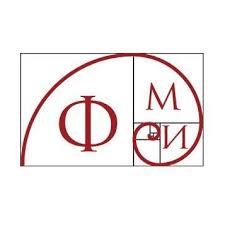
Софийски университет “Св. Климент Охридски”

Факултет по математика и информатика

Курсов проект по “Системи за електронен бизнес”

зимен семестър 2024/2025

Проект – Система за тийм-билдинги и събития

Изработил: Мария Маргаритова

Специалност: Информационни системи

Курс: 4

Гр. София

Декември 2024

**Съдържание**

1. Описание на учебния проект
2. Архитектура и дизайн на системата
3. Описание на е-бизнеса
4. Описание със стандартни методи на функционалността, обектите и потребителски случаи, които ще се реализират.
5. Описание на Базата от Данни
6. Описание на категориите и отделите
7. Реализация на потребителската кошница

* Редактиране и показване съдържание
* Реализиране и показване на обработване

1. UML диаграми
2. Описание на съхранените процедури
3. Модул за потребителските сметки, усъвършенствано обработване на потребителските поръчки, конвейер за обработка на поръчките, включващ оторизиране на кредитната карта.
4. Описание на х-ките за иновативност на предложения модел -разработване на нов бизнес модел или промяна на съществуващ
5. Описание на необходимостта от и реализация на свързване-интеграция с други системи
6. RUP model
7. Демо сървър
8. Стартиране на приложението
9. ***Описание на учебния проект***

Поради забързаното ежедневие и невъзможността сами да организираме събития, празненства и изненади за близките си, ние често търсим варианти, за да направим това чрез някаква платформа. Все по често фирмите правят тиймбилдинги, за да сплотяват колектива, организират се конференции, частни партита и тн. Също така и частните лица търсят начини да изненадат близките си с някакво преживяване. Точно заради това се създаде нуждата от проектът **Adventure Teambuilding Platform.** Той представлява платформа за електронна търговия, предназначена за закупуване на тиймбилдинг услуги, събития, частни партита, корпоративни мероприятия, изненада за любимия човек и подобни. Всички тези услуги са събрани на едно място – именно в нашата платформа като тази централизирана система, помага на потребителя да направи най-добрия избор за повода си! Тя предоставя възможност на потребителите да разглеждат и търсят различни услуги, организирани по категории и отдели, като същевременно предлага разнообразни функционалности като потребителска количка, обработка на поръчки, плащания чрез Stripe и история на поръчките.

1. ***Архитектура и дизайн на системата***

Системата следва **трислойна архитектура** (Three-Tier Architecture), която разделя компонентите на **потребителски интерфейс (UI)**, **бизнес логика (Backend)** и **база данни (Database)**.

За разработката на системата се използват съвременни технологии от Python Flask света, които предоставят добра гъвкавост и лесна поддръжка на проекта. За всеки слой от архитектура се използват следните технологии описани отдолу:

1. **Бизнес логика (Backend):**
   * **Flask**: Лек и мощен Python фреймуърк за изграждане на RESTful API и обработка на потребителски заявки. Предлага модулна структура, която позволява лесно разделяне на логиката на приложението. В този проект Flask обработва потребителските заявки, като валидира входните данни, извършва операции върху базата данни и връща отговори в JSON формат.
2. **Потребителски интерфейс (Frontend):**
   * **HTML/CSS**: HTML предоставя структурата на уеб страниците, докато CSS определя визуалния стил и оформление. В проекта те са използвани за създаване на удобен и интуитивен интерфейс за управление на продуктите, количката и поръчките.
   * **JavaScript**:
     + SweetAlert2: Библиотека за създаване на визуално привлекателни известия, като потвърждения, грешки или успехи. Използва се за подобряване на потребителското изживяване, като показва известия при добавяне на продукти в количката, плащания и други действия.
     + Fetch API: Асинхронен API за извършване на заявки към backend услугите. В този проект Fetch API позволява на фронтенда да комуникира с backend-а за извличане и изпращане на данни (напр. добавяне на продукти в количката, обработка на плащания).
3. **База данни:**
   * **Flask-Migrate**: Flask-Migrate е разширение, което използва Alembic за управление на миграции в SQLAlchemy. То осигурява лесен начин за прилагане на промени в структурата на базата данни, като добавяне на нови таблици или промяна на колони, без да се губят данните.
   * **SQLite**: Лека релационна база данни, подходяща за учебни цели и локално тестване. Тя съхранява данни за потребителите, продуктите, количките и поръчките. Релационният модел осигурява лесна поддръжка, разширяемост и ефективно управление на данните.
   * Базата данни е структурирана с релационен модел, позволяващ лесно разширяване и поддръжка.

***Релационен модел на базата данни:***

* + Базата данни е структурирана с отношения между таблици (напр. User, Product, CartItem, Order, OrderItem, Department, Category), което позволява интегрирана обработка на данни и лесна реализация на сложни заявки.

1. **Инфраструктура**:
   * **Python**: Основният език за разработка. Python е избран заради своята простота, мощност и богата екосистема от библиотеки, които улесняват бързото разработване.
   * **dotenv**: За управление на секретни ключове и конфигурационни данни чрез .env файл. Това осигурява сигурност, като предпазва чувствителната информация от изтичане.
2. **Допълнителни технологии**:
   * **Stripe** е платежна система, която осигурява обработка на онлайн плащания. В проекта Stripe е интегрирана за създаване на сесии за плащане и обработка на транзакции в реално време.
   * **Flask-JWT-Extended**: Разширение за Flask, което осигурява лесна интеграция на JSON Web Tokens (JWT). JWT токените позволяват на потребителите да се идентифицират чрез сигурен токен, съхраняван в cookies.
   * **Stripe API**: Позволява извършване на плащания, управление на поръчки и обработка на транзакции. Интеграцията включва създаване на checkout сесии и обратни връзки за потвърждение на успешни или неуспешни плащания.

