



Курсов проект по
Системи за е-бизнес
зимен семестър 2024/2025

Тема на проект:

**Система за електронна търговия на
технологични продукти и решения –**

Denev Computers

Изработил:

Мая Денева, ФН: 2MI0700013



Съдържание

1.	Описание на учебния проект	3
2.	Архитектура и дизайн на системата	3
2.1	Front-End:	4
2.1 a)	Технологии.....	4
2.1 b)	Използвани библиотеки.....	4
2.1 c)	Диаграма на компонентите.....	5
2.2	Back-End:	6
2.2 a)	Технологии.....	6
2.2 b)	Използвани технологии	6
2.3	База данни:	7
3.	Описание на е-бизнеса	7
3.1	Описание на х-ките за иновативност на предложени модел - промяна на съществуващ	7
3.2	Описание на необходимостта от и реализация на свързване-интеграция с други системи	9
3.2	Описание на необходимостта от и реализация на свързване-интеграция с други системи	9
4.	Описание със стандартни методи на функционалността, обектите и потребителски случаи, които ще се реализират	11
5.	Описание на Базата от Данни -- бр. отдели, категории, стоки/ услуги	12
6.	Описание на категориите и отделите	21
1.Създаване и показване на каталог в БД.....		21
2. Описание на съхранените процедури.....		22
7.	Етапи на разработка (RUP)	27
8.	Стартиране на приложението	29



1. Описание на учебния проект

"Денев Компютърс" ЕООД предлага услуги по компютърна поддръжка и продажба на нов и употребяван хардуер. След извършване на анализ на текущия начин на работа на фирмата, бяха установени следните проблеми:

- **Локален обхват на продажби:** Продажбите са силно ограничени до местния пазар. Липсата на онлайн каталог с продукти и информация за бизнеса значително затруднява разширяването на пазара извън града.
- **Непълно използване на репутацията:** Компанията има силна репутация в Попово, но липсата на централизирано място за информация относно услугите води до загуба на потенциални клиенти.
- **Неудобен модел за продажби:** Дистрибуторският модел на бизнеса ограничава достъпа на клиентите до пълния каталог с продукти. Клиентите разчитат основно на консултации с персонала, което може да доведе до неудовлетвореност и загуба на интерес.
- **Липса на проследяване на поръчки:** Клиентите нямат възможност да следят състоянието на своите поръчки или ремонти, което намалява прозрачността и доверието.

Основна цел на проекта:

Да се създаде система за електронен бизнес, която:

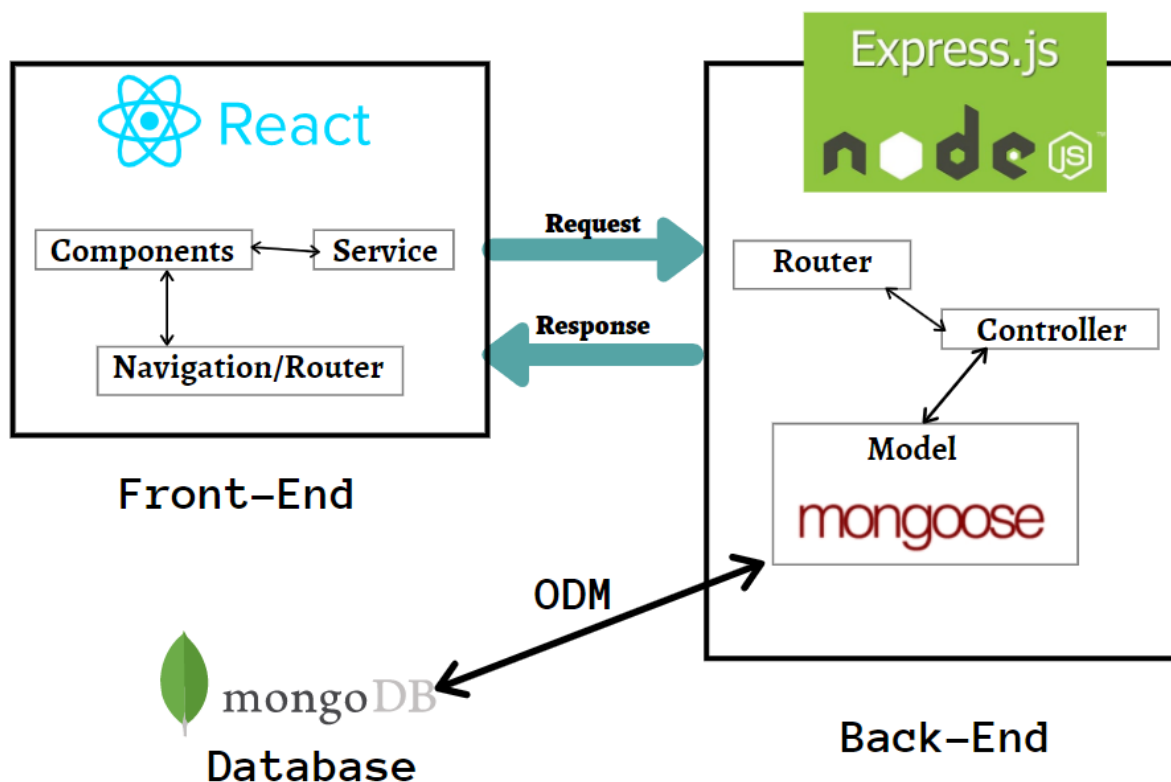
- Да разшири обхвата на продажби чрез предоставяне на онлайн платформа за разглеждане и покупка на продукти.
- Да използва мобилно-оптимизиран дизайн, за да бъде достъпна за широк кръг от клиенти.
- Да предоставя ясна и изчерпателна информация за предлаганите услуги и продукти.
- Да включва функционалност за проследяване на поръчки и ремонти, повишавайки прозрачността и удовлетвореността на клиентите.
- Да подобри конкурентоспособността на компанията и да привлече нови клиенти, като същевременно задържи текущите.

