Архитектурен проект

"Musical Shop"

Автори:
Александър Костадинов, Денис Табутов, Любослав Балев,

Симона Нешкова, Симона Кирилова

Дата: 05.11.2024 г.

I. Въведение

„Musical shop“ представлява онлайн платформа, предназначена за търговия с музикални инструменти и аксесоари. Проектът цели да улесни процеса на покупка, предоставяйки интуитивен интерфейс, който свързва купувачи с различни производители и доставчици. Този архитектурен проект описва структурата и компонентите на планираната система за онлайн музикален магазин. Той предоставя пълна картина на структурната организация на системата, включително основните компоненти, техните взаимовръзки и начина им на взаимодействие.

Участници:

- **Фронтенд разработчик**: Отговаря за изграждането на потребителския интерфейс, интеграцията на API за взаимодействие с бекенда и оптимизацията на производителността.
- **Бекенд разработчик**: Разработва API за управление на данни и функционалности на приложението, включително обработка на поръчки и плащания.
- **Разработчик на база данни**: Проектира структурата на базата данни и осигурява ефективност, сигурност и надеждност.

- **Софтуерен архитект**: Определя структурата на основните компоненти и техните връзки, избира подходящи технологии, осигурява сигурността и оптимизацията на системата.
- **QA специалист**: Провежда тестове за осигуряване на качеството, включително функционално тестване и ръчно тестване на потребителския интерфейс.

II. Предназначение на архитектурния проект

1. Обхват

Архитектурният проект обхваща всички аспекти на разработката, като включва:

- **Планиране:** Оценка на изискванията и нуждите на системата, определяне на ресурси и времеви рамки за всяка фаза на разработката.
- **Анализ на изискванията:** Определяне на основните функционалности на системата, включително онлайн покупка, интеграция с PayPal и възможност за регистрация чрез Google акаунт.
- **Дизайн:** Избор на архитектура, бази данни и технологии, проектиране на потребителския интерфейс и инфраструктурни нужди.
- **Програмиране:** Разработка на клиентски и сървърни компоненти, включително база данни и API.
- **Тестване:** Функционално, производително и сигурностно тестване на приложението.
- **Внедряване:** Избор на платформа за разгръщане на системата и мониторинг след пускане.

2. Актьори

- **Клиенти на магазина:** Потенциални потребители, които ще използват системата за закупуване на музикални инструменти и аксесоари.

- **Администратори на системата:** Отговорни за управлението на съдържанието и мониторинга на системата.
- **Разработчици и тестери:** Заинтересовани от архитектурния дизайн за правилна имплементация и тестване.

3. Използвани термини и символи

- **UML:** Unified Modeling Language, стандартен език за визуализация на архитектурни дизайни.
- **MVC:** Model-View-Controller, архитектурен шаблон за разделяне на логиката и представянето.
- **ER диаграма:** Entity-Relationship диаграма, използвана за моделиране на бази от данни.
- **API:** Интерфейс за комуникация между клиентската и сървърната част.
- **PayPal Sandbox:** Среда за симулиране на плащания без реални разходи.