**Архитектура на приложението**

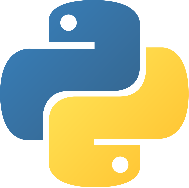
Client/Presentation layer



HTML, CSS, JavaScript, Jinja2

Python Flask, Flask-JWT-Extended, Flask-Migrate, Stripe API

Server/Business layer





Database layer



SQLAlchemy



***3.******Описание на е-бизнеса***

### 3.1. Основна идея

Целта на проекта е създаването на **онлайн платформа за тиймбилдинги, събития и празненства,** която предоставя на потребителите възможност да избират, персонализират и резервират различни услуги, като приключения, корпоративни събития, частни събития и празненства. Платформата предлага централизирано пространство, където се обединяват най-разнообразни предложения за всякакъв тип събития, по този начин избора е бърз и този, който взима решение може да избере най-подходящото нещо за повода си. Предлага се също и удобен начин за управление на поръчки, плащания и потребителска информация.

### 3.2. Основни потребители

* **Корпоративни клиенти**:
  + Компании, които организират тиймбилдинг събития за своите екипи.
  + Фокус върху услуги като "Приключенски дейности", "Конференции" и "Корпоративни събития" и тн.
* **Частни потребители**:
  + Индивидуални клиенти, които резервират частни събития като рождени дни, юбилеи, семейни събирания, романтични вечери и тн.

### 3.3. Бизнес процеси

#### 3.3.1 Каталог на услуги

* Услугите са разделени на **отдели** и **категории**, които улесняват потребителите при навигация и избор.
  + **Отдели**:
    - Отборни Приключения
    - Незабравими Празненства
    - Корпоративни събития
    - Частни събития
  + **Категории**:
    - За Отборни Приключения - На открито, На закрито, Адреналин
    - Незабравими Празненства – Детски рожден ден, Моминско парти, ергенско парти, Парти за компания
    - Частни събития – Семейни събития, Юбилеи, Романтична вечеря, Частна обиколка
    - Корпоративни събития – Конференции, Уъркшопи, Ритрийти, Промотиране на продукти

**Във всяка категория има поне по 10 продукта, от които потребителя може да избира!**

#### 3.3.2 Управление на количката

* Потребителите могат да добавят услуги в количката си, да редактират количествата или да премахват услуги.
* Платформата поддържа **анонимни потребители** чрез tracking ID, а за регистрирани потребители количката се свързва с техния акаунт.

#### 3.3.3 Обработка на поръчки

* Потребителите завършват поръчката си чрез плащане (интеграция със **Stripe**).
* Всеки потребител може да проследява поръчките си в **история на поръчките**.

#### 3.3.4 Плащания

* Интеграцията със Stripe осигурява:
  + Сигурни онлайн плащания.
  + Поддръжка на различни платежни методи.
  + Генериране на фактури за клиентите.

#### 3.3.5 Управление на потребителите

* Не регистрираните потребители могат да разглеждат и да добавят в количката, но могат да закупят продукт само като създадат профил.
* Регистрираните потребители имат достъп до персонализиран профил:
  + История на поръчките.
  + Управление на количката.
  + Възможност за преглед и редактиране на данни.

### 3.4. Предимства на платформата

#### 3.4.1 За клиенти

* **Лесен достъп**: Каталогът е структуриран и интуитивен за навигация.
* **Персонализация**: Клиентите могат да избират услуги според нуждите си.
* **Проследяване на поръчки**: Историята на поръчките осигурява лесен достъп до минали транзакции.

#### 3.4.2 За бизнес партньори

* **Интеграция**: Възможност за включване на допълнителни доставчици на услуги. Всеки, който предлага дадена услуга след договорка може да се включи в платформата.
* **Гъвкавост**: Поддържа разнообразни бизнес модели – от приключенски дейности до кетъринг услуги.

### 3.5. Интеграция с други системи

* **Stripe**: За обработка на плащания.
* **Email**: Изпращане на потвърждения за поръчки до клиентите.
* **Система за управление на инвентара** (потенциална интеграция в бъдеще): Управление на наличните ресурси и услуги.

### 3.6. Цели

* **Подобряване на потребителското изживяване**:
  + Удобна платформа за избор и резервиране на услуги.
* **Разширяване на клиентската база**:
  + Привличане на корпоративни клиенти и частни потребители.
* **Автоматизация**:
  + Намаляване на времето за обработка на поръчките чрез автоматизирани процеси.

### 3.7. Дългосрочна визия

* Добавяне на нови функционалности като:
  + Интеграция с доставчици за логистика.
  + Автоматизирани препоръки за подходящи услуги.
  + Филтрация за избор – брой хора, дата, цена и тн.
  + Нови партньори.
  + Ваучери за подарък.
  + Маркетингова стратегия.
  + Добавяне на възможности за платена реклама.
  + Добавяне на възможности за boost-ване на event.
  + Добавяне на функционалност за персонализиране на event.
* Разширяване на каталога с нови услуги, включително виртуални събития.

**Нашата визия е това да стане единствената платформа в България от където хората могат да търсят, купуват и организират събития. Винаги когато се нуждаят от тийм-билдинг или семейна изненада – те ще знаят че влизайки в платформата, те ще имат най-доброто преживяване. Тази платформа ще обединява в бъдеще всички, организатори на събития и това, което те предлагат.**

### *4. Описание на функционалността, обектите и потребителски случаи*

### 4.1. Функционални изисквания

#### Регистрация и вход за потребители

#### Регистрираните потребители:

#### Могат да създават акаунт чрез въвеждане на имейл и парола.

#### След успешна регистрация, потребителите имат достъп до персонализиран профил, който съдържа история на поръчките и други.

#### Входът се осъществява чрез въвеждане на имейл и парола, като системата валидира данните и генерира сигурен JWT токен за сесиите.

#### Потребителите могат да преглеждат историята на поръчките си и да управляват количката.

#### Потребителите могат да добавят продукти в количката.

#### Анонимните потребители:

#### Имат уникално генерирано tracking ID, което им позволява да добавят продукти в количката и да я редактират.

#### За да закупят услуга те трябва да се регистрират в системата.

#### Каталог на услуги

* Функционалност за преглед на отдели, категории и услуги.
* Визуализация на наличните услуги с изображения, описания и цени.

#### Управление на количката

* Добавяне, редактиране и премахване на продукти от количката.
* Възможност за преглед на стойността на количката и преминаване към плащане.