2. Архитектура и дизайн на системата

MERN стекът (MongoDB, Express.js, React, Node.js) е избран за този проект поради своята популярност и модерност в разработката на уеб приложения. Този стек предоставя пълна среда за разработка с JavaScript, улеснявайки интеграцията между front-end и back-end компонентите. MongoDB е NoSQL база данни, която осигурява гъвкавост за управление на данни. React предоставя ефективен и модулен подход за



изграждане на потребителски интерфейси, докато Express.js и Node.js позволяват изграждането на мащабируеми и бързи бекенд решения.



2.1 Front-End:

2.1 а) Технологии

- **React:** Рамка, използвана за изграждане на потребителския интерфейс.
 - **Components:** Логически модули, отговарящи за визуализацията и интеракциите с потребителя.
 - **Service:** Обработка на заявки към Back-End.
 - **Navigation/Router:** Управлява навигацията между различни части на приложението.
- **TailwindCSS:** Рамка с отворен код, предоставяща класове за стилизация.

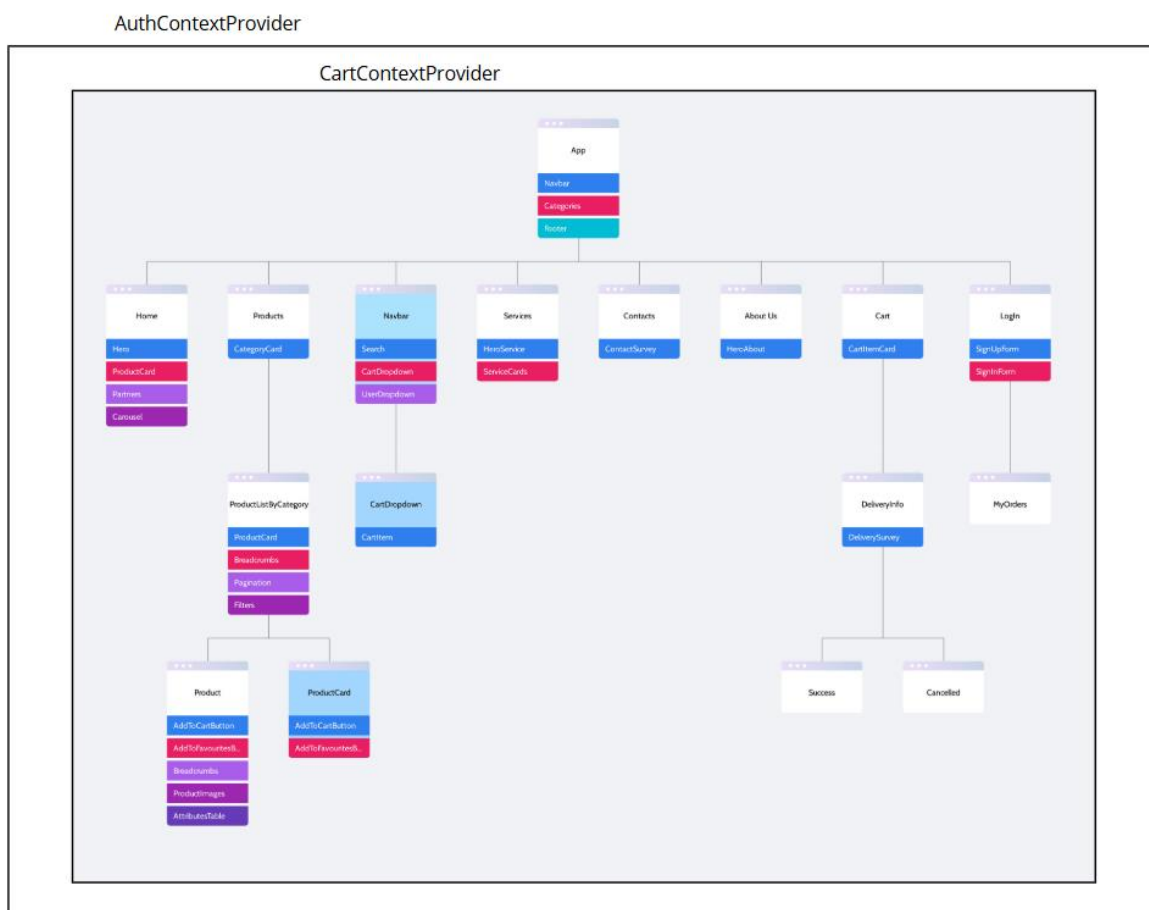
2.1 б) Използвани библиотеки

- **DaisyUI и Flowbite:** библиотеки за React компоненти, използващи TailwindCSS.
- **Axios:** за обработка на заявки към Back-end.



- **React-router-dom:** За навигацията между различни части на приложението.
- **Framer-motion:** Използван за анимация на компонент на началната страница.
- **SurveyJS:** JavaScript библиотека за форми и анкети, използвани за получаване на запитвания, и получаване на информация за поръчка.
- **@react-google-maps/api:** За изобразяване на гугъл карта с местоположение на офиса на Denev Computers, в страницата за контакти.
- **@stripe-stripe-js:** За интеграция с портал за плащане, предоставен от Stripe.

