



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ _____

КАФЕДРА _____ КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ _____

НАПРАВЛЕНИЕ ПОДГОТОВКИ _____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Веб-приложение для заказа еды «Delivery Borscht»

Студент ИУ6-52Б
(Группа)

(Подпись, дата) Д.В. Лабзунова
(И.О. Фамилия)

Руководитель курсовой работы

(Подпись, дата) _____
(И.О. Фамилия)

Консультант

(Подпись, дата) _____
(И.О. Фамилия)

2021 г.

Вставить задание на курсовую

Реферат

Расчетно-пояснительная записка 32 страницы, 3 части, 16 рисунков, 3 таблицы, 5 источников, 1 приложение.

ВЕБ-СЕРВИС, ВЕБ-ПРИЛОЖЕНИЕ, ДОСТАВКА, ЕДА, РЕСТОРАН, КАФЕ.

Объектом разработки является веб-сервис доставки еды «Delivery Borscht».

Цель работы – проектирование и реализация веб-сервиса доставки еды, позволяющего заказывать на дом еду из кафе и ресторанов и имеющего функционал фильтрации ресторанов по различным признакам (средний чек, рейтинг, скорость доставки) и также функционал сравнения нескольких корзин с товарами.

В результате работы был спроектирован сервис, позволяющий для владельца заведения зарегистрировать свой ресторан, а для клиентов – заказать еду. А также было проведено тестирование программного продукта методами белого и черного ящика и юзабилити-тест.

Пользователями данного приложения могут быть все, кого интересует доставка еды на дом, а также владельцы заведений общественного питания.

Содержание

Введение.....	6
1. Анализ требований и уточнение спецификаций.....	7
1.1. Анализ задания и выбор технологии, языка и среды разработки.....	7
1.2. Разработка диаграммы вариантов использования.....	8
1.3. Анализ хранимой информации и выбор способа ее хранения.....	10
1.4. Разработка диаграмм деятельности.....	13
2. Проектирование структуры и компонентов программного продукта.....	18
2.1. Разработка интерфейса пользователя.....	18
2.1.1. Посторонние диаграммы состояний интерфейса.....	18
2.1.2. Разработка форм интерфейса.....	20
2.2. Разработка структурной схемы программного продукта.....	25
3. Выбор стратегии тестирования и разработка тестов.....	26
3.1. Тестирование структурным контролем.....	26
3.2. Unit-тесты.....	28
3.3. Usability-тест.....	28
Заключение.....	30
Список используемых источников.....	31
Приложение А. Техническое задание.....	32
Приложение Б. Руководство пользователя.....	32

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ТЗ – техническое задание.

ООП – объектно-ориентированное программирование.

БД – база данных.

СУБД – система управления базами данных.

Go – Golang, язык программирования, который был разработан корпорацией Google.

JS – JavaScript, язык, обычно используемый как встраиваемый для программного доступа к объектам приложений.

UI – User Interface, (дословно «пользовательский интерфейс») — то, как выглядит интерфейс и то, какие физические характеристики приобретает.

Фронтенд (англ. front-end) — клиентская сторона пользовательского интерфейса к программно-аппаратной части сервиса.

Бэкенд (англ. back-end) — программно-аппаратная часть сервиса, отвечающая за функционирование его внутренней части.

Презентер (англ. presenter) — объект, который управляет отображением данных.

CI/CD – (continuous integration/continuous delivery) - это комбинация непрерывной интеграции (continuous integration) и непрерывного развертывания программного обеспечения.

Корзина онлайн-магазина – страница, на которую клиент отправляет товары, чтобы затем приобрести их.

Введение

Данная работа посвящена проектированию и разработке веб-сервиса доставки еды, способного функционировать в браузерах любых устройств. Разрабатываемый веб-сервис может быть использован владельцами заведений общественного питания, а также любыми желающими заказать еду с доставкой на дом. Данный сервис позволяет выбрать подходящий ресторан, собрать в нем корзину с продуктами, ввести данные об адресе доставки, заказать еду и отслеживать стадии обработки заказа.

В ходе выполнения работы были рассмотрены аналоги данного предложения, позволяющие заказать еду на дом.

Актуальность разработки заключается в том, что в рассмотренных аналогах нет системы фильтрации, по которой можно было бы сократить список интересующих ресторанов, а также функционала сравнения нескольких корзин с продуктами из разных ресторанов, чтобы ускорить и облегчить выбор еды для заказа.

1. Анализ требований и уточнение спецификаций

1.1. Анализ задания и выбор технологии, языка и среды разработки

Методологией программирования было выбрано объектно-ориентированное программирование. ООП является основой всех современных приложений и имеет удобное и практическое применение. Она основана на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса.

Для разработки бэкенда веб-сервиса был использован язык Golang. Данный язык отличается своей скоростью и удобной организации многопоточности, а также микросервисной архитектуры.

Для разработки фронтенда веб-сервиса был использован язык JavaScript, как самый популярный язык для разработки фронтенда приложений. Верстка страниц выполнена с помощью HTML и CSS.

Средой разработки были выбраны одни из самых распространенных IDE от компании JetBrains для разработки: для разработки бэкенда на Go – Goland, для разработки фронтенда на JavaScript, HTML и CSS - Webstorm. Данные IDE имеют удобный пользовательский интерфейс. В числе их плюсов: интерфейс с вкладками, возможность писать в двух окнах на одном мониторе, умное авто-завершение кода, быстрая навигация, подсказки параметров функций, обнаружение ошибок синтаксиса и комментарии к ним.

В качестве СУБД был выбран PostgreSQL за ее скорость, бесплатность, большое количество документаций и прочей информации в интернете, наличие дополнительных библиотек (в нашем случае нам нужна библиотека Postgis для работы с адресами),

Веб-сервис имеет микросервисную архитектуру. Микросервисы - это небольшие, легкие модули, каждый из которых хорошо выполняет определенную задачу и взаимодействует с другими модулями при создании более крупных приложений. Таким образом, правильно построенная микросервисная архитектура позволяет ускорить работу приложения, распределяя нагрузку на несколько серверов.

Каждый микросервис выполнен с соблюдением Clean Architecture [6], позволяющей архитектуре не зависеть от UI и базы данных, а также быть легко тестируемой. Это достигается разделением на слои и следованием Dependency Rule (правилу зависимостей). Dependency Rule говорит нам, что внутренние слои не должны зависеть от внешних. То есть наша бизнес-логика и логика приложения не должны зависеть от презентеров, UI, баз данных и т.п. Это правило позволяет строить системы, которые будет проще поддерживать, потому что изменения во внешних слоях не затронут внутренние слои. Выделяется три основных слоя: Entities (бизнес-логика общая для многих приложений), Use Cases (логика приложения), Interface Adapters (адаптеры между Use Cases и внешним миром).