III. Архитектурен обзор

1. Use-case изглед

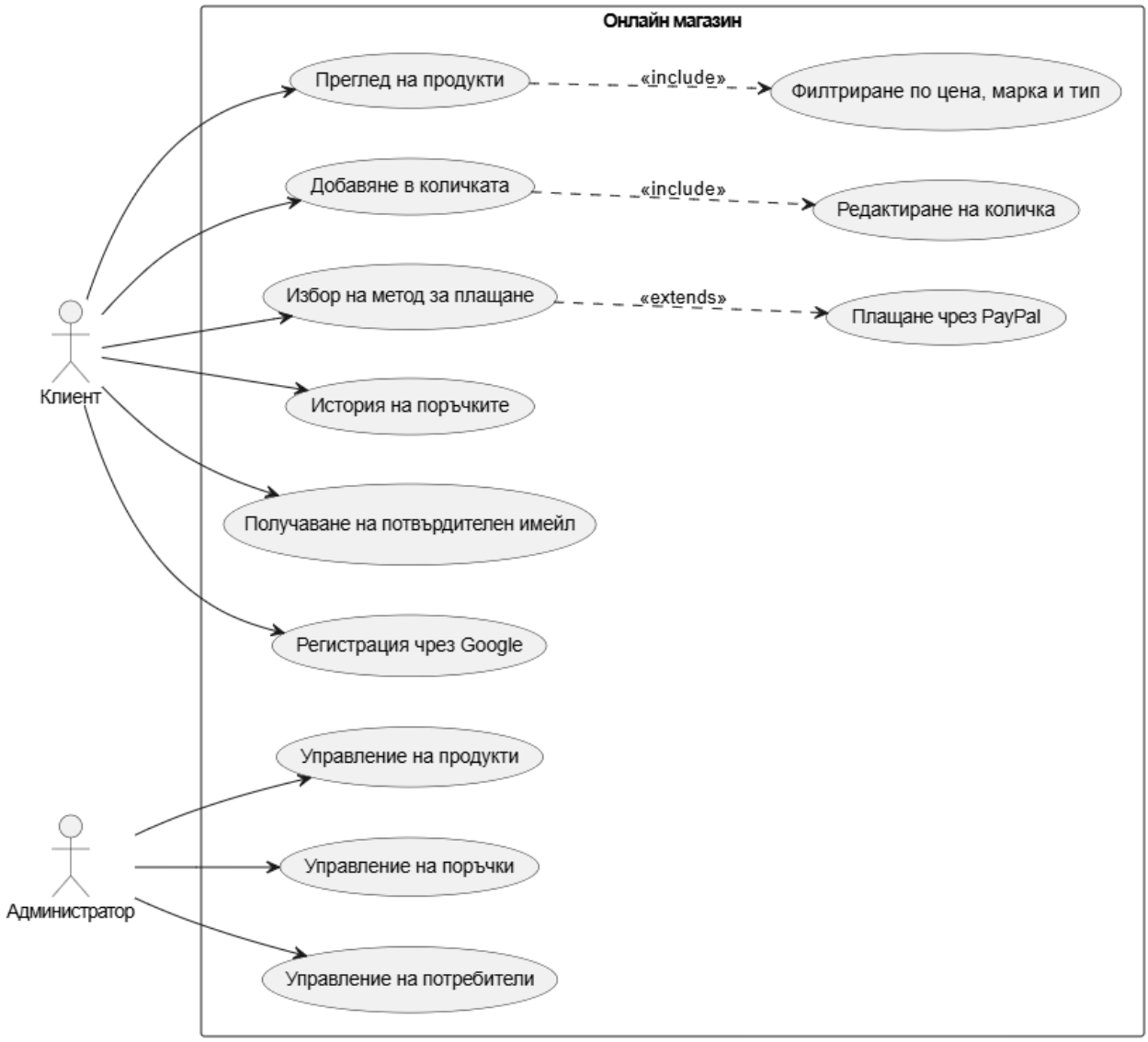
Цел и обхват:
Use-case изгледът дефинира основните взаимодействия между различните видове потребители и системата. Той дава цялостна представа за функционалностите, които системата ще поддържа от гледната точка на потребителите и техните цели.

Основни Use-case сценарии:

- **Клиенти:**

- ***Преглед на продуктите:*** Клиентът може да разглежда наличните продукти в различни категории, включително филтриране по цена, марка и тип на продукта.
- ***Добавяне в количката:*** Клиентът може да добавя избрани продукти в количката и да ги редактира (добавяне, премахване, актуализиране на количество).
- ***Избор на метод за плащане:*** Възможност за избор на метод за плащане, включително PayPal. При избиране на PayPal, потребителят се пренасочва към тестова (sandbox) среда за завършване на плащането.
- ***История на поръчките:*** След завършване на поръчка, клиентът може да прегледа статуса на своите поръчки и историята на предишни покупки.
- ***Получаване на потвърдителен имейл:*** След успешна поръчка, потребителят получава потвърдителен имейл с детайли.
- ***Регистрация чрез Google:*** Потребителите могат да се регистрират и влизат в системата чрез своя Google акаунт

- **Администратори:**
  - ***Управление на продукти:*** Добавяне, премахване и актуализиране на продукти и техните характеристики (категория, марка, наличност).
  - ***Управление на поръчки:*** Следене на текущи поръчки, актуализиране на статус на поръчките (в процес на подготовка, изпратена, доставена).
  - ***Управление на потребители:*** Управление на акаунтите на потребителите, включително деактивиране при нужда.



