Документация на проект "SEBIZ"

1. Описание на учебния проект

Проектът представлява уеб платформа за дистрибуция и игра на дигитални игри. Основната му цел е да предостави на потребителите възможност да закупуват и играят игри директно в своя уеб браузър, без необходимост от инсталация. Платформата позволява на разработчици да качват своите игри, които след това стават достъпни в общия каталог. Системата включва модул за управление на потребителски акаунти, лична библиотека с игри за всеки потребител и система за препоръки, която предлага нови заглавия въз основа на предишните покупки и предпочитания.

2. Архитектура и дизайн на системата

Системата е изградена на базата на модерна многослойна архитектура, която разделя отговорностите и осигурява гъвкавост и лесна поддръжка.

- **Бекенд (Backend):** Сървърната част е разработена с **ASP.NET Core Web API**. Тя следва ясна архитектурна подредба, разделена на следните слоеве:

    - **Controllers:** Приемат HTTP заявки и управляват потока на данни.
    - **Services:** Съдържат основната бизнес логика на приложението.
    - **Domain:** Дефинира моделите на данните (Data Models) и договорите за трансфер на данни (Data Transfer Objects - DTOs).
    - **Data:** Отговаря за комуникацията с базата данни.

- **Фронтенд (Frontend):** Клиентската част е едностранично приложение (Single Page Application - SPA), изградено с библиотеката **React** и инструментариума **Vite**. То предоставя динамичен и интуитивен потребителски интерфейс.

- **База данни (Database):** За съхранение на данните се използва **MongoDB** - NoSQL документно-ориентирана база данни. Това решение осигурява гъвкавост при работа с разнообразни структури от данни, каквито са игрите и потребителските профили.

- **Комуникация:** Връзката между фронтенд и бекенд се осъществява чрез **RESTful API**. Бекенът предоставя API крайни точки (endpoints), които фронтендът извиква, за да изпраща и получава данни в JSON формат.

- **Съхранение на файлове:** Изображенията към игрите, както и самите файлове на игрите (експортнати от Godot Engine), се съхраняват директно на **локалната файлова система** на сървъра.

3. Описание на е-бизнеса

Бизнес моделът на платформата е фокусиран върху създаването на отворена екосистема за независими разработчици и геймъри.

- **Основна концепция:** Платформата функционира като дигитален магазин, в който всеки регистриран потребител може да качва и предлага за продажба свои собствени игри, стига те да са съвместими с уеб среда (експортнати от Godot Engine).

- **Трансакции:** На настоящия етап на разработка, трансакциите се извършват директно между купувача и създателя на играта, като платформата не удържа комисионна. Това насърчава участието и растежа на общността.

- **Модел на приходите (бъдещо развитие):** Предвижда се възможност за бъдеща монетизация чрез въвеждане на малък процент комисионна от всяка продажба, реализирана през платформата. Други потенциални източници на приходи могат да бъдат абоментри планове с допълнителни функционалности за разработчици или платени промоционални позиции в каталога.

- **Целева аудитория:** Проектът е насочен както към независими разработчици на игри, които търсят лесен начин да дистрибутират своите продукти, така и към геймъри, които искат да откриват нови и интересни заглавия и да ги играят моментално, без инсталация.

4. Описание на Базата от Данни

Проектът използва **MongoDB** като система за управление на бази от данни. Изборът на NoSQL база данни е продиктуван от необходимостта от гъвкава схема, която лесно може да се развива и променя.

Базата данни се състои от четири основни колекции: `Users`, `Games`, `Transactions` и `GameUsage`.

Колекция `Users`

Тази колекция съхранява информация за регистрираните потребители. Всеки документ в колекцията има следната структура:

- `_id` (ObjectId): Уникален идентификатор на документа.
- `Username` (String): Потребителско име.
- `Email` (String): Електронна пощта на потребителя.
- `PasswordHash` (String): Хеширана парола на потребителя.
- `OwnedGamesIds` (Array of Strings): Списък с идентификаторите (`_id`) на игрите, които потребителят е закупил.
- `Balance` (Double): Баланс на счета на потребителя в платформата (в евро).

- `Role` (String): Роля на потребителя ("User" или "Admin").