1.2. Разработка диаграммы вариантов использования

Первый шаг проектирования веб-сервиса – определение основных вариантов его использования. Для этого была разработана диаграмма вариантов использования, отображающая основные варианты взаимодействия приложения и пользователя.

После анализа ТЗ были получены следующие варианты взаимодействия пользователя и приложения:

- авторизация;
- выставление фильтров для поиска подходящего заведения;
- добавление в корзину блюд;
- оформление заказа;
- отслеживание информации о заказе;
- общение с рестораном в чате.

Для владельцев ресторанов:

- авторизация;
- добавление блюд;
- управление заказами;
- общение с заказчиками в чате.

Рассмотрим самый распространенный вариант использования сервиса.

Таблица 1 – описание вариантов использования **Оформление заказа**

Название варианта	Оформление заказа
Цель	Оформить заказ на доставку еды с доставкой на дом
Действующие лица	Пользователь
Краткое описание	Пользователь добавляет в корзину блюда, заполняет данные о себе, делает заказ, отслеживает его статус.
Тип	Основной

Таблица 2 – Вариант использования **Оформление заказа**

Действие пользователя	Отклик системы
<p>1. Пользователь авторизуется</p> <p>Пользователь добавляет в корзину блюда.</p> <p>3. Пользователь добавляет блюда в корзину.</p> <p>5. Пользователь заполняет данные о себе, оформляет заказ.</p> <p>7. Пользователь заходит на страницу «мои заказы», следит за их статусом.</p>	<p>2. Отправляется запрос на авторизацию, пользователю присваивается сессия, записываемая в cookie.</p> <p>4. Сервис запоминает предметы в корзине пользователя.</p> <p>6. Заказ сохраняется в базе данных. Теперь он доступен к просмотру рестораном.</p> <p>8. Сервис предоставляет информацию о текущем статусе заказа и отправляет уведомления о его смене.</p>

Разработанная диаграмма вариантов использования представлена на рисунке 1.

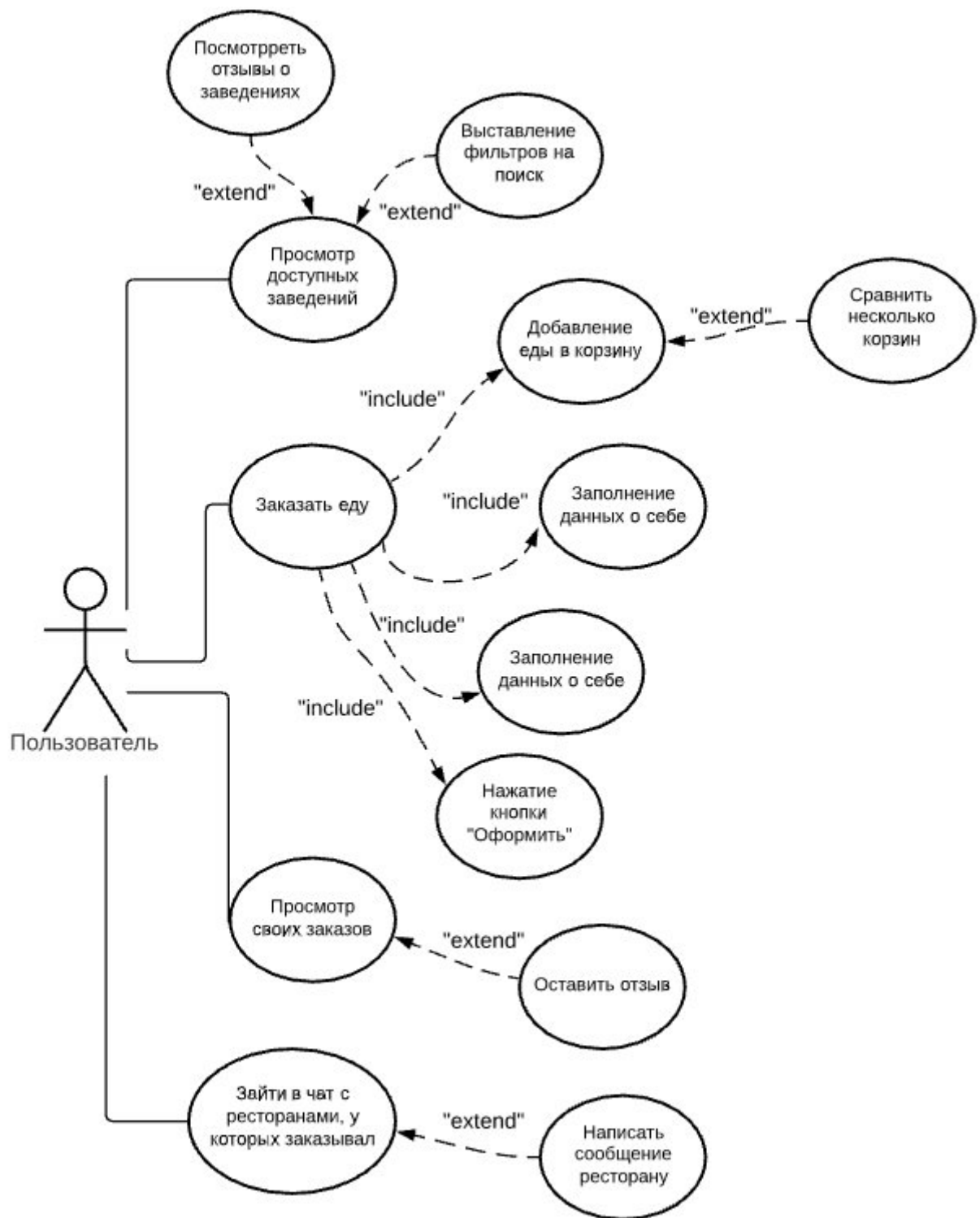


Рисунок 1 – Диаграмма вариантов использования

1.3. Анализ хранимой информации и выбор способа ее хранения

При анализе предметной области и технического задания была выявлена следующая информация для хранения:

- пользователь:

- адрес электронной почты;
 - телефон;
 - пароль;
 - фото;
 - адрес;
 - имя.
- ресторан:
 - описание;
 - стоимость доставки;
 - категория ресторана;
 - адрес электронной почты владельца;
 - телефон владельца;
 - пароль;
 - логотип;
 - адрес;
 - средняя оценка;
 - секции с едой.
- блюдо:
 - название;
 - описание;
 - изображение;
 - цена.
- адрес:
 - текстовый адрес;
 - широта;
 - долгота.
- заказ:
 - ресторан;
 - заказчик;
 - список блюд;
 - статус;
 - сумма;
 - время доставки;
 - время оформления заказа.

- корзина:
 - ресторан;
 - заказчик;
 - список блюд.

В данном списке информация уже сгруппирована по смысловым группам, на основе которых будут строиться таблицы в базе данных.