#### Обработка на поръчки

* Потвърждение и обработка на плащания чрез Stripe.
* Създаване на нова поръчка в базата данни.
* Изпращане на имейл с потвърждение за успешна поръчка.

#### История на поръчките

* Потребителите могат да преглеждат предишните си поръчки с информация за статус, дата и подробности за продуктите.

### 4.2. Нефункционални изисквания (FURPS+):

#### Functionality (Функционалност)

* Поддръжка на персонализирани профили и уникални tracking ID за анонимни потребители.
* Интеграция с Stripe за сигурно и бързо обработване на плащания.
* Динамично управление на каталога и количката.

#### Usability (Удобство)

* Прост и интуитивен потребителски интерфейс, създаден с HTML/CSS и JavaScript.
* Визуални известия (SweetAlert2) за обратна връзка с потребителя при действия, като добавяне в количка или потвърждение на поръчка.
* Мобилна съвместимост на уеб страниците.

#### Reliability (Надеждност)

* Сигурно съхранение на данни с помощта на SQLite.
* Използване на JWT токени за автентикация, осигуряващи сигурни сесии.
* Механизми за възстановяване при грешки в плащанията чрез Stripe.

#### Performance (Производителност)

* Бързо зареждане на каталога чрез асинхронни заявки с Fetch API.
* Оптимизирана структура на базата данни за ефективно извличане на данни.

#### Supportability (Поддръжка)

* Лесно управление на промените в базата данни с Flask-Migrate.
* Управление на конфигурации и ключове с помощта на dotenv.
* Модулна структура на кода с Flask Blueprints, което улеснява разширяемостта.

#### + (Допълнителни изисквания)

* **Security (Сигурност):**
  + Всички чувствителни данни се съхраняват сигурно (напр. хеширани пароли).
  + Транзакциите се обработват чрез Stripe, което гарантира високо ниво на защита.
* **Scalability (Мащабируемост):**
  + Базата данни може да бъде мигрирана към PostgreSQL за по-големи проекти.
  + API структурата позволява интеграция с други системи, като Google Analytics.
* **Maintainability (Поддръжка):**
  + Чист и добре структуриран код, разделен на модули.
  + Подробна документация за всички функции и интерфейси.

### 4.3. Обекти в системата

#### 1. Потребител (User)

* **Описание**: Обектът "Потребител" представлява регистриран потребител на платформата, който може да създава поръчки, управлява количката си и разглежда наличните продукти и услуги.
* **Полета**:
  + id - Уникален идентификатор за всеки потребител.
  + email - Имейл адрес, който е уникален и служи за идентификация.
  + password - Хеширана парола на потребителя за сигурност.
* **Методи**:
  + set\_password(password) - Хешира паролата на потребителя.
  + check\_password(password) - Проверява дали подадената парола съвпада с хешираната.

#### 2. Отдел (Department)

* **Описание**: Отделът е основна група, която съдържа категории, представляващи логическо групиране на продукти.
* **Полета**:
  + id - Уникален идентификатор на отдела.
  + name - Име на отдела (напр. "Приключения", "Семейни събития").
* **Връзки**:
  + categories - Списък от категории, свързани с този отдел.

#### 3. Категория (Category)

* **Описание**: Категорията е група от продукти, принадлежащи към определен отдел.
* **Полета**:
  + id - Уникален идентификатор на категорията.
  + name - Име на категорията (напр. "Романтична вечеря").
  + department\_id - Идентификатор на свързания отдел.
* **Връзки**:
  + products - Списък от продукти, свързани с категорията.

#### 4. Продукт (Product)

* **Описание**: Продуктът представлява конкретен артикул или услуга, която може да бъде добавена в количката и поръчана.
* **Полета**:
  + id - Уникален идентификатор на продукта.
  + name - Име на продукта (напр. "Каяк приключение").
  + description - Описание на продукта.
  + price - Цена на продукта.
  + recommended - Дали продуктът е препоръчан (булев тип).
  + image\_url - URL за изображение на продукта.
  + category\_id - Идентификатор на свързаната категория.
* **Връзки**:
  + category - Категорията, към която принадлежи продуктът.
  + cart\_items - Продуктът, свързан с елементите в количката.

#### 5. Количка (CartItem)

* **Описание**: Обектът "Количка" представлява временно хранилище за избраните продукти от потребител или гост (анонимен потребител с tracking ID).
* **Полета**:
  + id - Уникален идентификатор на елемента в количката.
  + product\_id - Идентификатор на продукта.
  + user\_id - Идентификатор на потребителя (ако е вписан).
  + tracking\_id - Уникален идентификатор за гости.
  + quantity - Количество на избрания продукт.
* **Връзки**:
  + product - Продуктът, свързан с елемента.
  + user - Потребителят, свързан с елемента (ако е вписан).
* **Методи**:
  + Добавяне на продукт.
  + Промяна на количество.
  + Премахване на продукт.

#### 6. Поръчка (Order)

* **Описание**: Обектът "Поръчка" съдържа всички детайли за успешно създадена поръчка, включително списък с артикули и статус на обработка.
* **Полета**:
  + id - Уникален идентификатор на поръчката.
  + user\_id - Идентификатор на потребителя, създал поръчката.
  + created\_at - Дата на създаване на поръчката.
  + total\_price - Обща стойност на поръчката.
  + status - Статус на поръчката (напр. "Pending", "Completed").
* **Връзки**:
  + order\_items - Списък с артикули, свързани с поръчката.
* **Методи**:
  + Създаване на поръчка.
  + Актуализация на статус.

#### 7. Артикул в поръчка (OrderItem)

* **Описание**: Обектът "Артикул в поръчка" представлява отделен продукт, добавен към дадена поръчка.
* **Полета**:
  + id - Уникален идентификатор на артикула.
  + order\_id - Идентификатор на свързаната поръчка.
  + product\_id - Идентификатор на продукта.
  + quantity - Количество на продукта.
  + price - Цена на продукта.
* **Връзки**:
  + product - Продуктът, свързан с артикула.
  + order - Поръчката, към която принадлежи артикулът.