Колекция `Games`

Тази колекция съдържа информация за всички игри, налични в платформата. Всеки документ има следната структура:

- `_id` (ObjectId): Уникален идентификатор на играта.
- `Name` (String): Име на играта.
- `Description` (String): Описание на играта.
- `Price` (Double): Цена на играта.
- `Genre` (String): Жанр на играта (напр. "Екшън", "Стратегия").
- `Developer` (String): Име на разработчика или студиото.
- `ReleaseDate` (DateTime): Дата на издаване.
- `Tags` (Array of Strings): Етикети, описващи играта.
- `CreatedById` (String): ID на потребителя, който е качил играта.
- `ImagePath` (String): Път до изображението на играта във файловата система.
- `GameFilePath` (String): Път до файла на играта (.pck, .wasm и други Godot експортирани файлове).

Колекция `Transactions`

Тази колекция съхранява всички финансови трансакции - покупки и депозити на средства. Всеки документ в колекцията има следната структура:

- `_id` (ObjectId): Уникален идентификатор на трансакцията.
- `UserId` (String): ID на потребителя, който е извършил трансакцията.
- `GameId` (String): ID на закупената игра (може да е null при депозит).
- `Amount` (Double): Сума на трансакцията в евро.
- `Timestamp` (DateTime): Дата и време на трансакцията.
- `Type` (String): Тип трансакция ("Purchase" за покупка или "Deposit" за депозит).
- `Status` (String): Статус на трансакцията ("Completed", "Pending", "Failed").
- `Description` (String): Описание на трансакцията.

Колекция `GameUsage`

Тази колекция проследява как потребителите користат игрите. Всеки документ съхранява информация за сесия на игра:

- `_id` (ObjectId): Уникален идентификатор на записа.
- `UserId` (String): ID на потребителя.
- `GameId` (String): ID на играта.
- `StartTime` (DateTime): Време на началото на сесията.
- `EndTime` (DateTime): Време на края на сесията.

- `DurationMinutes` (Integer): Продължителност на сесията в минути.
- `SessionCount` (Integer): Брой пъти, че потребителят е пускал тази игра.
- `LastPlayedDate` (DateTime): Дата на последния път, когато игра е била пущена.

Тази колекция се използва за:

- Събиране на данни за употреба на игри
- Подпомагане на системата за препоръки (анализиране на жанровете, които потребителят предпочита)
- Аналитика и отчетност

Връзки между колекциите

Връзките между колекциите са следните:

- **Users ↔ Games:** Връзка от тип "много към много" - един потребител може да притежава много игри, и една игра може да бъде закупена от много потребители. Реализира се чрез масива `OwnedGamesIds` в документите на `Users`.
- **Users ↔ Transactions:** Връзка от тип "един към много" - един потребител може да има много трансакции.
- **Games ↔ Transactions:** Връзка от тип "един към много" - една игра може да бъде закупена много пъти (много трансакции се отнасят до една игра).
- **Users ↔ GameUsage:** Връзка от тип "един към много" - един потребител има много записи за употреба.
- **Games ↔ GameUsage:** Връзка от тип "един към много" - една игра може да бъде играна от много потребители.

5. Описание на категориите и отделите

В контекста на проекта, категоризацията на игрите е опростена и се базира основно на полето `Genre` (Жанр) в колекцията `Games`. Липсва по-сложна йерархична структура като "отдели" и "подкатегории", което е подходящо за началния етап на проекта.

Каталог и категоризация

- **Основна категория:** Жанрът на играта (`Action`, `RPG`, `Strategy` и т.н.) служи като основен филтър и категория за търсене и организиране на каталога. Потребителите могат да разглеждат игри, групирани по жанр.
- **Търсене:** Системата позволява търсене по име, жанр, тагове и други ключови характеристики на игрите, което улеснява намирането на желаното съдържание.

Администриране на каталога

Управлението на каталога (добавяне, редактиране и изтриване на игри) се осъществява чрез RESTful API ендпойнтите, предоставени от бекенда. Тези операции

са достъпни за потребители със съответните права (в бъдеще може да се въведе ролева система).

Основните API ендпойнти за управление на каталога са:

- **Създаване на игра (POST /api/Game):** Позволява добавянето на нова игра в базата данни.
- **Получаване на всички игри (GET /api/Game):** Извлича списък с всички налични игри, който се визуализира в каталога.
- **Получаване на конкретна игра (GET /api/Game/{id}):** Извлича детайлна информация за една игра.
- **Редактиране на игра (PUT /api/Game/{id}):** Позволява обновяване на информацията за съществуваща игра.
- **Изтриване на игра (DELETE /api/Game/{id}):** Премахва игра от каталога.

6. Описание на съхранените процедури

Проектът използва MongoDB, която е NoSQL база данни. За разлика от релационните бази данни (като SQL Server, MySQL), MongoDB не поддържа съхранени процедури (stored procedures) в традиционния им вид.