2. Логически изглед

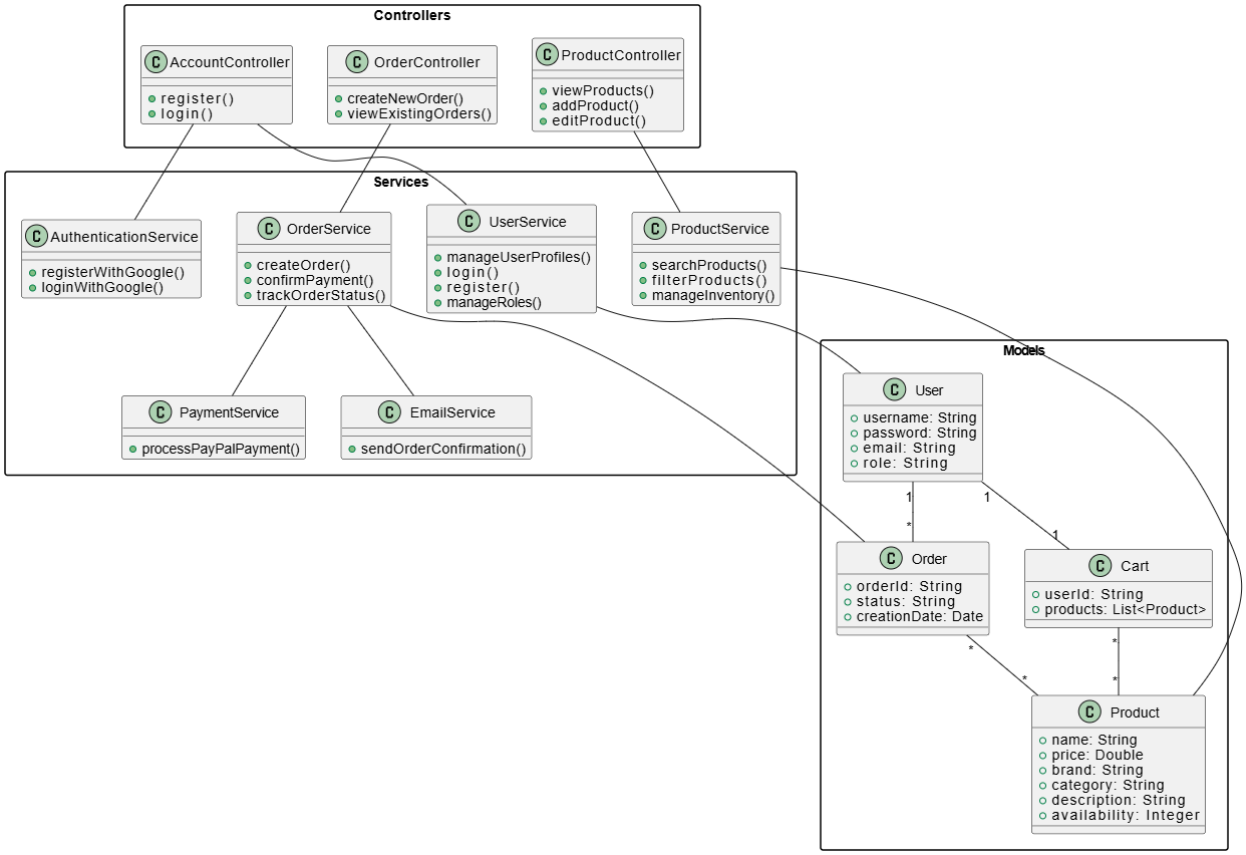
Цел и обхват:
Логическият изглед описва структурните елементи на системата като модули, класове и компоненти и техните зависимости. Този изглед ще даде на разработчиците информация за разпределението на функционалностите и основните слоеве на системата.

Основни компоненти:

- **Модели (Models):**
    - ***Product:*** Представя музикални инструменти и аксесоари. Включва атрибути като име, цена, марка, категория, описание и наличност.
    - ***User:*** Информация за клиентите и администраторите, като потребителско име, парола, имейл и роля.
    - ***Order:*** Поръчка, създадена от клиента. Включва ID на поръчката, статус (създадена, в процес на обработка, доставена) и дата на създаване.
    - ***Chart:*** Представя текущите продукти в количката на потребителя, които не са завършени като поръчка.
- **Services:**
    - ***ProductService:*** Съдържа логиката за управление на продуктите (като търсене, филтриране и управление на наличност).
    - ***OrderService:*** Справя се с бизнес логиката за създаване на поръчки, потвърждаване на плащания и проследяване на статуса.
    - ***UserService:*** Управление на потребителски профили, влизане, регистрация и роли (клиент или администратор).
    - ***PaymentService:*** Интеграция с PayPal за обработка на плащания, конфигуриран за sandbox среда.
    - ***EmailService:*** Отговаря за изпращането на потвърдителни имейли при завършена поръчка, използвайки шаблони и детайли за поръчката.1
    - ***AuthenticationService:*** Обработва регистрацията и входа чрез Google

- **Контролери (Controllers):**
    - ***ProductController:*** Обработва заявки, свързани с продукти, като преглед, добавяне и редактиране.