### 4. Потребителски случаи

Начален списък на потребителските случаи

|  |  |  |  |
| --- | --- | --- | --- |
| **Име** | **Главен актьор** | **Кратко описание** | **Приоритет** |
| **Регистрация на потребител** | Потребител | Потребителят въвежда своите данни (имейл, парола) и системата създава нов профил в базата данни. | Среден |
| **Вход в системата** | Потребител | Потребителят въвежда имейл и парола, системата проверява данните и предоставя достъп до профила. | Среден |
| **Добавяне на продукт в количката** | Потребител | Потребителят избира продукт от каталога и го добавя в количката. Ако е влязъл в профила, продуктът се свързва с неговия профил, иначе с tracking ID. | Висок |
| **Преглед на количката** | Потребител | Потребителят преглежда текущите артикули в количката, включително количества, цени и общата сума. | Среден |
| **Редактиране на количката** | Потребител | Потребителят може да променя количествата на продуктите или да премахва продукти от количката. | Висок |
| **Завършване на поръчка** | Потребител | Потребителят натиска бутона "Checkout", системата обработва поръчката, включително плащането, и я съхранява в базата данни. | Висок |
| **Преглед на историята на поръчките** | Потребител | Потребителят може да види списък с предишните си поръчки, включително статус, дата и детайли за продуктите. | Нисък |
| **Обработка на плащане** | Платежен портал | Системата изпраща данните за плащане към платежния портал, който връща потвърждение за успешна или неуспешна транзакция. | Висок |
| **Разглеждане на категории и продукти** | Потребител | Потребителят може да преглежда списък с наличните категории. След като избере категория, системата му показва продуктите, свързани с нея. | Среден |

#### **Кратки описания на 5 избрани потребителски случая**

#### Случай 1: Регистрация на потребител

1. Потребителят отваря формата за регистрация.
2. Въвежда имейл и парола.
3. Системата проверява дали имейлът вече съществува.
4. Системата проверява дали и паролите съвпадат.
5. Ако имейлът е уникален и паролите съвпадат, системата хешира паролата и създава акаунт.
6. Успешно съобщение се показва на потребителя и се изпраща в home page.

#### Случай 2: Добавяне на продукт в количката

1. Потребителят преглежда продуктите в каталога.
2. Избира продукт и натиска бутон „Add to Cart“.
3. Системата проверява дали продуктът вече е в количката.
   * Ако да, увеличава количеството.
   * Ако не, добавя нов запис в таблицата CartItem.
4. Потребителят получава съобщение за успешна добавка.

#### Случай 3: Завършване на поръчка

1. Потребителят отваря количката и натиска бутон „Checkout“.
2. Ако не е влязъл в акаунта си, системата го пренасочва към страницата за вход.
3. Ако няма профил се регистрира и тогава влиза в профила си.
4. Системата изчислява общата стойност на поръчката и пренасочва към Stripe за плащане.
5. След успешно плащане се създава запис в таблицата Order и артикулите се добавят в OrderItem.
6. Потребителят получава потвърждение за успешна поръчка.

#### Случай 4: Преглед на история на поръчките

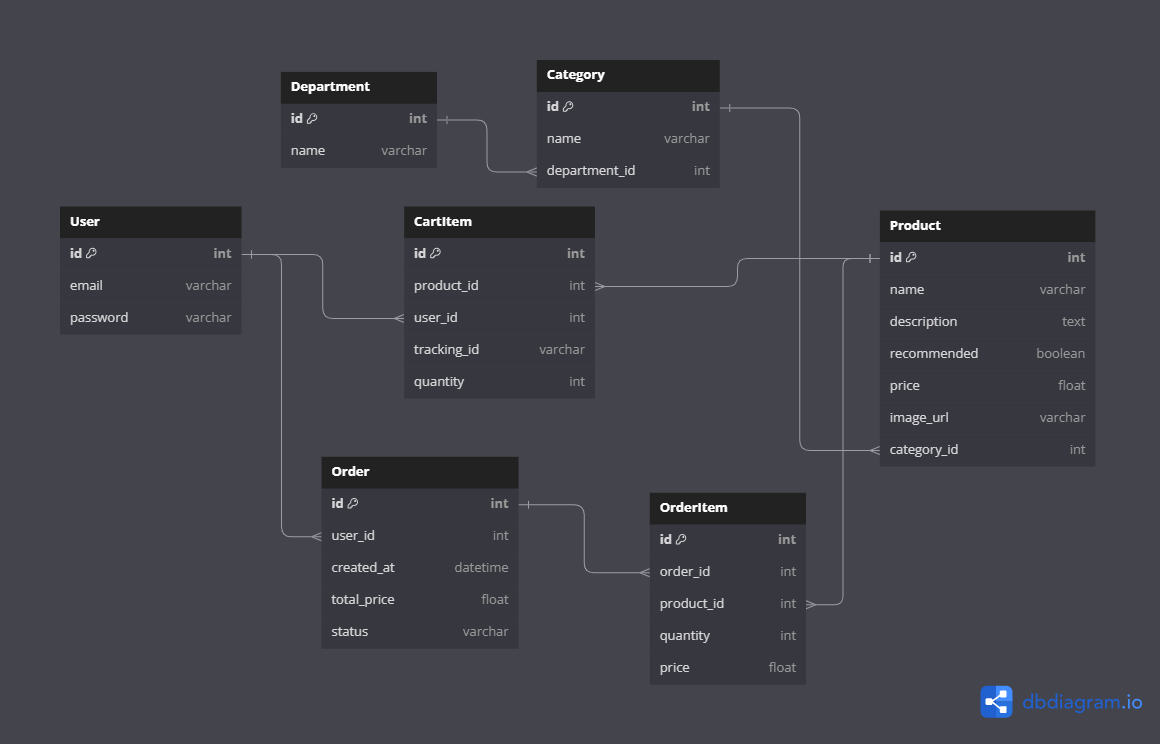
1. Потребителят отваря страницата „Orders“.
2. Системата зарежда всички поръчки, свързани с потребителя.
3. За всяка поръчка се показват:
   * Номер на поръчката.
   * Дата на създаване.
   * Статус.
   * Общата цена и артикули.
4. Потребителят може да разгледа детайлите на всяка поръчка.