Цялата бизнес логика, включително валидации, изчисления и операции с данни, е имплементирана в **service слоя (Service Layer)** на ASP.NET Core приложението. Този подход има следните предимства:

- **Централизация на логиката:** Бизнес правилата са на едно място, което улеснява тяхната поддръжка и модификация.
- **Независимост от базата данни:** Приложението не е силно обвързано с конкретната имплементация на базата данни.
- **По-лесно тестване:** Бизнес логиката може да бъде тествана изолирано от базата данни чрез unit тестове.

7. Модул за потребителски сметки и обработка на поръчки

Модул за потребителски сметки

Системата разполага с базов модул за управление на потребителските акаунти, който предоставя основните функционалности:

- **Регистрация (POST /api/User/register):** Позволява на нови потребители да създадат акаунт, предоставяйки потребителско име и парола.
- **Вход (POST /api/User/login):** Автентикира потребителите и им предоставя достъп до системата.
- **Сигурност на паролите:** Паролите на потребителите се съхраняват в хеширан вид в базата данни. Използваният алгоритъм е SHA256.

- **Забележка:** За реално приложение се препоръчва използването на по-сигурни и модерни алгоритми за хеширане като **BCrypt** или **Argon2**, които са по-устойчиви на "brute-force" атаки.

Обработка на потребителски поръчки

На текущия етап, процесът на "поръчка" включва използването на количка за пазаруване, където потребителите могат да добавят множество игри преди финализиране на покупката.

- **Количка за пазаруване (Shopping Cart):**

  - Потребителите могат да добавят игри в количката си директно от каталога или страницата с детайли на играта.
  - Количката се съхранява локално в браузъра на потребителя (localStorage), което позволява запазване на съдържанието между сесиите.
  - Функционалности на количката:
    - Преглед на добавените игри с изображения, имена и цени.
    - Премахване на отделни игри от количката.
    - Изчисляване на обща сума за всички елементи.
    - Изчистване на цялата количка.
  - Количката е имплементирана като React Context (CartContext), който управлява състоянието на приложението и синхронизира с localStorage.

- **Конвейер на поръчката:**

  1. Потребителят добавя желани игри в количката.
  2. От страницата на количката, потребителят може да прегледа всички елементи и общата сума.
  3. Натиска бутон "Purchase All" за закупуване на всички игри в количката.
  4. Системата проверява баланса на потребителя и извършва покупките последователно за всяка игра.
  5. При успешна покупка, игрите се добавят към библиотеката на потребителя (масив `OwnedGamesIds`), а количката се изчиства.
  6. Потребителят се пренасочва към страницата "Моите игри" за достъп до закупените заглавия.

- **Обработка на грешки:** Ако някоя покупка се провали (например поради недостатъчен баланс или технически проблеми), системата продължава с останалите и информира потребителя за неуспешните елементи.

Оторизиране на кредитна карта (Бъдеща функционалност)

Пълноценна система за обработка на плащания и оторизиране на кредитни карти **не е имплементирана** на този етап, но архитектурата позволява нейното бъдещо добавяне.

- **Предложен модел за развитие:**
    1. **Интеграция с платежен оператор (Payment Gateway):** Избор и интеграция на външна услуга като **Stripe** или **PayPal**. Тези платформи предоставят сигурни и готови решения за обработка на картови данни, свеждайки до минимум рисковете за сигурността.
    2. **Промяна в конвейера на поръчката:**
        - След стъпка 2 (натискане на "Купи"), потребителят ще бъде пренасочен към формата за плащане на избрания оператор.
        - След успешно плащане, платежният оператор ще изпрати потвърждение (callback/webhook) до бекенда на нашата система.
        - Едва след получаване на това потвърждение, бекендът ще добави играта към библиотеката на потребителя.
    3. **Сигурност:** По този начин чувствителна информация като номера на кредитни карти няма да се съхранява и обработва от нашата система, а от специализиран и сертифициран доставчик на платежни услуги.

8. Иновативност на предложения модел

Иновативността на проекта се проявява в няколко ключови аспекта, които го отличават от стандартните платформи за дистрибуция на игри и предлагат нов бизнес модел.

Интеграция с Godot Engine и "Instant Play"

Основният иновативен елемент е възможността за **незабавно стартиране и игра на игрите директно в уеб браузъра**.