- ***OrderController:*** Обработва заявки, свързани с поръчките, включително създаване на нова поръчка и преглед на съществуващи поръчки.
- ***AccountController:*** Справя се с функционалности, свързани с акаунтите на потребителите, като регистрация и вход.



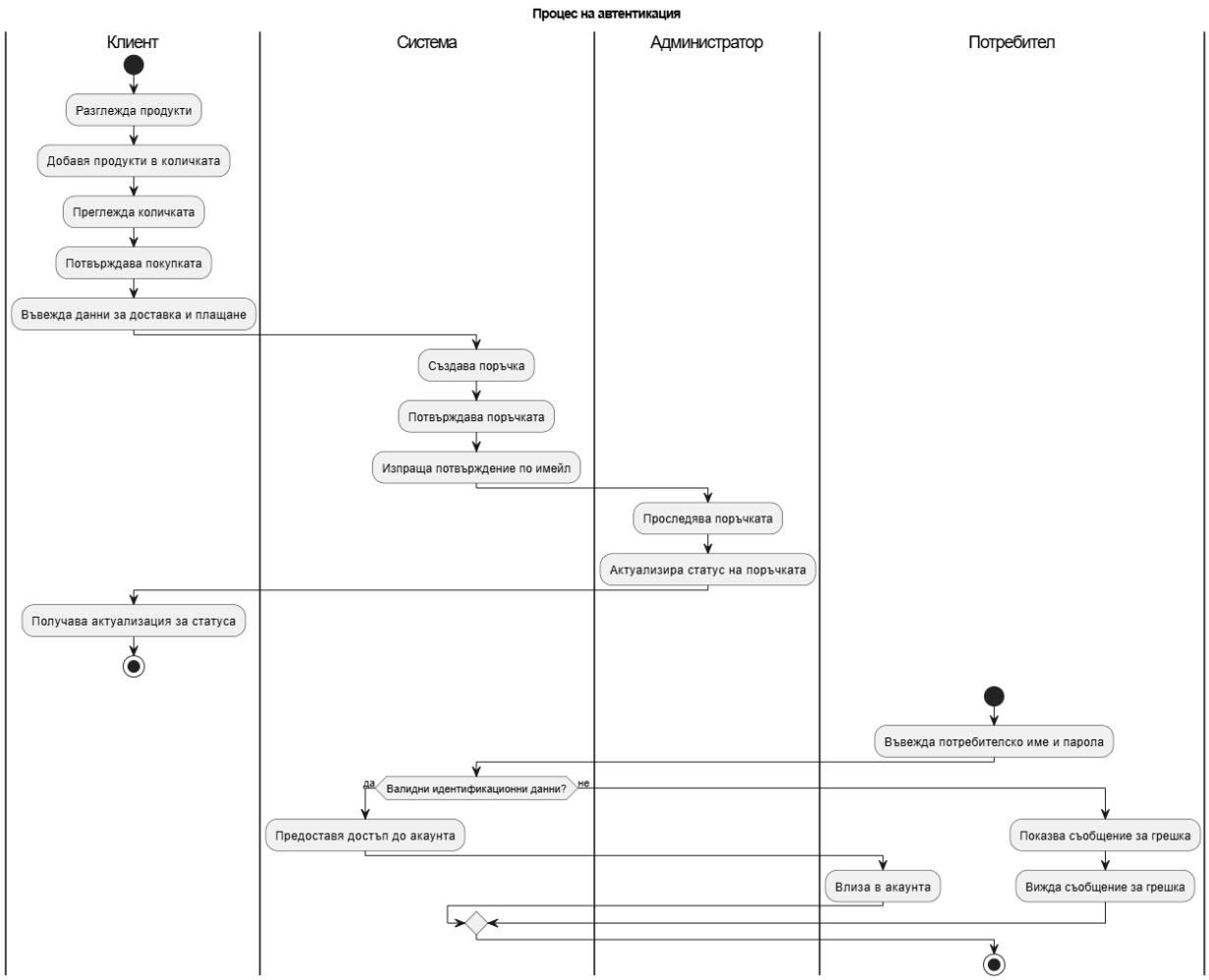
3. Процесен изглед

Цел и обхват:
Процесният изглед показва потока на информация и последователността на действията в системата, както и взаимодействието между компонентите в реализацията на ключови функционалности.