#### Случай 5: Редактиране на количката

1. Потребителят отваря страницата на количката.
2. Преглежда добавените продукти и решава да редактира количеството.
3. Натиска бутон „+“ за увеличаване на количеството или „-” за намаляване.
4. Системата проверява:
   * Ако количеството е намалено до 0, продуктът се премахва от количката.
   * Ако количеството се увеличава, актуализира записа в базата данни.
5. Обновената сума и списък с артикули се показват в количката.

***5. Описание на Базата от Данни***

E/R диаграма на базата данни



**Описание на таблиците в базата данни**

**Таблица: User - Потребител**

|  |  |  |
| --- | --- | --- |
| **Име на колоната** | **Тип** | **Описание** |
| id | number | Уникален идентификатор на потребителя. Автоматично се генерира. |
| email | string | Имейл адрес на потребителя. Използва се за вход в системата. |
| password | string | Хеширана парола на потребителя. |

**Таблица: Department - Отдел**

|  |  |  |
| --- | --- | --- |
| **Име на колоната** | **Тип** | **Описание** |
| id | number | Уникален идентификатор на отдела. Автоматично се генерира. |
| name | string | Име на отдела. Задължително поле. |

**Таблица: Category - Категория**

|  |  |  |
| --- | --- | --- |
| **Име на колоната** | **Тип** | **Описание** |
| id | number | Уникален идентификатор на категорията. Автоматично се генерира. |
| name | string | Име на категорията. Задължително поле. |
| department\_id | number | Връзка към таблицата **Department**. Посочва към кой отдел принадлежи. |

**Таблица: Product - Продукт**

|  |  |  |
| --- | --- | --- |
| **Име на колоната** | **Тип** | **Описание** |
| id | number | Уникален идентификатор на продукта. Автоматично се генерира. |
| name | string | Име на продукта. Задължително поле. |
| description | string | Описание на продукта. |
| recommended | boolean | Флаг, показващ дали продуктът е препоръчан. |
| price | float | Цена на продукта. |
| image\_url | string | URL адрес към изображението на продукта. |
| category\_id | number | Връзка към таблицата **Category**. Посочва към коя категория принадлежи. |

**Таблица: CartItem - Артикул в количката**

|  |  |  |
| --- | --- | --- |
| **Име на колоната** | **Тип** | **Описание** |
| id | number | Уникален идентификатор на артикула. Автоматично се генерира. |
| product\_id | number | Връзка към таблицата **Product**. Посочва кой продукт е добавен. |
| user\_id | number | Връзка към таблицата **User**. Ако потребителят е влязъл в профила си. |
| tracking\_id | string | Уникален идентификатор за анонимен потребител. |
| quantity | number | Количество на артикула. |

**Таблица: Order - Поръчка**

|  |  |  |
| --- | --- | --- |
| **Име на колоната** | **Тип** | **Описание** |
| id | number | Уникален идентификатор на поръчката. Автоматично се генерира. |
| user\_id | number | Връзка към таблицата **User**. Посочва кой потребител е направил поръчката. |
| created\_at | datetime | Дата и час на създаване на поръчката. |
| total\_price | float | Обща стойност на поръчката. |
| status | string | Статус на поръчката (напр. „Pending“, „Completed“). |

**Таблица: OrderItem - Артикул в поръчка**

|  |  |  |
| --- | --- | --- |
| **Име на колоната** | **Тип** | **Описание** |
| id | number | Уникален идентификатор на артикула. Автоматично се генерира. |
| order\_id | number | Връзка към таблицата **Order**. Посочва към коя поръчка принадлежи. |
| product\_id | number | Връзка към таблицата **Product**. Посочва кой продукт е поръчан. |
| quantity | number | Количество на продукта в поръчката. |
| price | float | Единична цена на продукта към момента на поръчката. |

***6. Описание на отделите и категориите***

#### Брой отдели, категории и стоки/услуги

1. **Отдели**:
   * Отделите представляват основните бизнес направления на електронния магазин.
   * Брой: **4**.
     + **Team Adventures** - Приключенски дейности за екип.
     + **Unforgettable Celebrations** – Незабравими празненства.
     + **Private Events** - Частни събития.
     + **Corporate Events** - Корпоративни събития.
2. **Категории**:
   * Във всеки отдел има няколко категории. Те са разделени смислово и съдържат конкретни видове услуги или стоки, достъпни за поръчка.
   * Пример за категории:
     + За Отборни Приключения - На открито, На закрито, Адреналин
     + Незабравими Празненства – Детски рожден ден, Моминско парти, ергенско парти, Парти за компания
     + Частни събития – Семейни събития, Юбилеи, Романтична вечеря, Частна обиколка
     + Корпоративни събития – Конференции, Уъркшопи, Ритрийти, Промотиране на продукти
   * **Стоки/Услуги**:
   * Всеки продукт или услуга се класифицира в определена категория.
   * Примери за стоки/услуги:

* В отдел „Отборни приключения“ – категория „На открито“ – „Hiking adventure“, “Kayaking Expedition”, “River rafting”, etc.
* В отдел „Корпоративни събития“ – категория „Конференции“ – „Post-Event Report“, “Conference room rental”, “On-Site Catering”, etc.

**Всяка категория съдържа поне по 10 услуги!**

#### Данни за отдели и категории

* Отделите и категориите са организирани йерархично в базата данни, като всеки отдел може да има множество категории, а всяка категория може да съдържа множество стоки/услуги.
* Взаимоотношения:
  + Отделът е родител на категориите.
  + Категорията е родител на продуктите.

#### Връзка с потребители и количка

* Стоките/услугите могат да бъдат добавяни към потребителската количка чрез таблицата CartItem, която свързва конкретния продукт с количката на даден потребител.

### *7. Реализация на потребителската кошница*

#### 7. 1. Реализация на потребителската кошница

Потребителската кошница е основен компонент на системата, който позволява на потребителите да избират, съхраняват и управляват стоки/услуги, които планират да закупят. Тази функционалност е реализирана така, че да отговаря както на потребности на регистрирани потребители, така и на нерегистрирани потребители, използвайки tracking\_id.