2.1 с) Диаграма на компонентите



Разяснение:

- Бял цвят – Страници
- Ярък цвят – Компонентите, вложени в страниците
- Син цвят – Разбивка на компоненти, които съдържат допълнителни компоненти
- Рамка – Контекст, който осигурява данни и функции за всички компоненти, които се намират в неговата рамка.



Забележки:

- `App.js` служи като основен компонент, който обединява всички други компоненти на приложението. Той се импортира и се рендерира в главния файл (`index.js`), което стартира приложението. Използва се за конфигуриране на маршрути между отделните страници. Задава общия "скелет" на приложението, като включва общи компоненти като навигация, футър.
- Страниците са дървовидно подредени по примерна последователност.

2.2 Back-End:

2.2 а) Технологии

- **Express.js:** Уеб сървър, изграден върху Node.js, отговорен за обработка на заявки и предоставяне на отговори.
 - **Router:** Маршрутизатор, който разпределя заявките към съответните контролери.
 - **Controller:** Съдържа логиката за обработка на бизнес логиката.
 - **Model (Mongoose):** Управлява взаимодействието с MongoDB чрез ODM (Object Document Mapper).
 - **JavaScript:** Програмният език използван за реализация.

2.2 б) Използвани технологии

- **mongoose:** Object Document Mapper (ODM) за работа с MongoDB. Позволява дефиниране на модели на данни и лесна манипулация с тях в базата данни.
- **xml2js:** Библиотека за преобразуване на XML данни в JavaScript обекти. Използвана за интеграция с външна система, която предоставя данни в XML формат.
- **cors:** Middleware за управление на Cross-Origin Resource Sharing. Разрешава взаимодействия между различни домейни, което е ключово за разделянето на front-end и back-end.
- **dotenv:** Управлява конфигурационни параметри на приложението чрез `.env` файл, например URL на базата данни, API ключове и други чувствителни данни.
- **express-session:** Библиотека за управление на потребителски сесии. Поддържа функционалности като аутентикация и запазване на състоянието на количка между различни заявки.
- **connect-mongo:** Осигурява интеграция между сесиите на Express.js и MongoDB. Позволява съхранение на потребителски сесии в MongoDB, което осигурява по-голяма сигурност и устойчивост.
- **nodemailer:** Използва се за изпращане на имейли. В проекта се използва за изпращане на потвърждения за поръчки.



- **stripe:** Библиотека за интеграция със Stripe API. Използва се за обработка на плащания в системата, включително създаване и потвърждаване на транзакции.
- **passport:** Библиотека за аутентикация. Поддържа множество стратегии за идентификация, включително локални и социални.
- **passport-local:** Конкретна стратегия за локална аутентикация с потребителско име и парола. Използва се за управление на вход и регистрация.
- **bcrypt:** Използва се за криптиране на пароли. Тази библиотека защитава чувствителните данни на потребителите чрез хеширане преди съхранение в базата данни.

2.3 База данни:

- **MongoDB:** NoSQL база данни, използвана за съхраняване на информация за продукти, заявки и поръчки.
 - Свързана с Back-End чрез Mongoose за лесна манипулация на данните.

3. Описание на е-бизнеса

Системата за електронен бизнес на "Денев Компютърс" разширява възможностите на компанията, като осигурява нови канали за взаимодействие с клиенти. Основните аспекти на е-бизнеса включват:

- **Разширяване на продажбите:** Чрез онлайн платформа клиентите имат възможност да разглеждат и закупуват продукти, независимо от тяхното местоположение. Това премахва географските ограничения на бизнеса.
- **Мобилна достъпност:** Дизайнът на системата е оптимизиран за мобилни устройства, което осигурява удобен достъп за по-широк кръг от клиенти.
- **Информация за услуги и продукти:** Онлайн платформата предоставя изчерпателна и ясна информация за всички предлагани продукти и услуги, като по този начин се увеличава информираността на клиентите.
- **Проследяване на поръчки и ремонти:** Системата включва функционалност, която позволява на клиентите да следят текущото състояние на поръчките и ремонтите си, което повишава тяхното доверие и удовлетвореност.
- **Подобряване на конкурентоспособността:** Чрез модернизиране на бизнес модела, системата помага на компанията да привлича нови клиенти и да задържа съществуващите, като същевременно засилва позицията ѝ на пазара.

3.1 Описание на х-ките за иновативност на предложения модел - промяна на съществуващ

Предложен Модел: Dropshipping Модел



Запазва, но подобрява и улеснява съществуващия дистрибуторски модел.

Dropshipping е бизнес модел за електронна търговия, при който продавачът не поддържа собствен инвентар от продукти. Вместо това, когато клиент направи поръчка, продавачът предава тази поръчка на трета страна (доставчик или производител), която се грижи за изпращането на продукта директно на клиента.

Основни участници в Dropshipping модела:

- **Продавач (Retailer):**
 - Това е лицето или компанията, която управлява онлайн магазина (Днев Компютърс).
 - Продавачът избира продукти, които иска да предлага на клиентите, и ги рекламира на своята платформа.
 - Продавачът обработва клиентските поръчки и получава плащанията, но не държи инвентар и не извършва доставката.
- **Клиент:**
 - Купувачът, който прави поръчка през онлайн магазина на продавача.
 - Клиентът вижда само платформата на продавача и не знае, че продуктът идва от трета страна.
- **Доставчик (Supplier):**
 - Това е производителят или търговецът на едро, който притежава инвентара.
 - Доставчикът получава поръчки от продавача и отговаря за опаковането и изпращането на продуктите директно до клиента.