Для ускорения работы и нормализации базы данных были добавлены также связывающие таблицы блюдо-корзина, корзина-заказ и корзина-пользователь. Когда пользователь набирает блюда в корзину, блюдо добавляется в таблицу блюдо-корзина. Если пользователь сделал заказ, то корзина «открепляется» от него путем удаления ее из строки корзина-пользователь и «прилепляется» к заказу путем добавления ее в таблицу корзина-заказ.

Схема базы данных приведена на рисунке номер 2.

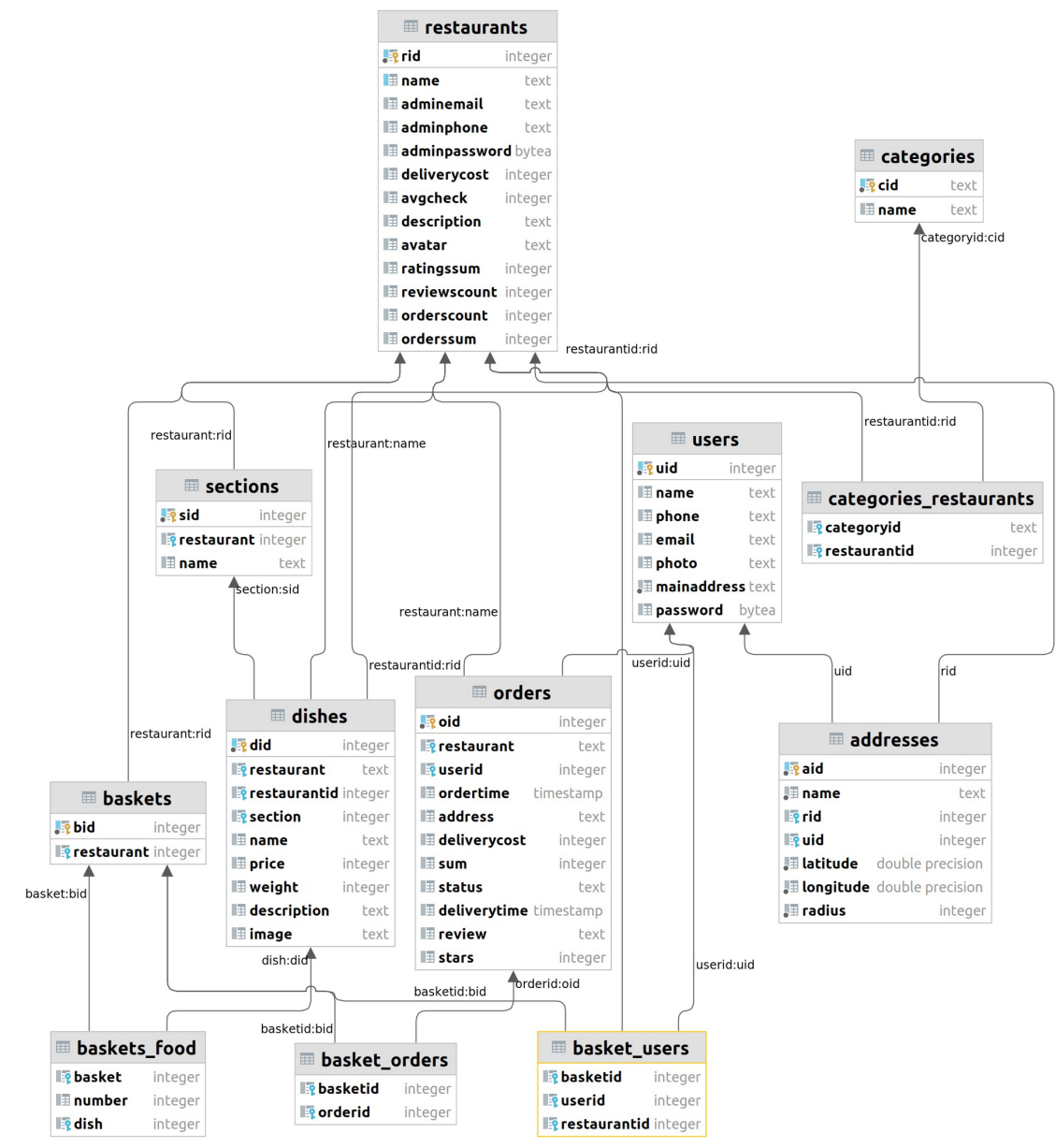


Рисунок 2 – Схема базы данных.

1.4. Разработка диаграмм деятельности

Для подробного описания взаимодействия пользователя и сервиса были разработаны диаграммы деятельности для таких процессов, как:

- заказ пиццы в ресторане со средним чеком до 2000р;
- создание своего ресторана и наполнение его информацией.

Диаграмма деятельности для заказа по заданным параметрам приведена на рисунке 3. Она показывает, что пользователь должен авторизоваться и отметить свой адрес для создания заказа. Ввод данных сопровождается валидацией. Далее пользователь может выбрать категорию кухни и выставить интересующий его фильтр (в данном случае –

средний чек). При оформлении заказа, если телефон пользователя и адрес были введены ранее, то они заполнятся автоматически. Если пользователь хочет исправить эти данные, то результат его ввода пройдет валидацию. В случае невалидных данных пользователю выведется уведомление о неверно указанных данных, и кнопка «оформить заказ» будет недоступна. После успешного ввода данных процесс создания заказа завершается.

