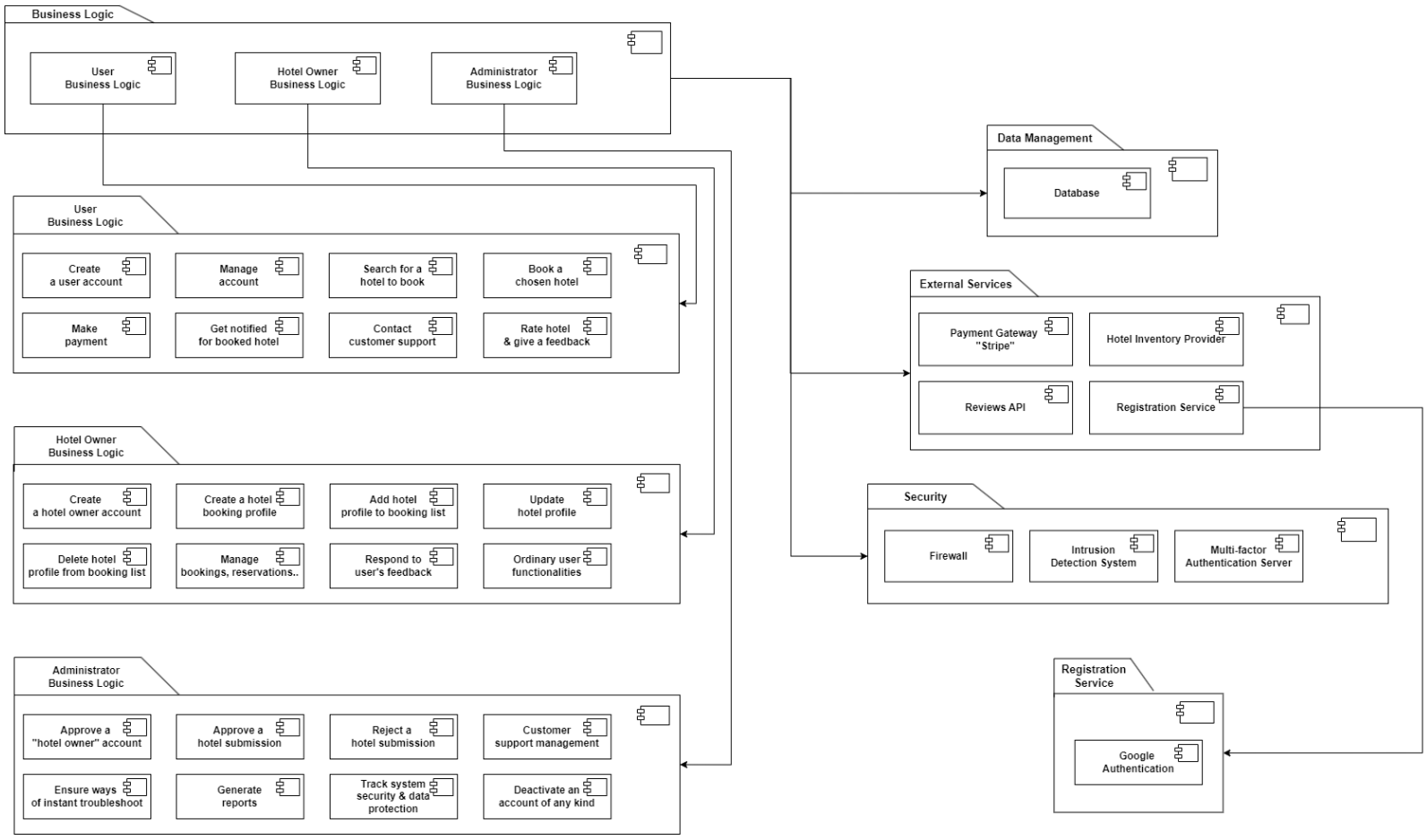



**Диаграма 5:**
Deployment Diagram

# 5.4 Development View:

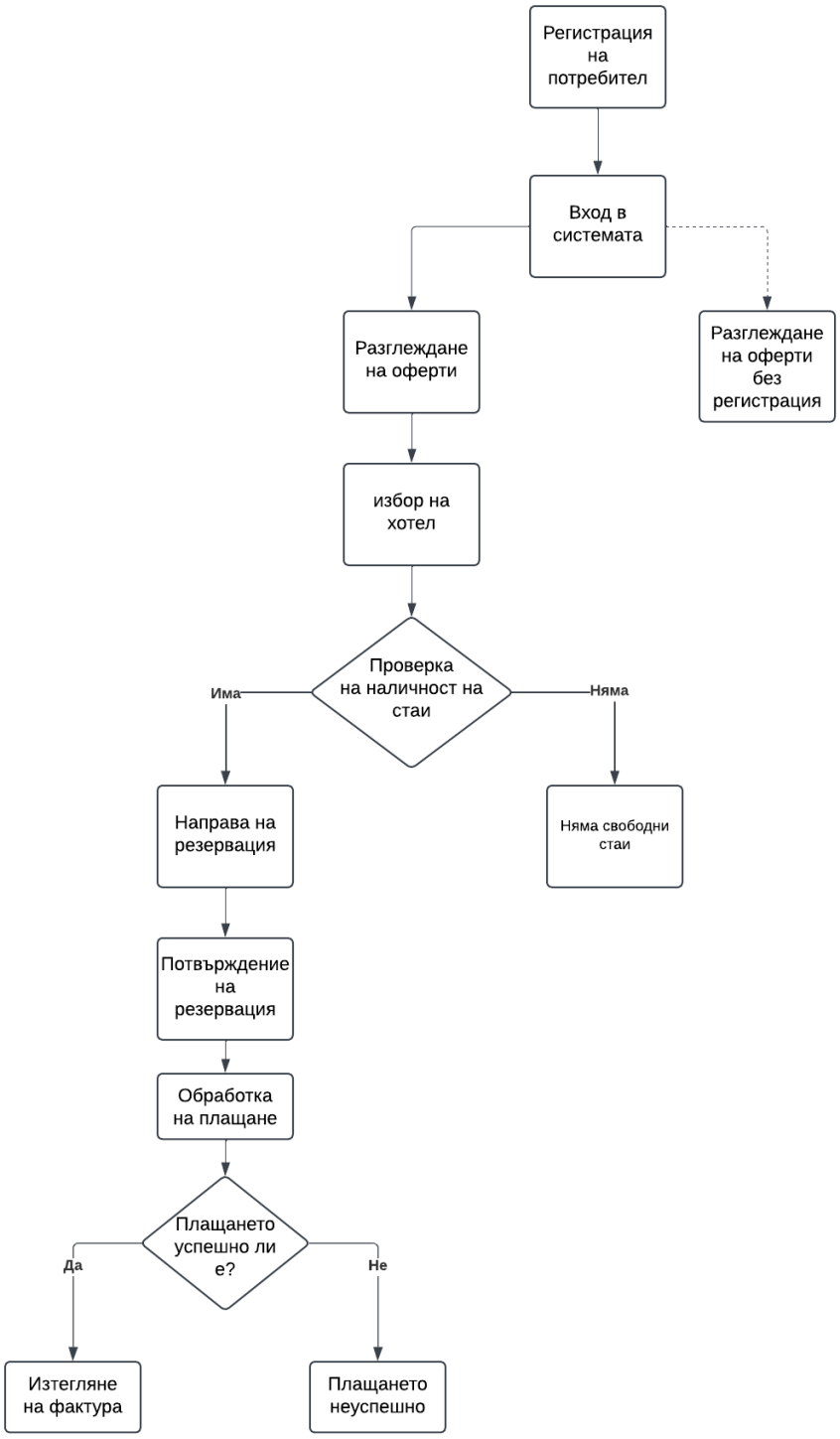



**Диаграма 6:**
Component Abstract Diagram




**Диаграма 7:**
Component Detailed Diagram

# 5.5 Process View:




**Диаграма 8:**
Activity Diagram