Ползи от Dropshipping:

- **Нисък риск за стартиране на бизнес:**
 - Тъй като не е необходимо да се закупува инвентар предварително, рискът от загуби, свързани с непродадени продукти, е минимален.
- **Гъвкавост на продуктовия каталог:**
 - Продавачът може лесно да добавя или премахва продукти от каталога без нужда от управление на физически стоки.
- **Лесно тестване на пазара:**
 - Dropshipping позволява бързо тестване на различни продукти или ниши, за да се идентифицират най-печелившите.
- **Без необходимост от физически склад:**
 - Липсата на нужда от складови пространства намалява разходите за управление и поддръжка.
- **Възможност за глобален достъп:**
 - Системата позволява на продавачите да достигнат до клиенти от цял свят, тъй като доставчиците могат да обработват поръчки в международен мащаб.
- **Фокус върху маркетинга и клиентското обслужване:**



- Тъй като доставката и инвентарът са делегирани на доставчика, продавачите могат да се концентрират върху изграждането на марка и ангажиране на клиентите.
- **Лесна мащабируемост:**
 - Няма ограничение за броя на продуктите, които могат да се продават, защото не се изисква физическо управление на инвентар.

Как чрез проекта се улеснява начинът на работа:

1. Системата автоматично синхронизира продуктова база на дистрибутора – Denev Computers – с продуктовете бази на съответните доставчици-партньори.
2. Системата ще притежава административен панел за управление на поръчки, потребители, инвентар, предлагани услуги.
3. При обработка на поръчка, системата ще позволява детайлите за нея да се изпратят до доставчик автоматично.
4. Системата предлага каталог и актуален ценоразпис на услуги. Начин за контакт между фирма и клиент.
5. Потребителите могат да следят статуса на своите поръчки и ремонти.

3.2 Описание на необходимостта от и реализация на свързване-интеграция с други системи

Интеграцията с други системи е ключова част от ефективното управление на бизнес процесите в модерната електронна търговия. Тя позволява автоматизация, подобрява обслужването на клиентите и създава по-гъвкава и скалируема инфраструктура. Ето защо е необходима и как допринася за успеха на проекта:

3.2 Описание на необходимостта от и реализация на свързване-интеграция с други системи

Интеграцията с други системи е ключова част от ефективното управление на бизнес процесите в модерната електронна търговия. Тя позволява автоматизация, подобрява обслужването на клиентите и създава по-гъвкава и скалируема инфраструктура. Ето защо е необходима и как допринася за успеха на проекта:

1. Системи за онлайн плащане

- **Необходимост:** Онлайн плащанията са основен компонент на всяка платформа за електронна търговия. Интеграцията с платежни системи като Stripe, PayPal или други осигурява сигурни и удобни методи за плащане.
- **Ползи:**
 - Увеличава доверието на клиентите чрез сигурни транзакции.
 - Предоставя различни опции за плащане (кредитни карти, електронни портфейли и др.).



- Автоматизира обработката на плащанията, което намалява човешките грешки.

2. Системи за известия

- **Необходимост:** Комуникацията с клиентите в реално време е важна за удовлетворението им. Интеграцията със системи за известия, като имейл платформа, позволява информирание на клиентите за статуса на поръчките, промоции и други.
- **Ползи:**
 - Подобрява ангажираността на клиентите.
 - Намалява неяснотата за статусите на поръчките.
 - Предоставя персонализирано изживяване.

3. Системи за менажиране на инвентар

- **Необходимост:** За бизнеса с dropshipping е критично продуктовият каталог и наличностите да са синхронизирани в реално време с доставчиците. Интеграцията със системи за управление на инвентар осигурява актуализация на наличностите и цените.
- **Ползи:**
 - Минимизира риска от продажба на неналични продукти.
 - Автоматизира управлението на инвентара.
 - Осигурява точност и прозрачност в предлаганите продукти.

4. Google API и други външни услуги

- **Необходимост:** Използването на външни API, като Google Maps (за геолокация и доставка), Google Analytics (за анализ на потребителското поведение) и Google OAuth (за вход с Google акаунт), подобрява функционалността на платформата.
- **Ползи:**
 - Оптимизира процеса на доставка чрез точни локации.
 - Осигурява детайлен анализ на потребителската активност.
 - Улеснява регистрацията и входа на клиентите чрез социални мрежи.



4. Описание със стандартни методи на функционалността, обектите и потребителски случаи, които ще се реализират

1. Каталог на продукти и услуги

- Показване на категории и подкатегории.
- Филтриране на продукти по цена, характеристики и други атрибути.
- Препоръка на продукти въз основа на популярност.
- Показване на услуги и техните цени.

2. Управление на поръчки

- Добавяне на продукти в количката.
- Финализиране на поръчката.
- Заплащане на продукти чрез интеграция с платежната система Stripe.
- Уведомяване за статус на поръчка чрез имейл съобщения.
- Автоматично актуализиране на наличностите след направена поръчка.

3. Потребителска кошница

- Добавяне, премахване и редактиране на артикули в количката.
- Изчистване на количката.
- Проследяване на общото количество и стойност на продуктите в количката.

4. Потребителски профил и аутентикация

- Регистрация на потребители с валидиране на данните.
- Вход и изход чрез локална стратегия на Passport.js.
- Достъп до защитени ресурси само за аутентикирани потребители.
- Възможност за преглед на поръчки в секция „Моите поръчки“.