Основни процеси:

- **Процес на поръчка:**
    - Клиентът добавя продукти в количката.
    - Преминава към преглед на количката и потвърждава покупката.
    - Въвежда данни за доставка и плащане, след което поръчката се създава и потвърждава.
    - Администраторът може да проследи поръчката и да актуализира статуса ѝ до изпратена или доставена.

- **Процес на автентикация:**
    - Потребителят въвежда потребителско име и парола.
    - Системата проверява идентификационните данни в базата данни.
    - Ако данните са коректни, потребителят получава достъп до своя акаунт, а ако не са – получава съобщение за грешка.



4. Изглед на данните

Цел и обхват:
Изгледът на данните се фокусира върху структурата на базата от данни и логическите връзки между нейните таблици. Той ще осигури ясна представа за това как ще бъдат съхранявани и свързани данните в системата.

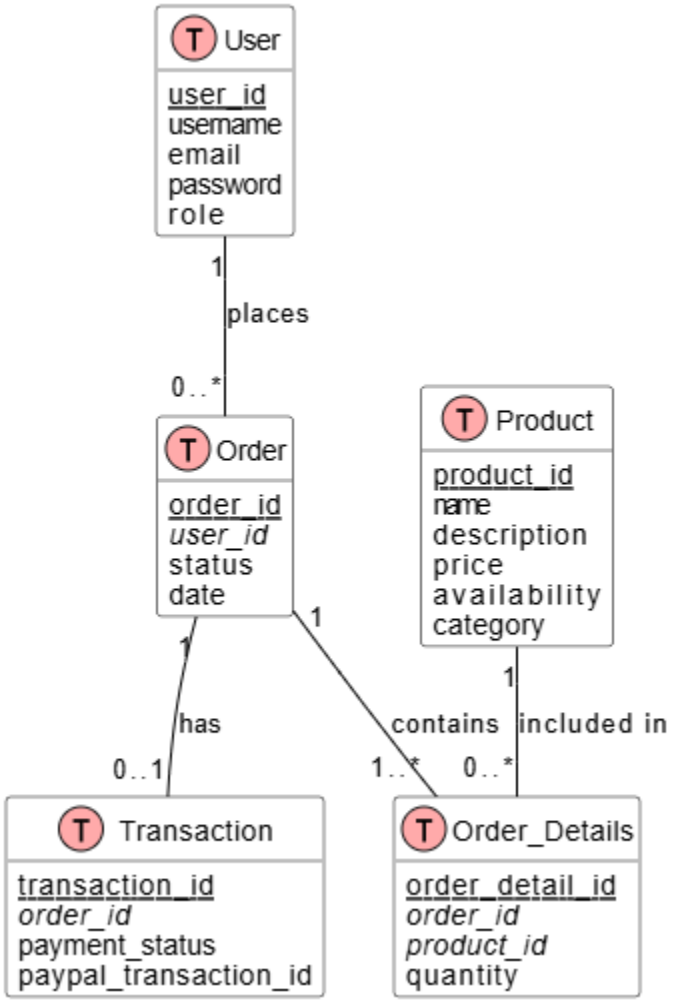
Основни таблици:

- **User (Потребител):** Съхранява информация за клиентите и администраторите, включително уникален идентификатор, потребителско име, имейл и роля.

- **Product (Продукт):** Представя артикули в магазина с уникален идентификатор, име, описание, цена, наличност и категория.
- **Order (Поръчка):** Съхранява информация за поръчките на клиентите, като ID на поръчката, потребителски ID, статус и дата.
- **Order_Details (Детайли на поръчка):** Мостова таблица между поръчките и продуктите, която съхранява всеки продукт, поръчан в рамките на определена поръчка, заедно с количеството.
- **Transaction:** Съхранява информация за транзакции, извършени чрез PayPal, включително ID на транзакцията и статус на плащането.

Релации:

- **1-M релация** между **User** и **Order** (един потребител може да има много поръчки).
- **M-M релация** между **Order** и **Product** чрез **Order_Details**.