**Функционалности:**

1. **Добавяне на стока/услуга към кошницата:**
   * Потребителят може да добави избрана стока/услуга към кошницата с натискане на бутон "Add to Cart".
   * Системата съхранява информация за продукта, неговата цена и количество.
   * Ако потребителят не е влязъл в профила си, системата използва tracking\_id (уникален идентификатор на сесията) за асоцииране на кошницата.
   * Ако потребителят е влязъл, кошницата се асоциира с неговия user\_id.
2. **Показване на текущото съдържание на кошницата:**
   * Стоките се показват в табличен формат с колони за:
     + Име на продукта.
     + Цена на единица.
     + Количество.
     + Обща цена за продукта.
3. **Редактиране на количката:**
   * Увеличава или намалява количеството на избрани продукти чрез "+" и "-" бутони.
   * Премахва артикули от кошницата чрез бутон "Remove".
   * При всяка промяна общата стойност на поръчката се обновява динамично.
4. **Сумиране на общата стойност на поръчката:**
   * Автоматично изчисляване на крайната сума на поръчката въз основа на количествата и цените на продуктите.
5. **Запис в базата данни:**
   * Кошницата се съхранява в таблицата CartItem:
     + Полета:
       - id - Уникален идентификатор на артикула в кошницата.
       - product\_id - Референция към продукта.
       - user\_id или tracking\_id - За асоцииране с потребителя.
       - quantity - Количество.
       - price - Цена на артикула.

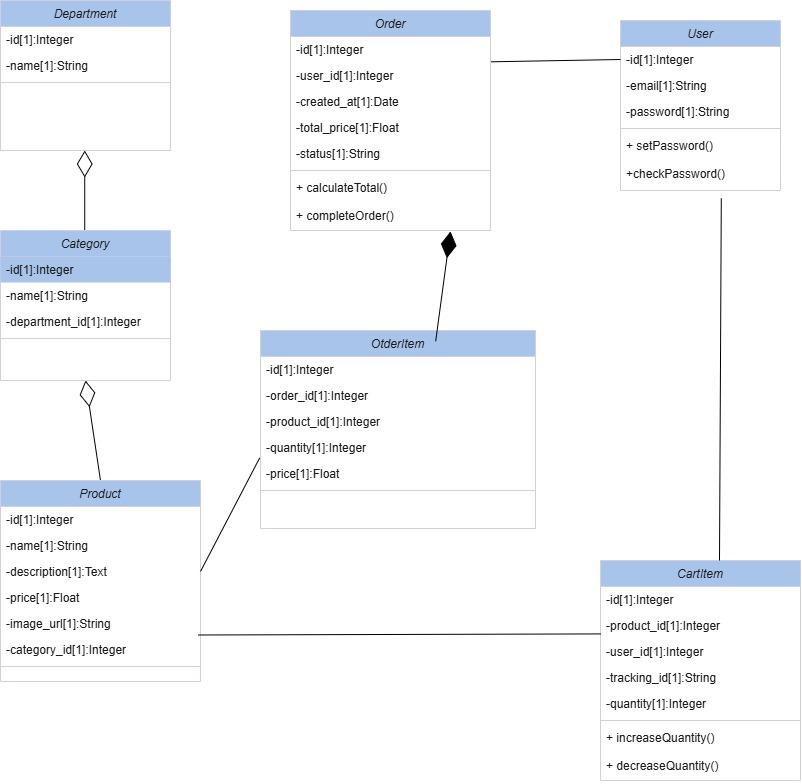
#### 7. 2. Реализиране и показване на обработването

Обработката на поръчките е съществена част от следващия етап на разработка и обхваща следните компоненти:

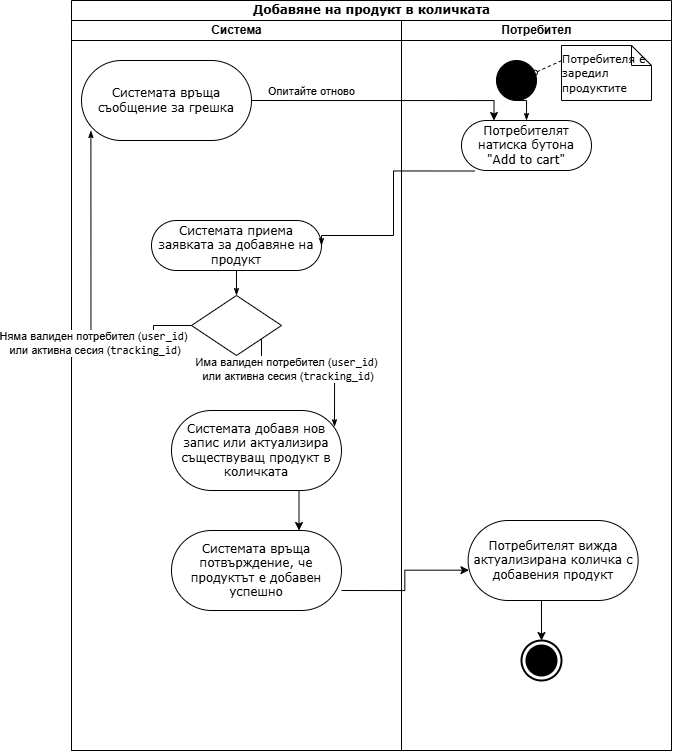
1. **Обработване на поръчката:**
   * След натискане на бутон "Checkout", поръчката се обработва, и системата:
     + Изчислява крайната стойност на поръчката.
     + Създава нов запис в таблицата Order.
     + Прехвърля всички артикули от CartItem към OrderItem.
2. **Показване на статус на поръчка:**
   * Ако плащането не е успешно, потребителят се пренасочва към страница “Cancel”.
   * След успешно обработване, потребителят се пренасочва към страница "Success", която:
     + Показва общата сума на поръчката.
     + Списък на закупените стоки.
     + Дава опция да се върне към home page.
3. **Показване на история на поръчките:**
   * Потребителят може да види историята на поръчките си в отделна страница.
   * За всяка поръчка се показват:
     + Дата.
     + Списък на продуктите.
     + Общата цена.
     + Статус на поръчката (например: "Завършена").
4. **Връзка с платежната система:**
   * Реализация на свързване със Stripe за обработка на плащания.
   * При успешна обработка се показва страница за потвърждение ("Payment Successful").