5. Административни функции

- Управление на продукти, услуги, категории и поръчки чрез административен панел.
- Автоматична синхронизация с външни бази данни (например, XML файл за продукти от доставчици).
- Управление на статусите на поръчките (например, „Обработка се“, „Изпратена“, „Завършена“).

6. Контакт и поддръжка

- Форма за контакт, която позволява на потребителите да изпращат съобщения.
- Използване на Google Maps API за визуализация на местоположението на офиса.

7. Интеграции



- Интеграция с платежната система Stripe за обработка на плащания.
- Системи за известия (имейли) чрез Nodemailer.
- Извличане на данни от XML файлове и синхронизация с базата данни.

Потребителски случаи:

1. Потребителски случаи за клиентите

- **Търсене на продукти:** Клиентът разглежда категории, използва филтри и намира нужния продукт.
- **Добавяне в количката:** Клиентът добавя избран продукт в количката и проверява общата стойност.
- **Финализиране на поръчка:** Клиентът попълва данни за поръчката и избира метод на плащане.
- **Проследяване на поръчки:** Клиентът следи статуса на своята поръчка през профила си.
- **Изпращане на съобщение:** Клиентът използва формата за контакт за запитвания.
- **Запазване на продукт в любими:** Клиентът запазва продукт в любими с цел да има бърз достъп до него.
-
- **Създаване на профил:** Клиентът създава профил в системата.
- **Влизане в профил:** Клиентът влиза в своя профил.

2. Потребителски случаи за администраторите

- **Управление на инвентара:** Администраторът добавя, редактира или премахва продукти и услуги.
- **Обработка на поръчки:** Администраторът променя статусите на поръчките.
- **Синхронизация на данни:** Системата автоматично синхронизира каталога с външни източници.

3. Защитени действия

- **Достъп до профил:** Само аутентифицирани потребители могат да разглеждат поръчките си или да редактират данни.
- **Промяна на количката:** Потребителят може да добавя или премахва артикули, но с проверки за наличност.

5. Описание на Базата от Данни -- бр. отдели, категории, стоки/услуги



Основни таблици/колекции:

- Колекция **products** – Продукти
- *Продуктите са извлечени от XML файл с продукти.*
Стойностите към полетата се попълват автоматично.

<i>Име на полето</i>	<i>Тип</i>	<i>Задължително</i>	<i>Описание</i>
_id	Number	Да	Идентификационен номер на продукт.
category	String	Да	Представява String, съдържащ категории и подкатегории, в които попада продукта, разделени със запетаи.
image	Array of Strings	Не	Изображения на продукта под формата на URL адреси.
title	String	Да	Име на продукта.
price	Number	Да	Текуща цена на продукт.
old_price	Number	Да	Оригинална цена на продукт, преди намаление.
instock	Number	Не	Наличност на продукта.



part_number	Number	Не	Номер на продукт, код.
attributes	Array of key-value pairs	Не	Атрибути, технически характеристики на продукт.
features	String	Не	Атрибути, технически характеристики на продукт, под текстов формат с разделител между атрибутите.
order_count	Number	Не (със стойност по подразбиране 0)	Брой колко пъти е бил поръчван даденият продукт. Увеличава се при направена поръчка, съдържаща продукта.

- **Колекция *services* - Услуги**

<i>Име на полето</i>	<i>Тип</i>	<i>Задължително</i>	<i>Описание</i>
_id	ObjectId (генерирано от MongoDB)	Да	Идентификационен номер на услуга.
name	String	Да	Описателно наименование на услугата.



price	Number	Не	Цена на услуга, ако е със липсваща стойност – цената е по договаряне.
category	String	Да	Категория услуги, към които спада дадената услуга.

- **Колекция *categories* - Категории**

<i>Име на полето</i>	<i>Тип</i>	<i>Задължително</i>	<i>Описание</i>
_id	ObjectId (генерирано от MongoDB)	Да	Идентификационен номер на категория.
main	String	Да	Наименование на главна категория.
image	String	Да	Описателно Изображение под форма на URL адрес.
subcategories	Array of Strings	Да	Категория услуги, към които спада дадената услуга.
type	String	Да	Дали категорията спада към отдел продукти или отдел услуги.

- **Колекция *orders* - Поръчки**



Име на полето	Тип	Задължително	Описание
_id	ObjectId (генерирано от Mongodb)	Да	Идентификационен номер на категория.
userId	ObjectId (генерирано от Mongodb)	Не	Идентификационен номер на потребител, направил поръчката (null ако е поръчано без регистрация, като гост)
fullName	String	Да	Име и фамилия на получател
phone	String(10)	Да	Телефонен номер на получател.
email	String	Да	Електронна поща на получател.
deliveryType	String	Да	Тип на доставка – enum между 4 посочени избора.
paymentType	String	Да	Начин на плащане – с наложен платеж или онлайн.
billingAddress	String	Не	Адрес за доставка. Става задължително при избор на опция,



			различна от офис на Denev Computers.
city	String	Не	Град за доставка. Става задължително при избор на опция, различна от офис на Denev Computers.
zip	String	Не	Пощенски код. Става задължително при избор на опция, различна от офис на Denev Computers.
items	Array of objects	Да	Масив с продуктите в поръчката.
totalQuantity	Number	Да	Количество продукти в поръчката.
totalPrice	Number	Да	Цена на поръчка общо.
status	String(Enum)	Да	Статус на поръчка.
createdAt	Date	Не (default Date.now)	Кога е създадена поръчката.