- **Техническа реализация:** Това е постигнато чрез интеграция с **Godot Engine** и неговата възможност за експорт на проекти към **HTML5 (WebAssembly)**. Фронтендът на приложението (React) динамично зарежда необходимите файлове на играта (`.js`, `.wasm`, `.pck`) и инициализира Godot енджина в специално предназначен за това HTML `<canvas>` елемент. По този начин се елиминира нуждата потребителят да инсталира каквото и да е на своето устройство.
- **Промяна на бизнес модела:** Тази функционалност променя съществуващия модел на дистрибуция, който обикновено изисква сваляне и инсталация на играта. Моделът "Instant Play" е изключително удобен за потребителите, позволява им да изпробват игри веднага и премахва бариери пред покупката. Той е особено подходящ за независими и "casual" игри.

Персонализирана система за препоръки

Вторият иновативен аспект е вградената система за препоръки, която цели да подобри потребителското изживяване и да увеличи продажбите.

- **Алгоритъм:** Системата анализира библиотеката с игри на всеки потребител. Тя идентифицира **най-често срещаните жанрове** в нея и на тази база предлага други игри от същите жанрове, които потребителят все още не притежава.

- **Нова стойност:** За разлика от стандартните магазини, които често разчитат на общи класации или платени промоции, тази система предлага **високо персонализирано съдържание**. Това помага на потребителите да откриват нови игри, които отговарят на техните вкусове, и същевременно подпомага по-малко известни разработчици, чиито игри биха се вписали в предпочитанията на конкретен потребител.

Отворена екосистема за разработчици

Предложеният модел насърчава създаването на **отворена и достъпна екосистема**, което само по себе си е промяна спрямо по-затворените платформи. Като позволява на всеки да качва своите уеб-базирани игри и първоначално не удържа комисионна, проектът стимулира креативността и улеснява навлизането на нови разработчици на пазара.

9. Интеграция с други системи

Архитектурата на проекта е проектирана с мисъл за гъвкавост и бъдещо разширение, което включва интеграция с разнообразни външни системи.

Настоящи интеграции

- **MongoDB (Отдалечена база данни):** Проектът е конфигуриран да работи с отдалечена инстанция на MongoDB. Връзката се осъществява чрез Connection String, което позволява лесна промяна на хостинг средата на базата данни.
- **Godot Engine (HTML5 Export):** Ключова интеграция, която позволява изпълнението на игри в браузъра. Платформата не комуникира директно с Godot Engine, а със статичните файлове, генерирани от него.

Необходимост от бъдещи интеграции

За да се превърне в пълноценна комерсиална платформа, проектът има необходимост от интеграция със следните типове системи:

- **Платежни оператори (Payment Gateways):**
    - **Необходимост:** Автоматизиране на процеса на продажба, сигурно обработване на плащания и елиминиране на нуждата от ръчни трансакции.
    - **Реализация:** Интеграция с API на услуги като **Stripe** или **PayPal**. Това ще позволи на потребителите да плащат с кредитни/дебитни карти директно в платформата, а на разработчиците - автоматично да получават приходите си.
- **Системи за имейл маркетинг (Email Services):**
    - **Необходимост:** Подобряване на комуникацията с потребителите - изпращане на потвърждения за регистрация, известия за нови игри, промоционални кампании и възстановяване на забравени пароли.

    - **Реализация:** Интеграция с услуги като **SendGrid**, **Mailgun** или други SMTP сървъри.
    - **Пример за имейл функционалност:**
        - **Приветствено писмо:** След регистрация на нов потребител, системата автоматично изпраща приветствено писмо на неговата електронна пощта.
        - **Потвърждение на покупка:** След успешна покупка на игра, потребителят получава имейл с потвърждение, съдържащ детайлите на трансакцията.
        - **Известие при депозит:** Когато потребител добави средства към своя баланс, получава имейл потвърждение.
        - **Препоръки:** Периодично, системата може да изпраща имейли с препоръчани игри въз основа на предпочитанията на потребителя.
        - **Уведомления:** Имейл известия при важни събития (разпродажба, нова игра в предпочитана категория, профил актуализирана и т.н.).

- **Социални мрежи (Social Logins):**

    - **Необходимост:** Улесняване на процеса на регистрация и вход за потребителите.
    - **Реализация:** Интеграция с OAuth 2.0 протоколите на платформи като **Google**, **Facebook**, **GitHub** и др. Това ще позволи на потребителите да създават акаунт и да влизат в системата с едно кликване.

- **Системи за аналитики (Analytics Services):**

    - **Необходимост:** Събиране на данни за поведението на потребителите - кои игри се разглеждат най-много, колко време се играят, какъв е процентът на успешни покупки.
    - **Реализация:** Интеграция с инструменти като **Google Analytics** или **Mixpanel** чрез добавяне на техните скриптове за проследяване във фронтенд приложението.