### *8. UML Диаграми*

#### 8. 1. Диаграма на класовете



#### 8. 2. Диаграма на дейността (Activity Diagram) - Пример



#### 8.2. Диаграма на последователността (Sequence Diagram)

*Поради технически ограничения, диаграмата на процеса за обработка на поръчка не е включена в настоящата документация. Вместо това предоставяме текстово описание, което детайлно описва етапите на процеса.*

Потребител -> Потребителски интерфейс (UI): Натиска бутон "Завърши поръчката"

UI -> Сървър: Изпраща заявка за завършване на поръчката

Сървър -> Платежен портал: Обработва плащането

Платежен портал --> Сървър: Потвърждение на плащането

Сървър -> База данни: Създава поръчка

Сървър -> База данни: Добавя артикули към OrderItem

База данни --> Сървър: Потвърждава добавянето

Сървър -> База данни: Премахва артикули от CartItem

База данни --> Сървър: Потвърждава премахването

Сървър --> UI: Потвърждава успешното завършване на поръчката

UI --> Потребител: Показва страница "Успешно плащане"

**->**: **Представлява заявка или действие.**

**-->: Представлява отговор или потвърждение.  
Отстъпите показват вложени или зависими действия.**

### *9. Описание на съхранените процедури*

Съхранените процедури в проекта са реализирани чрез Python код и SQLAlchemy ORM за взаимодействие с базата данни. Те осигуряват автоматизация на важни операции и поддържат функционалността на платформата. Основните процедури включват:

#### Потребител (User)

* **GET**: Извличаме всички потребители или конкретен потребител по id.
* **POST**: Регистрация на нов потребител чрез email и парола. Автоматично се хешира паролата и потребителят се записва в базата данни.
* **UPDATE**: Обновяване на информацията за потребителя, като промяна на email или парола.
* **DELETE**: Изтриване на потребител по id.

#### Продукт (Product)

* **GET**: Извличане на всички продукти или продукт по id.
* **POST**: Създаване на нов продукт, като се предоставят име, описание, цена и изображение.
* **UPDATE**: Обновяване на информация за продукт, като цена или описание.
* **DELETE**: Изтриване на продукт по id.

#### Категория (Category)

* **GET**: Извличане на всички категории или категория по id.
* **POST**: Създаване на нова категория с име и описание. Възможност за асоцииране с отдел.
* **UPDATE**: Промяна на информация за категория, като име или описание.
* **DELETE**: Изтриване на категория по id.

#### Отдел (Department)

* **GET**: Извличане на всички отдели или конкретен отдел по id.
* **POST**: Създаване на нов отдел с предоставяне на име и описание.
* **UPDATE**: Обновяване на информация за отдел, като име или описание.
* **DELETE**: Изтриване на отдел по id.

#### ● Кошница (CartItem)

* **GET**: Извличане на всички артикули в кошницата, асоциирани с user\_id или tracking\_id.
* **POST**: Добавяне на продукт в кошницата с посочване на product\_id, количество и цена.
* **UPDATE**: Актуализация на количеството на продукт в кошницата.
* **DELETE**: Изтриване на артикул от кошницата по id.

#### Поръчка (Order)

* **GET**: Извличане на всички поръчки или конкретна поръчка по id.
* **POST**: Създаване на нова поръчка с посочване на user\_id, обща стойност и статус. Прехвърляне на артикули от кошницата в поръчката.
* **UPDATE**: Промяна на статус на поръчката (например, от "Pending" към "Completed").
* **DELETE**: Изтриване на поръчка по id.

#### ● Артикул в поръчка (OrderItem)

* **GET**: Извличане на всички артикули от конкретна поръчка.
* **POST**: Създаване на нов артикул в поръчката с посочване на product\_id, количество и цена.
* **UPDATE**: Актуализация на количеството или цената на артикул в поръчка.
* **DELETE**: Изтриване на артикул от поръчка по id.

### *10. Модул за потребителските сметки, усъвършенствано обработване на потребителските поръчки, конвейер за обработка на поръчките, включващ оторизиране на кредитната карта*

* **Модул за потребителски сметки:**
  + Потребителите могат да създават акаунти, да се вписват и да виждат персонализирана информация като поръчки и история на покупките.
  + Използва се JWT (JSON Web Token) за сигурна автентикация и идентификация.
* **Усъвършенствано обработване на поръчките:**
  + При завършване на поръчка се генерира уникален идентификатор на поръчката (order\_id).
* **Конвейер за обработка на поръчките:**
  + Включва логика за обработка на плащания чрез Stripe API.
  + Проверява валидността на въведените данни за кредитна карта и оторизира плащането.
  + При успешна транзакция поръчката се маркира като „Завършена“.

### *11. Описание на характеристиките за иновативност на предложения модел*

* **Иновативен каталог за екипни събития:**
  + Предоставя система, която обединява всички възможности за събития в България на едно място. Достъпна, прегледна и с интуитивен дизайн.
  + Предоставя каталог, който комбинира различни видове събития – тиймбилдинги, частни партита, корпоративни събития и т.н.
* **Гъвкав модел за управление на потребители:**
  + Позволява на потребителите да използват функционалностите на платформата както като гости, така и като регистрирани потребители.
  + Гост потребителите имат възможност да прехвърлят количката си към акаунта си след регистрация.
* **Интеграция с модерни платежни системи:**
  + Използване на Stripe за лесно и сигурно обработване на онлайн плащания.
  + Осигуряване на защита чрез JWT.

### *12. Описание на необходимостта от и реализация на свързване-интеграция с други системи*

### Използването на външни системи подобрява качеството на системата, като предоставя вече тествани и удобни функционалности от други места, които правят потребителското изживяване още по-добро.