- **Схема cartItem** – продукт в количка



- Схемата е дефиниция на обект, който се пази в колекция. В случая *cartItem* се изпълва в колекциите *Количка* и *Поръчка* – самия продукт в количка не съществува самостоятелно, извън тези колекции.

Име на полето	Тип	Задължително	Описание
_id	Number	Да	Идентификационен номер на артикула.
title	String	Да	Наименование на артикула.
image	String	Не	Изображение на артикула под форма на URL адрес.
quantity	Number	Да	Количество - този артикул колко пъти се среща в количката/поръчката.
unitPrice	Number	Да	Единична цена на артикула.
price	Number	Да	Единична цена на артикул умножена по количеството.
availableCount	Number	Да	Това поле следи количеството артикул, добавен в количка, дали съответства на наличността на



			продукта. Съответно позволява или забранява на потребител да добави артикул в количка.
--	--	--	--

- **Колекция *carts* – Количка**

<i>Име на полето</i>	<i>Тип</i>	<i>Задължително</i>	<i>Описание</i>
_id	ObjectId (генерирано от MongoDB)	Да	Идентификационен номер на количка.
sessionId	String	Да	Идентификатор на потребителска сесия, свързващ количката със сесия.
items	Array of objects	Не	Артикули в количката с необходима информация за тях.
totalQuantity	Number	Не (по подразбиране 0)	Общ брой на всички артикули в количката.
totalPrice	Number	Не (по подразбиране 0)	Обща цена на всички артикули в количката.

- **Колекция *sessions* – Сесии**

- Тази колекция се създава автоматично от *express-session* и *connect-mongo*.



<i>Име на полето</i>	<i>Тип</i>	<i>Задължително</i>	<i>Описание</i>
_id	String	Да	Идентификатор на потребителска сесия.
expires	Date	Да	Докога е валидна сесията
session	Object	Да	Информация за сесията, като бисквитки, конфигурации

• **Колекция *messages* – Съобщения**

<i>Име на полето</i>	<i>Тип</i>	<i>Задължително</i>	<i>Описание</i>
_id	ObjectId (генерирано от MongoDB)	Да	Идентификатор на съобщение.
name	String	Да	Име и фамилия на запитващия.
email	String	Да	Имейл на запитващия.
message	String	Да	Съобщение на запитващия.

• **Колекция *users* – Потребители**

<i>Име на полето</i>	<i>Тип</i>	<i>Задължително</i>	<i>Описание</i>
----------------------	------------	---------------------	-----------------



_id	ObjectId (генерирано от Mongodb)	Да	Идентификатор на потребител.
fullName	String	Да	Име и фамилия на потребител.
email	String	Да, и уникално	Имейл на потребителя.
password	String	Да	Парола на потребителя.

6. Описание на категориите и отделите

1. Създаване и показване на каталог в БД

Създаването на колекции в нерелационната база данни се извършва чрез дефиниране на **схеми** и **модели**. Те са дефинирани в бекенд частта на приложението, в папка `models`. За да се работи с базата данни, приложението се свързва с MongoDB.

1. Създаване на схема (Schema):

Схемата е структурата, която дефинира какви полета ще съдържа документа в колекцията и какъв тип данни се очаква за всяко поле. В Mongoose схемите също така позволяват добавяне на валидации, дефолтни стойности, уникални ограничения и други функционалности.

2. Създаване на модел (Model):

Моделът е интерфейс за взаимодействие с колекцията в MongoDB. Моделът се създава въз основа на схемата и предоставя методи за CRUD (Create, Read, Update, Delete) операции.

3. Създаване на документи в колекцията:

С помощта на модела се създават нови документи и се записват в базата данни. С модела може също така да се извличат данни от колекцията, да се актуализират съществуващи документи.



Продуктовият каталог е извлечен от линк към XML файл, предоставен от партньор-доставчик на фирмата. Съответният линк се актуализира на всеки кръгъл час, което създава нужда системата също да актуализира продуктивия си каталог през час.

Данните от файла се извличат и се запазват в базата данни след изпълняване на функция във файла `“sync.js”`. Насрочена е тя да се изпълнява на всеки кръгъл час.

Каталогът с услуги е извлечен от съществуващ такъв, под формата на CSV файл, който бе въведен директно в платформата на MongoDB Compass.

2. Описание на съхранените процедури

Controllers

Контролерите съдържат бизнес логиката на приложението, както и логиката за обработка на заявките, изпратени от фронтенд чатта. Те действат като посредници между маршрута (route) и данните. Контролерите приемат входящите данни от заявката, обработват ги (например достъпват базата данни, извършват изчисления) и връщат подходящ отговор към клиента.

Routes

Маршрутите (Routes) определят как приложението отговаря на определени HTTP заявки (GET, POST, PUT, DELETE) за конкретни URL пътища. Те се използват, за да свързват заявките с подходящите контролери. Маршрутите осигуряват ясна структура и организация на заявките в приложението.

Products

Маршрут (route)	HTTP Заявка (request)	Контролер функция	Описание
/top	GET	getTopProducts	Извлича препоръчаните продукти, на база колко пъти са били продавани през сайта.
/	GET	getAllProducts	Извлича всички продукти.
/category	GET	getAllCategories	Извлича всички уникални категории, към които принадлежат продуктите. (testing цели)
/by-category	GET	getProductsByCategory	Извлича всички продукти, принадлежащи на категория или подкатегория. При приложени филтри във



			фронтенда, филтрира продуктите в категорията и изпраща филтрираните продукти, попадащи в категорията.
/by-category/prices	GET	getPriceRangeByCategory	Помощна функция за филтърът по цена. По подадена категория, намира най-високата цена на продукт, и най-ниската такава, за да ги изобрази на динамичния слайдър.
/by-category/product/:id	GET	getProductById	По подадено ID извлича информация за продукт.
/by-category/attributes	GET	getProductAttributes	Помощна функция за филтриране по атрибути (характеристики). По подадени категория или подкатегория, намира и връща какви атрибути се срещат сред продуктите в тази категория, както и какви са уникалните стойности към тях, с цел да бъдат изобразени.