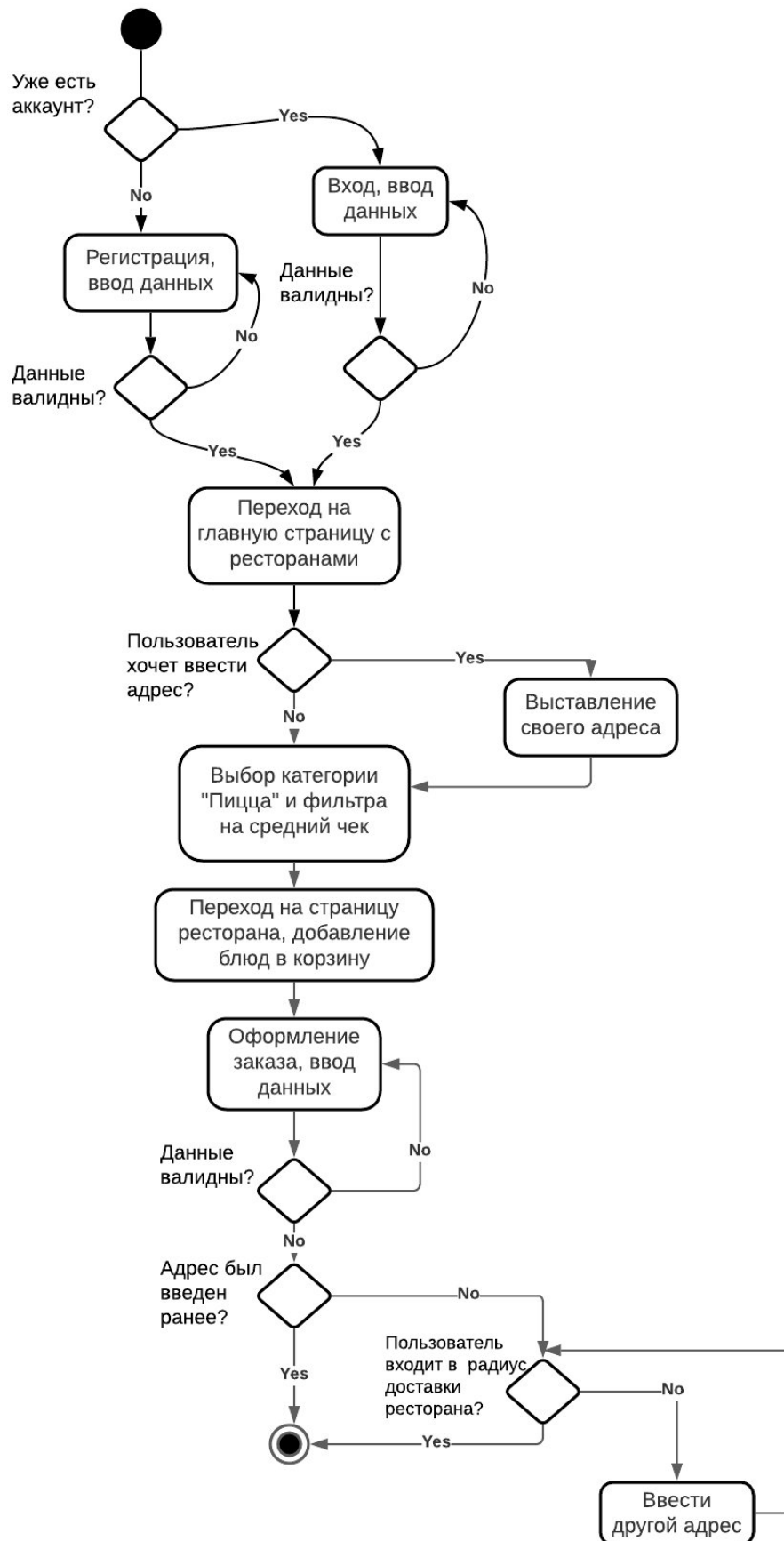


Рисунок 3 – Диаграмма деятельности для создания заказа с параметрами «Пицца из ресторана со средним чеком до 2000р»

Если пользователь не введет адрес до выбора ресторана, то может оказаться так, что он находится вне радиуса доставки ресторана, и тогда ему придется или ввести другой адрес, или выбрать другой ресторан. Радиус доставки ресторана указывает сам ресторан при регистрации.

Диаграмма деятельности по созданию своего ресторана и наполнению его информацией приведена на рисунке 4. Под «наполнением информацией» имеется в виду добавление категорий блюд и самих блюд.

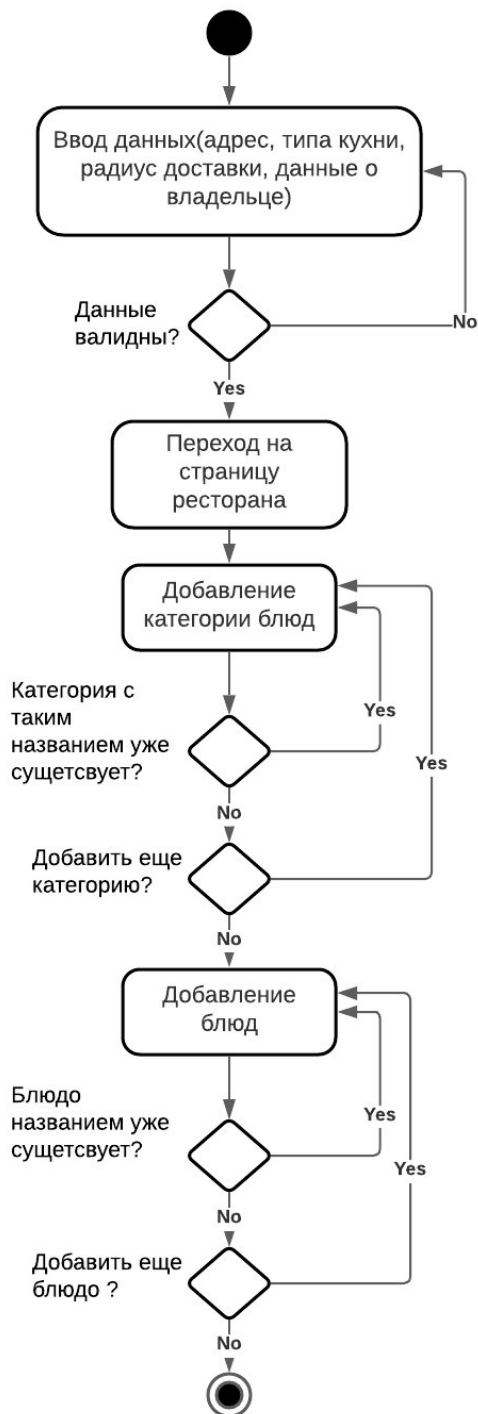


Рисунок 4 – Диаграмма деятельности для создания ресторана

Ввод данных о ресторане сопровождается валидацией и уведомлением о невалидных данных. Секций и блюд можно добавлять неограниченное количество, при этом их названия не должны повторяться в рамках одного ресторана. Для блюд можно добавить также фото и описание.

2. Проектирование структуры и компонентов программного продукта

2.1. Разработка интерфейса пользователя

В ходе разработки интерфейса были соблюдены определенные правила, сформулированные с целью того, чтобы интерфейс был понятным и простым для пользователя.

Интерфейс должен быть последовательным. Все возможные пользовательские действия должны быть логичными и уместными, и должны соответствовать диаграммам последовательности действий.

Должны присутствовать определенные «паттерны», к которым пользователи привыкли в силу того, что они используются повсеместно. Например, сверху страницы должна быть шапка так называемого «навбара», на которой находится логотип, авторизация и переход в профиль.

Интерфейс не должен быть «шумным». То есть не должно быть слишком много всего, чтобы пользователь с легкостью находил нужный функционал.

Ошибки валидации, а также любые другие ошибки должны быть понятно описаны пользователю, чтобы было понятно, что делать, когда возникла ошибка.

Легкая возможность вернуться назад и отменить свои действия. Эта функция уменьшает беспокойство, поскольку пользователь знает, что ошибки можно отменить. Легкое изменение действий стимулирует изучение незнакомых вариантов. Единицами обратимости могут быть одно действие, ввод данных или полная группа действий.