5. Изглед на внедряването

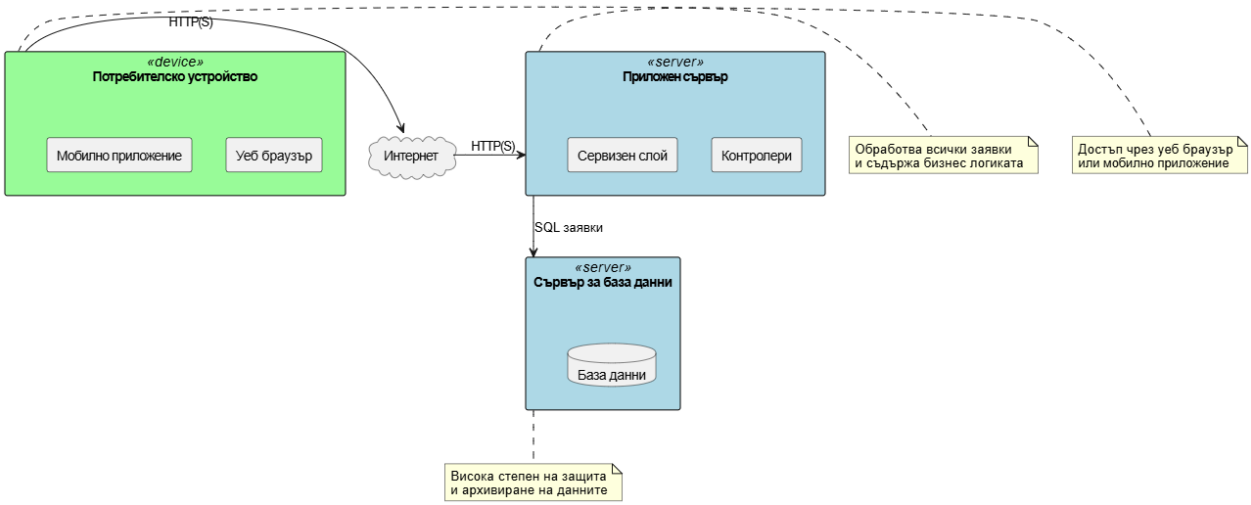
Цел и обхват:
Изгледът на внедряването представя разпределението на софтуера върху хардуерни ресурси и мрежовата архитектура, която ще се използва. Този изглед осигурява на системните администратори информация за нужната инфраструктура.

Инфраструктурни компоненти:

- **Потребителско устройство:** Достъпът до системата ще се осъществява чрез уеб браузър или мобилно приложение.
- **Приложен сървър:** Отговаря за бизнес логиката и обработва всички заявки от потребителите. Тук ще се намират контролерите и сервизните слоеве.
- **Сървър за базата данни:** Съхранява всички данни за потребителите, продуктите и поръчките. Изисква се висока степен на защита и архивиране на данните.

Комуникации:

- **HTTP(S)** протокол за сигурна комуникация между потребителските устройства и приложния сървър.
- **SQL** заявки между приложния сървър и базата данни за достъп и актуализация на данни.