* **Интеграция със Stripe:**
  + Използва се за обработка на плащания, валидиране на данни за кредитни карти и управление на транзакции.
  + Потребителите могат да извършват плащания директно през платформата, без да напускат приложението.
* **Интеграция с реклами, за да се постига допълнителна печалба (планирано):**
* **Интеграция с маркетингови платформи (планирано):**
  + В бъдеще може да се добавят интеграции със социални медии за промотиране на събития и получаване на данни за клиентите.
* **Модул за имейл нотификации:**
  + Изпраща автоматични известия на потребителите при успешна поръчка или промяна на статус на поръчката.

***13. RUP - Rational Unified Process***

Методология за управление на софтуерни проекти, която предоставя насоки, модели и най-добри практики за успешното планиране, разработка и внедряване на софтуерни системи. RUP следва итеративен и инкрементален подход, което означава, че проектът се разделя на множество итерации, всяка от които добавя стойност към крайния продукт.

#### **Четирите основни фази на RUP:**

1. **Иницииране (Inception)**: Определяне на целите, събиране на изискванията и идентифициране на основните рискове.
2. **Разширение (Elaboration)**: Създаване на основната архитектура и уточняване на изискванията.
3. **Изграждане (Construction)**: Разработка на основната функционалност и тестване на системата.
4. **Предаване (Transition)**: Доставяне на продукта и обучение на потребителите.

**Легенда за таблицата по-долу:**

П1, П2 - Планиране в съответните итерации

Д1,Д2 - Детайлизиране в съответните итерации

И1, И2 - Изграждане в съответните итерации

Р1, Р2 - Предаване в съответните итерации

0-5 - Приоритет на дейността: минимален (0) - максимален (5)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Етапи | Име | П1 | Д1 | Д2 | И1 | И2 | И3 | Р1 | Р2 |
| 1 | Списък на заинтересованите страни | 5 | 4 | 3 | 0 | 0 | 0 | 0 | 0 |
| 1 | Събиране на изисквания | 5 | 4 | 3 | 0 | 0 | 0 | 0 | 0 |
| 1 | Домейн модел | 3 | 4 | 5 | 1 | 1 | 0 | 1 | 0 |
| 1 | Анализ на системи с аналогична функционалност | 4 | 4 | 3 | 3 | 1 | 1 | 0 | 0 |
| 1 | Визия | 4 | 5 | 3 | 1 | 1 | 1 | 0 | 0 |
| 2 | Определяне на функционални изисквания | 5 | 5 | 4 | 2 | 2 | 1 | 2 | 1 |
| 2 | Определяне на данните за съхранение | 4 | 5 | 5 | 2 | 1 | 1 | 0 | 0 |
| 2 | Създаване на дизайн на базата данни | 3 | 5 | 4 | 2 | 2 | 1 | 0 | 0 |
| 2 | Идентифициране на роли и потребители | 4 | 4 | 3 | 2 | 2 | 0 | 0 | 0 |
| 2 | Нефункционални изисквания | 2 | 3 | 5 | 2 | 2 | 1 | 1 | 0 |
| 2 | Модел на потребителските случаи | 3 | 4 | 5 | 2 | 2 | 1 | 0 | 0 |
| 2 | Речник | 2 | 3 | 2 | 1 | 2 | 1 | 1 | 0 |
| 2 | Определяне на технологии и инструменти | 1 | 5 | 4 | 3 | 1 | 1 | 0 | 0 |
| 2 | UML Диаграми | 2 | 4 | 5 | 2 | 1 | 1 | 0 | 0 |
| 2 | CD Модели | 2 | 4 | 5 | 1 | 1 | 0 | 0 | 0 |
| 2 | Потребителски интерфейс | 1 | 4 | 5 | 3 | 3 | 2 | 1 | 1 |
| 3 | Изграждане на система и база от данни и тяхното свързване | 0 | 0 | 1 | 3 | 5 | 4 | 2 | 1 |
| 3 | Функционалности от първи етап на разработка | 0 | 0 | 1 | 5 | 5 | 4 | 3 | 1 |
| 3 | Функционалности от втори етап на разработка | 0 | 0 | 1 | 5 | 5 | 4 | 3 | 1 |
| 3 | Функционалности от трети етап на разработка | 0 | 0 | 1 | 3 | 5 | 4 | 2 | 1 |
| 3 | Създаване на интеграция на Stripe за плащания | 0 | 0 | 1 | 1 | 2 | 5 | 1 | 1 |
| 3 | Разработка на фронт-енд (потребителски интерфейс) | 0 | 0 | 1 | 2 | 4 | 3 | 2 | 0 |
| 3 | Тестване на системата | 0 | 0 | 0 | 2 | 3 | 4 | 5 | 5 |
| 4 | Потребителски обучения и тестове | 0 | 0 | 0 | 0 | 1 | 2 | 5 | 4 |
| 4 | Създаване на документация | 0 | 0 | 0 | 2 | 1 | 1 | 5 | 5 |
| 4 | Финализиране и предаване на проекта | 0 | 0 | 0 | 1 | 1 | 2 | 5 | 5 |

### *14. Демо сървър*

Приложението е разработено с **Flask**, използвайки вграден демо сървър за управление на HTTP заявки и сесии. Сървърът е настроен да „слуша“ на порт **5000** по подразбиране. За фронтенд частта на приложението, използваща **HTML/CSS/JavaScript**, файловете се зареждат локално от клиента или през Flask.

### *15. Стартиране на приложението*

#### **Инструкции за успешно стартиране на приложението:**

1. **Клонирайте репото**:  
   **Стартиране на сървъра**:  
   cd server
2. **Инсталирайте необходимите зависимости:**  
   pip install -r requirements.txt
3. **Стартирайте сървъра:**  
   flask run

Сървърът ще „слуша“ на порт **5000**.

Отворете браузър и навигирайте до:  
http://127.0.0.1:5000/

1. **База данни**:

Уверете се, че базата данни е конфигурирана и мигрирана. Можете да изпълните миграции с:  
flask db upgrade

1. **Тестване на функционалността**:

За да използвате платформата, заредете клиентската част и започнете да разглеждате или добавяте продукти към количката.

### Забележки:

* Уверете се, че Python версията е съвместима с изискванията на проекта.
* Убедете се, че всички зависимости, посочени в requirements.txt, са инсталирани.
* Ако срещнете проблеми с базата данни, проверете настройките в конфигурационния файл (config.py или .env).