2.1.1. Посторонние диаграммы состояний интерфейса

Для описания возможных действий в программном продукте была составлена диаграмма состояний интерфейса, представленная на рисунке 5.

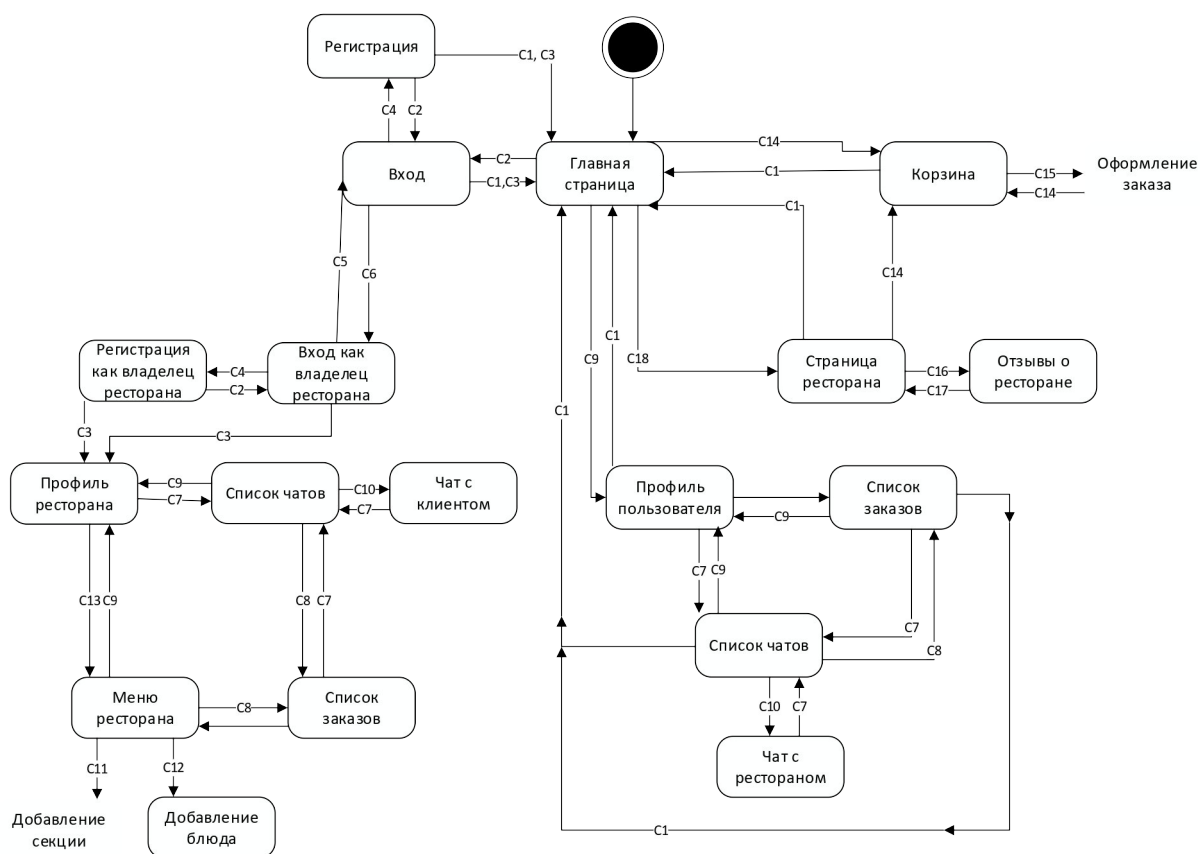


Рисунок 5 – Диаграмма состояний интерфейса

C1 – нажатие на логотип приложения. Переход на главную страницу.

C2 – нажатие кнопки «Вход». Переход на авторизацию.

C3 – нажатие войти/зарегистрироваться. Успешная авторизация.

C4 – нажатие на «я тут впервые». Переход на регистрацию.

C5 – нажатие на «у меня нет ресторана». Переход на авторизацию от имени .пользователя.

C6 – нажатие на «войти как владелец ресторана». Переход на авторизацию от имени ресторатора.

C7 – Нажатие на «Чаты».

C8 – Нажатие на «Заказы».

C9 – Нажатие на «Профиль».

C10 – Нажатие на конкретный чат.

C11 – Нажатие на кнопку добавления секций.

C13 – Нажатие на кнопку добавления блюд.

C14 – Нажатие на корзину.

C15 – Нажатие на «Оформить заказ».

C16 – Нажатие на «Отзывы».

C17 – Нажатие вне зоны окна с отзывами.

C18 – Нажатие на ресторан.

2.1.2. Разработка форм интерфейса

Интерфейс, как было написано выше, должен быть удобным и понятным. Также он не должен содержать слишком маленьких элементов или слишком много цветов. Разработанные страницы показаны на рисунках 6-15.

Главная страница содержит в себе список ресторанов и фильтры для более точного поиска. Также сверху – header, в котором можно указать адрес, перейти в корзину и профиль, если пользователь авторизован, или перейти на авторизацию, если еще нет.