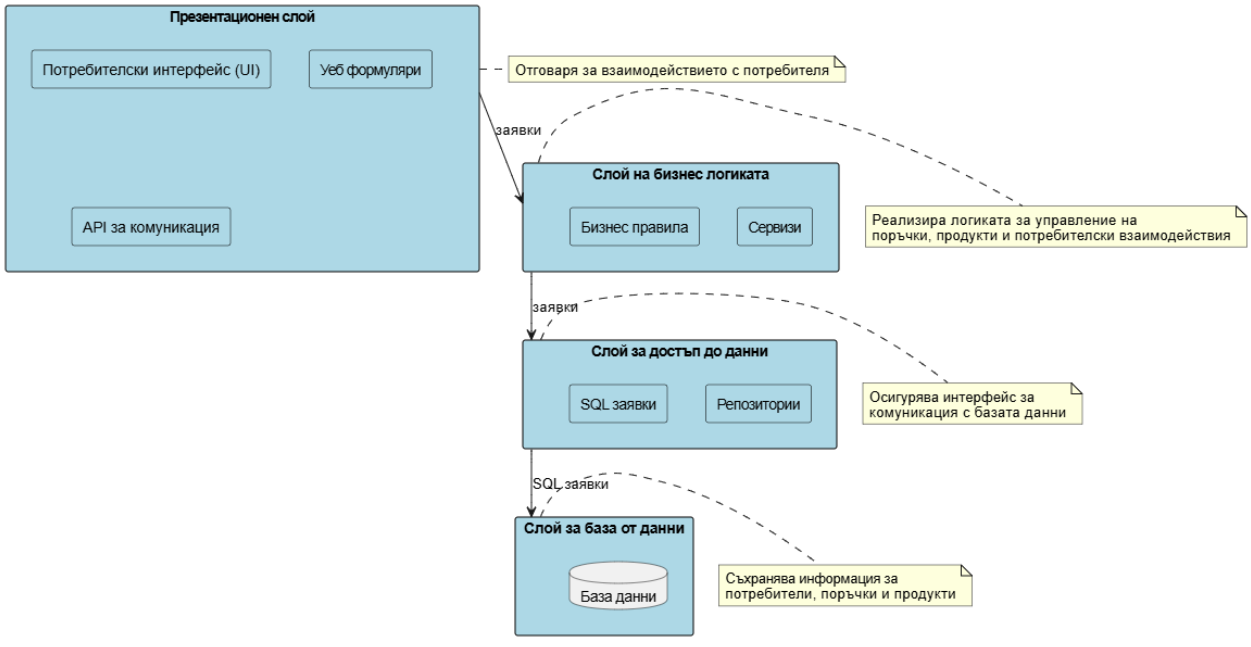


6. Изглед на имплементацията

Цел и обхват:
Изгледът на имплементацията описва детайли относно реализацията на системата, включително използваните технологии и стандарти за кодиране. Този изглед ще даде представа на разработчиците за подхода към имплементация и организационната структура на кода.

Основни слоеве на архитектурата:

- **Презентационен слой (Presentation Layer)**: Отговаря за взаимодействието с потребителя, включително потребителския интерфейс (UI), уеб формулярите и API за комуникация, които предават заявки към бизнес логиката.
- **Слой на бизнес логиката (Business Logic Layer)**: Съдържа сервизите и бизнес правилата, които реализират логиката за управление на поръчките, продуктите и потребителските взаимодействия.
- **Слой за достъп до данни (Data Access Layer)**: Включва репозитории и SQL заявки, които осигуряват интерфейс за комуникация с базата данни.
- **Слой за база от данни (Database Layer)**: Съдържа самата база данни, в която се съхранява информация за потребителите, поръчките и продуктите.



IV. Нефункционални изисквания

1. **Достъпност**:

    - Системата трябва да е достъпна за потребителите през стандартни уеб браузъри, с планирано време за поддръжка и актуализации.
    - Основните функции трябва да бъдат налични дори при частични откази, а логическите модули на системата трябва да бъдат лесни за архивиране, с периодично експортиране на данните в архивни файлове.

2. **Разширяемост**:

    - Системата ще е проектирана с архитектура, която позволява добавяне на нови модули или функционалности.
    - Структурата на проектните модули ще позволява лесно добавяне на нови контролери и услуги без значителни промени в основния код.
    - Ще се поддържа се ясна слоеста архитектура, като всяка нова функционалност се имплементира в подходящия слой, например в контролери или сервизи.

3. **Производителност**:

    - Системата трябва да поддържа плавно и отзивчиво функциониране при стандартно натоварване. Това се постига чрез ефективно управление на заявки към базата данни и оптимизация на зареждането на данни.
    - Често използваните данни ще се кешират в рамките на сесията на потребителя, за да се намалят повторните заявки към сървъра.

4. **Сигурност**:

- Системата трябва да осигурява основна защита за потребителските данни, включително ограничаване на достъпа до административните функции въз основа на роля, хеширане на паролите в базата данни и защита срещу SQL инжекции чрез параметризирани заявки.

5. **Възможност за тестване**:

   - Системата трябва да позволява проверка на основните функционалности чрез модулно и ръчно тестване. Логическите компоненти ще са разделени, за да могат да бъдат тествани самостоятелно (например контролери, модели и услуги).
   - Проектът е организиран така, че лесно да се внедряват модулни тестове при бъдещи разширения.

6. **Използваемост**:

   - Потребителският интерфейс на системата трябва да бъде интуитивен и лесен за навигация.
   - Елементите на интерфейса да са ясно обозначени и лесно достъпни, като се осигурява бърз достъп до основни функции като преглед на продукти и добавяне в количката.