Products

Общ маршрут за всички маршрутизатори: api

Общ маршрут за продуктите: /products

Маршрут (route)	HTTP Заявка (request)	Контролер функция	Описание
/top	GET	getTopProducts	Извлича препоръчаните продукти, на база колко пъти са били продавани през сайта.
/	GET	getAllProducts	Извлича всички продукти.
/category	GET	getAllCategories	Извлича всички уникални категории, към които принадлежат продуктите. (testing цели)
/by-category	GET	getProductsByCategory	Извлича всички продукти, принадлежащи на категория или подкатегория. При приложения



			филтри във фронтенда, филтрира продуктите в категорията и изпраща филтрираните продукти, попадащи в категорията.
--	--	--	--

Services

Общ маршрут за услугите: /services

Маршрут (route)	HTTP Заявка (request)	Контролер функция	Описание
/	GET	getServiceByCategory	По подадена категория връща всички услуги, които попадат в нея.

Contact

Маршрут (route)	HTTP Заявка (request)	Контролер функция	Описание
/contact	POST		По подадени данни за потребител и подадено съобщение, ги запазва в базата данни.

Categories

Общ маршрут: /categories

Cart

Общ маршрут: /cart

Маршрут (route)	HTTP Заявка (request)	Контролер функция	Описание
/	POST	addToCart	Добавя продукт към количката. 1. По подадено id на продукт, намира съответния продукт. 2. Чрез бисквитка пазира id на сесията, намира количката, към която трябва да се прибави продукта, или създава сесия ако още не е инициализирана такава.



			3.Ако продуктът вече го има в количката, и наличността му позволява увеличаване, увеличава се количеството му, Ако все още няма такъв продукт в количката, се запазва информация за него.
/	GET	getCart	По sessionId, намира съответстващата количка и я връща на потребителя.
/remove	POST	removeOneFromCart	Премахва продукт от количката. 1.По подадено id на продукт, намира съответния продукт. 2.Чрез бисквитка пази id на сесията, намира количката, от която трябва да се премахне продукта. 3.Ако продуктът го има в количката, и количеството е по-голямо от 1, намалява се количеството. Ако количеството е точно 1, всецяло се изтрива артикула от кошницата.
/remove-item	POST	removeItemFromCart	Премахва продукт, без значение неговото количество, от кошницата.
/clear-cart	DELETE	clearCart	Изтрива количката.

Payment

Маршрут (route)	HTTP Заявка (request)	Контролер функция	Описание
/create-checkout-session	POST	createSession	Интеграция с платежната система Stripe. ID на поръчката се подава като параметър в URL с цел да се следи нейния статус. 1.Създава се сесия със stripe



			<p>2. Подават се продуктите от количката, предварително преобразувани в подходящ формат</p> <p>3. Указани са валута, тип плащане</p> <p>4. Пренасочва към страница след успешна транзакция, или друга за неуспешна транзакция</p>
--	--	--	---

Order

Общ маршрут: /orders

Маршрут (route)	HTTP Заявка (request)	Контролер функция	Описание
/	POST	saveOrder	Запазва информация за поръчка в базата данни. Изпраща имейл за потвърдена поръчка.
/ product-order-count	POST	updateOrderCount	Когато е поръчан даден продукт, намалява се неговата наличност и се увеличава броят order_count (показател за препоръчан продукт)
/:orderId	DELETE	deleteOrder	Изтрива поръчка от БД.
/:orderId/updateStatus	POST	setOrderStatus	По подадени id на поръчка и нов статус, променя статуса й.
/my-orders	GET	getUserOrders	Извлича потребителя от сесията, и ако има авторизиран такъв, връща списък с неговите поръчки.

Auth

За менажиране на потребителски профили.

Общ маршрут: /auth



Маршрут (route)	HTTP Заявка (request)	Контролер функция	Описание
/register	POST	Register	По подадени данни, проверява има ли съществуващ такъв, ако няма го запазва в базата.
/login	POST	Login	Проверява предоставените от потребителя данни (потребителско име и парола) с помощта на дефинирана стратегия. Ако аутентикацията е успешна, <code>req.login(user)</code> създава сесия за потребителя и го маркира като логнат.
/login	GET	Logout	Затваря текущата сесия, като премахва информацията за потребителя от сесията.
/profile	GET	getProfile	Проверява дали потребителят е аутентикиран (т.е. има активна сесия). Ако потребителят е аутентикиран, връща защитен ресурс заедно с информацията за потребителя от сесията (<code>req.user</code>).

7. Етапи на разработка (RUP)

1. Планиране

- Проведени интервюта с представители на Denev Computers, проучване начина на работа
- Заявяване и получаване на достъп до XML линк с продукти на партньор на фирмата.
- Изготвяне на начална визия за проекта
- Определяне на функционални изисквания
- Определяне данните, които ще се съхраняват

2. Детайлизиране

- Изготвяне на дизайн на потребителския интерфейс

С цел дизайн на потребителския интерфейс са предприети следните стъпки:

2.1 Изследвания и анализ на данни

2.1 а) Целева аудитория

Профил на потребителите:



Мъже, на възраст **25-50 години**.