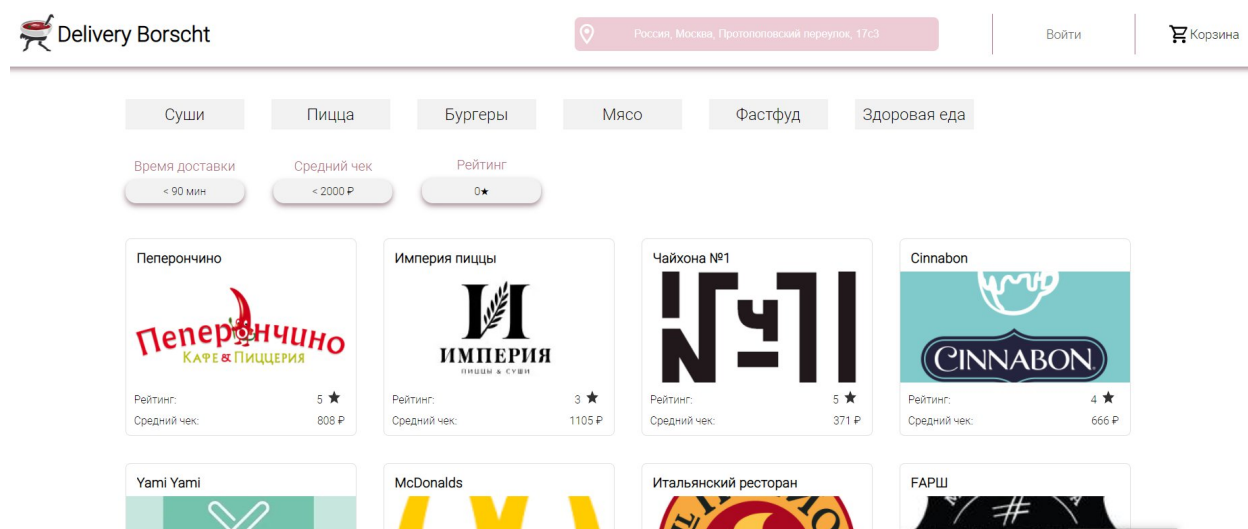
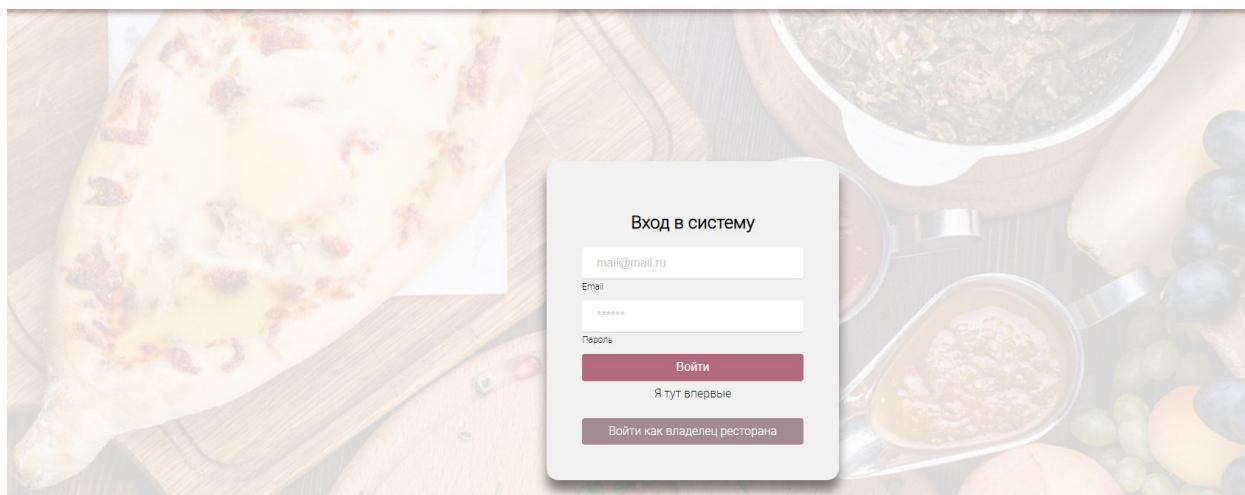


Рисунок 6 – Главная страница

Форма авторизации обычного пользователя и ресторана выглядят аналогично, а формы регистрации пользователя и ресторана отличаются вводимыми параметрами.



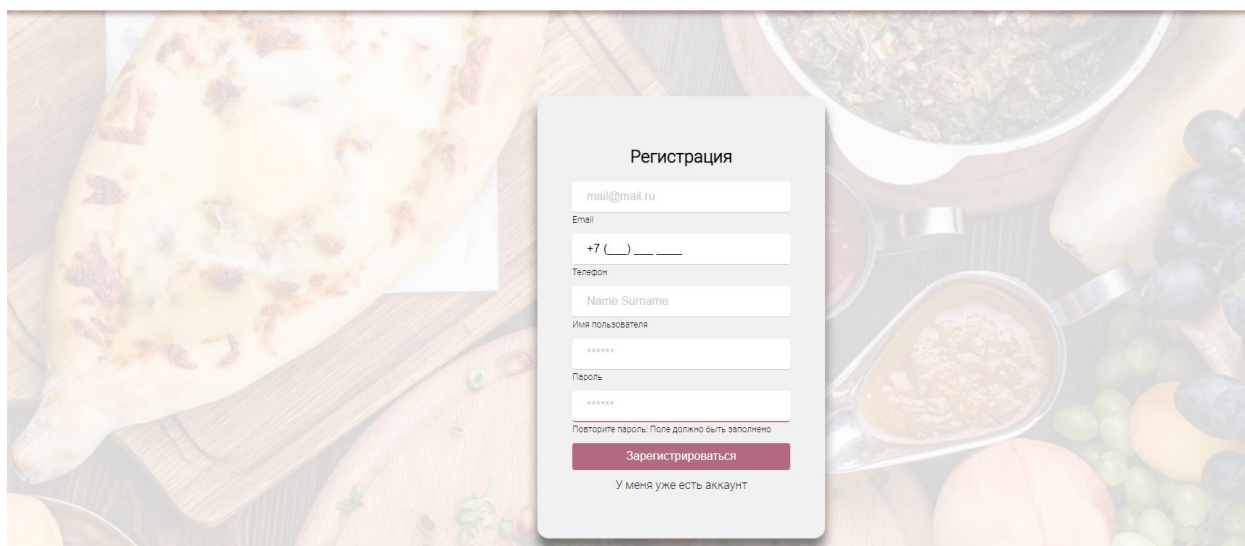
Вход в систему

Email

Пароль

[Я тут впервые](#)

Рисунок 7 – Форма входа



Регистрация

Email

Телефон

Имя пользователя

Пароль

Повторите пароль: Поле должно быть заполнено

[У меня уже есть аккаунт](#)

Рисунок 8 – Форма регистрации пользователя

Регистрация ресторана

Электронная почта:

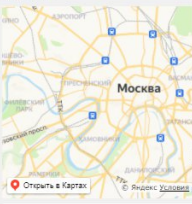
Телефон:

Название ресторана:

Адрес:

Пароль:

Повторите пароль:

Введите адрес: 

1000 м

Введите радиус покрытия (в метрах):

[Зарегистрировать свой ресторан](#)

[Уже зарегистрирован](#)

[У меня нет ресторана, я просто хочу кушать](#)

Рисунок 9 – Форма регистрации ресторана

Страница ресторана содержит название ресторана, его рейтинг, секции меню, само меню, корзину, стоимость доставки.

Cinnabon

Время доставки: 122 мин | Доставка: 89 P | ★ 4 | Отзывы

Выпечка | Капкейки и пирожные | Панини

Выпечка

Мико-ролл 190 P [В корзину](#)

Пшеничная смесь, маргарин, вода, дрожжи, корица, коричневый сахар, сыр сливочный, сахарная пудра, экстракт ванили, экстракт лимона, топпинг карамельный.

Пеканбон 305 P [В корзину](#)

Пшеничная смесь, маргарин, вода, дрожжи, корица, коричневый сахар, карамельный топпинг, орех пекан, сыр сливочный, сахарная пудра, экстракт ванили, экстракт лимона.

Синнабон 260 P [В корзину](#)

Пшеничная смесь, маргарин, вода, дрожжи, корица, коричневый сахар, сыр сливочный, сахарная пудра, экстракт ванили, экстракт лимона.

Шокобон 280 P [В корзину](#)

Корзина

Пеканбон 305 рублей 1

Доставка 89 P

Итого 394 P

[Сравнить](#) [Оформить](#)

Рисунок 10 – Страница ресторана

При нажатии на «Отзывы» можно почитать отзывы.

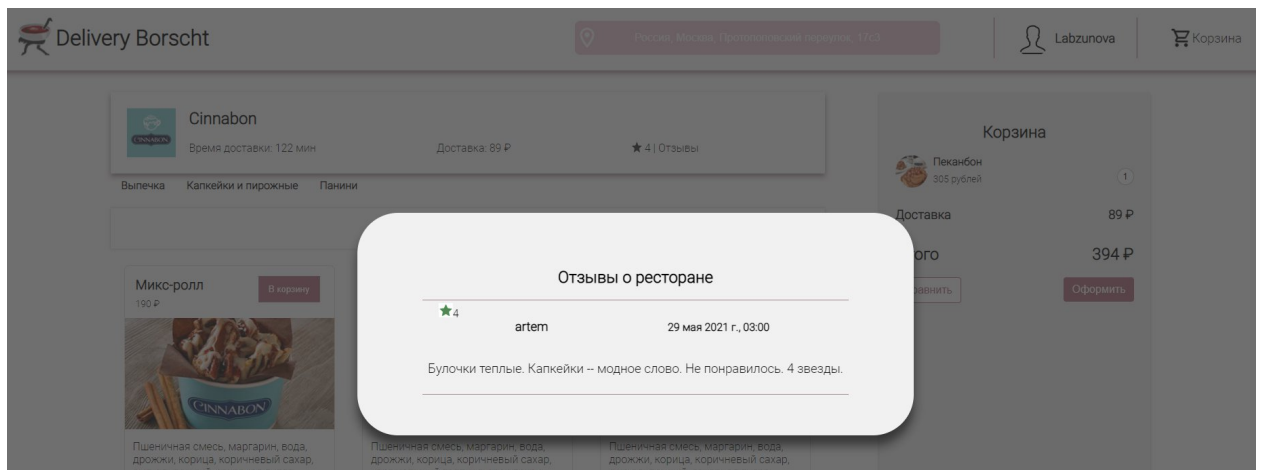


Рисунок 11 – Отзывы о ресторане

При нажатии «оформить» в корзине пользователь попадает на страницу оформления заказа.

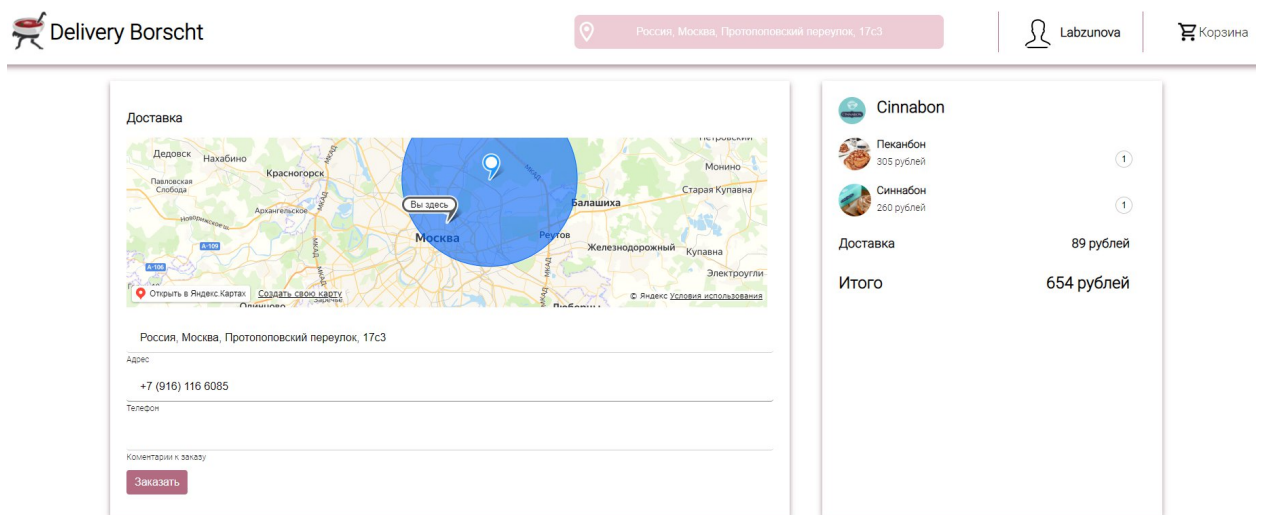


Рисунок 12 – Оформление заказа

После оформления пользователь попадает на страницу со своими заказами, где может отслеживать их статус.

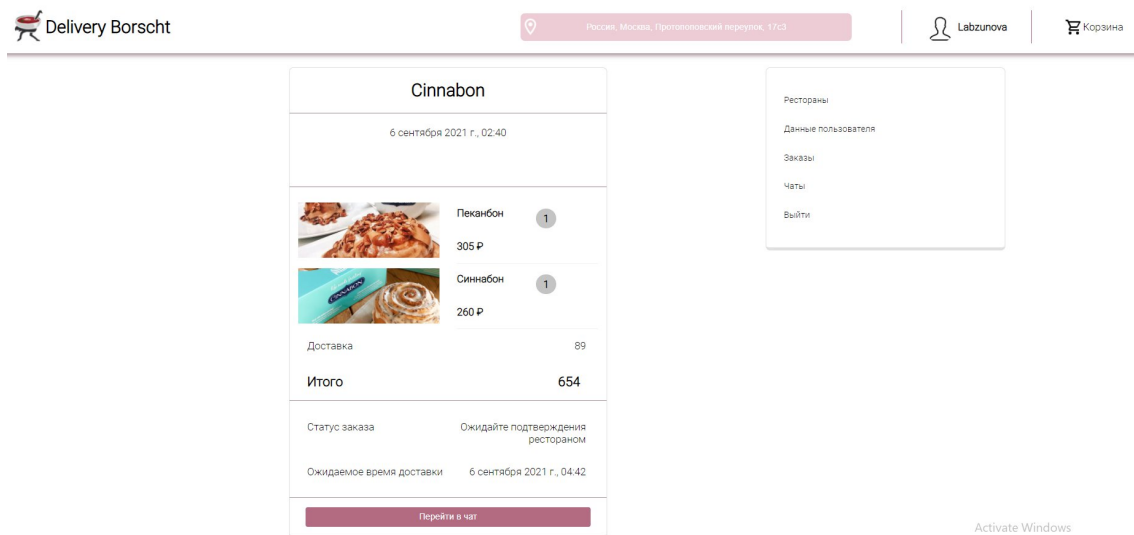


Рисунок 13 – Страница с заказами пользователя

При нажатии на «Данные пользователя» в правом меню можно посмотреть и изменить свои данные.