Работят на пълно работно време и разчитат на компютри за професионалните си задължения.

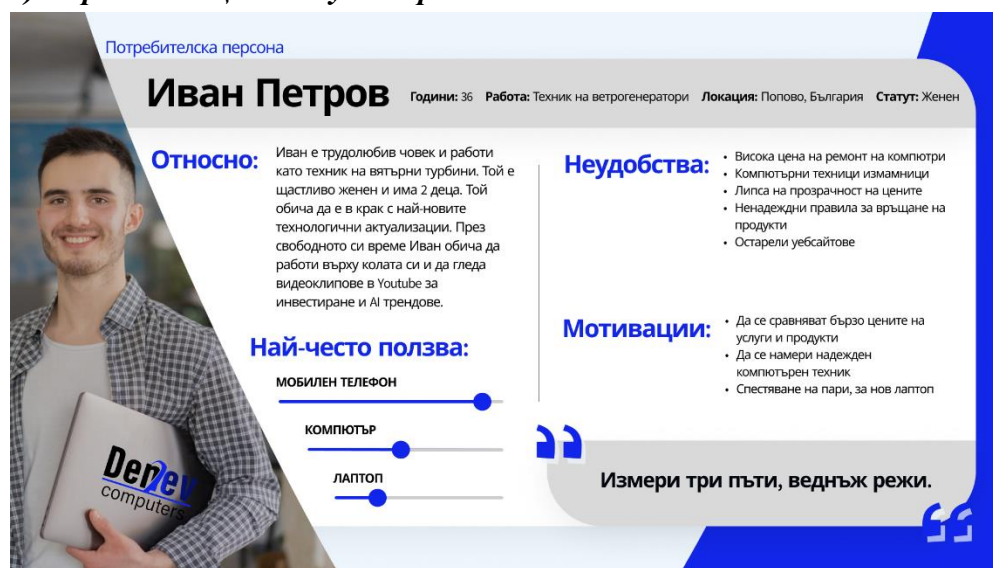
При технически проблеми се обръщат към DenevComputers, познавайки компанията от препоръки.

Поведение:

Предпочитат да пазаруват онлайн, основно от eMAG.

Ползват мобилни устройства за браузване.

2.1 б) Персона на целева аудитория:



Потребителска персона

Иван Петров Години: 36 Работа: Техник на ветрогенератори Локация: Попово, България Статус: Женен

Относно: Иван е трудолюбив човек и работи като техник на вятърни турбини. Той е щастливо женен и има 2 деца. Той обича да е в крак с най-новите технологични актуализации. През свободното си време Иван обича да работи върху колата си и да гледа видеоклипове в Youtube за инвестиране и AI трендове.

Неудобства:

- Висока цена на ремонт на компютри
- Компютърни техники измамници
- Липса на прозрачност на цените
- Ненадеждни правила за връщане на продукти
- Остарели уебсайтове

Мотивации:

- Да се сравняват бързо цените на услуги и продукти
- Да се намери надежден компютърен техник
- Спестяване на пари, за нов лаптоп

Най-често ползва:

- МОБИЛЕН ТЕЛЕФОН
- КОМПЮТЪР
- ЛАПТОП

Измери три пъти, веднъж режи.

2.2 Извършен конкурентен анализ

Функционалности и предлагани услуги:

DenevComputers изостава в мобилната оптимизация, докато конкуренти като eMAG вече предлагат мобилни приложения.

Уникалната услуга за сглобяване на компютри е предимство, което може да бъде по-добре комуникирано.

Възможности за диференциация:

Фокус върху персонализирани услуги и подобро потребителско изживяване.

- Избор на среда за разработка и технологии за реализация
- Дизайн на базата данни

3. Изграждане

- Разработване на начална страница
- Разработване на каталог продукти



- Реализиране потребителска кошница чрез сесии
 - Реализиране на поръчка
 - Интеграция с платежна система
 - Интеграция с имейл система
 - Реализиране създаване и менажиране на потребителски профили
4. Предаване
- Системата предстои да се пусне в експлоатация в интернет пространството.

8. Стартиране на приложението

1. Клонирание/разархивиране на проекта

2. Настройка на бекенда

1. Навигация до директорията на бекенда:

```
cd DenevComputers/server
```

2. Инсталиране на зависимости:

```
npm install
```

3. Конфигуриране на .env файл:

- Създайте .env файл в корена на бекенд директорията и добавете необходимите настройки, като:

```
PORT=8081  
MONGO_URI=your_mongodb_connection_string  
STRIPE_SECRET_KEY=your_stripe_secret_key
```

4. Стартиране на сървъра:

```
npm run dev
```

3. Настройка на фронтенда

1. Навигация до директорията на фронтенда:

```
cd DenevComputers/ecommerce-frontend
```

2. Инсталиране на зависимости:

```
npm install
```

3. Конфигуриране на .env файл:



- Създайте `.env` файл в директорията на фронтенда и добавете:

```
REACT_APP_API_URL=http://localhost:5173
```

```
REACT_APP_GOOGLE_API_KEY=your_google_api_key
```

```
REACT_APP_STRIPE_PUBLIC_KEY=your_stripe_public_key
```

4. Стартиране на фронтенда:

```
npm run dev
```

4. *Стартиране на MongoDB*

- Уверете се, че MongoDB е стартиран локално или използвайте облачна услуга като MongoDB Atlas.

5. *Достъп до приложението*

- Фронтенд приложението ще бъде достъпно на:

```
http://localhost:5173
```