Рисунок 14 – Страница с данными пользователя

Со стороны ресторана можно добавлять и редактировать секции меню и блюда.

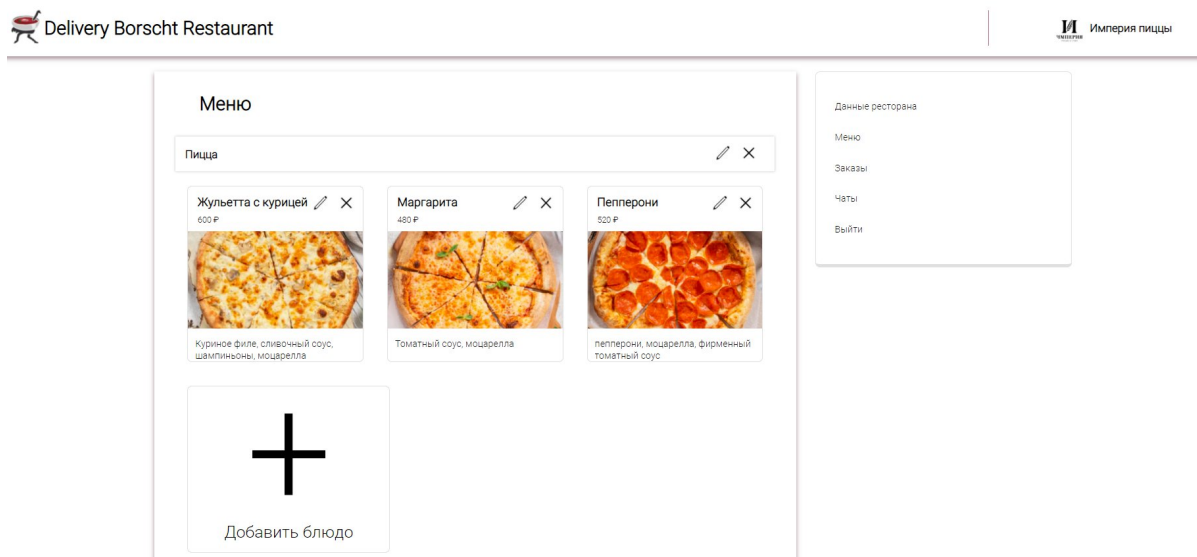


Рисунок 15 – Меню со стороны ресторана

2.2. Разработка структурной схемы программного продукта

С целью разбить программу на микросервисы была составлена структурная схема. В ходе анализа процессов в приложении было выявлено четыре подсистемы.

Главный сервис. В него приходят изначально все запросы. Он обрабатывает запросы на отдачу списка ресторанов, заказов, профиля пользователя, изменение профиля пользователя и ресторана. Остальные запросы он распределяет по микросервисам, описанным ниже;

Микросервис авторизации. В нем происходит обработка запросов авторизации: входа и регистрации.

Микросервис корзины. В нем обрабатываются запросы, связанные с корзиной: добавление в корзину, удаление из корзины, запрос на отдачу корзины.

Микросервис чатов. В нем обрабатываются запросы, связанные с чатами: получение всех чатов, получение сообщений конкретного чата, отправка сообщения.

На основе выделенных микросервисов была составлена структурная схема программного продукта.



Рисунок 16 – Структурная схема программного продукта

3. Выбор стратегии тестирования и разработка тестов

3.1. Тестирование структурным контролем

Первым методом тестирования было выбрано тестирование структурным контролем. Данное тестирование позволяет определить типовые ошибки, часто встречающиеся в коде. Список вопросов для структурного контроля пополняется с годами и опытом программистов. Результаты тестирования приведены в таблице 3.

Таблица 3 – Структурный контроль

Вопрос	Результат тестирования	Вывод
Все ли переменные инициализированы?	Да, иначе IDE показал бы ошибку.	Все переменные проинициализированы.
Присутствуют ли переменные со сходными именами?	Нет, иначе IDE показал бы ошибку.	Переменных с одинаковыми именами нет
Не выходят ли индексы за границы массивов?	Все обращения к массивам происходят в рамках длины массива.	Индексы не выходят за границы массивов.
Корректно ли выполнены вычисления с переменными различных типов (в том числе с использованием	Да, иначе IDE показал бы ошибку.	Вычисления с переменными различных типов выполнены корректно.

целочисленной арифметики)?		
Будут ли корректно завершены циклы?	Да, так как возможные ошибки обработаны, и выход из цикла будет либо по окончании счетчика, либо в ходе ошибки.	Циклы будут завершены корректно.
Будет ли завершена программа?	Каждый запрос пройдет полностью и отдаст ответ. Либо полностью пройдет весь алгоритм, либо обнаружит ошибку и сообщит о ней в ответе.	Программа завершена будет.
Существуют ли циклы, которые не будут выполняться из-за нарушения условия входа? Корректно ли продолжатся вычисления?	Таких циклов не существует. Если цикл не зайдет в свое тело – значит, так и нужно. В случае ошибок они обрабатываются и отдаются в ответе.	Все циклы выполняются и завершаются. Ошибки обработаны.
Существуют ли поисковые циклы? Корректно ли отрабатываются ситуации «элемент найден» и «элемент не найден»?	Все ситуации обработаны. В случае «элемент найден» и «элемент не найден» программа отдаст адекватный ответ.	Ситуации обрабатываются корректно.
Не изменяет ли подпрограмма аргументов, которые не должны изменяться?	Нет. Переменные, которые не должны меняться, отмечены как константы.	Программа не меняет аргументов, которые не должны меняться.
Не происходит ли нарушения области действия глобальных и локальных переменных с	Переменных с одинаковыми названиями нет.	Нарушения области действия глобальных и локальных переменных с одинаковыми именами не

одинаковыми именами?		происходит.
----------------------	--	-------------

Вывод из структурного контроля: программа работает корректно и типовых ошибок не обнаружено.

3.2. Unit-тесты

Unit-тесты — модульные тесты, применяемые в различных слоях приложения. Цель модульного тестирования — изолировать отдельные части программы и показать, что по отдельности эти части работоспособны.

В ходе работы были написаны Unit-тесты для большей части функционала. Окончательное покрытие кода тестами – 60%. Тестировалось как корректное исполнение программы, так и ситуации, в которых должны возникать ошибки. Тестировались все уровни архитектуры – handlers, usecase, repository.

Для имитации работы нижних слоев архитектуры использовались моки. Моки(mock) - это такие классы-заглушки (и соответственно объекты), которые позволяют избавиться от внешних зависимостей при модульном (unit) тестировании. Для создания моков использовался фреймворк GoMock. Для имитации работы базы данных в тестах самого нижнего слоя архитектуры – repository использовался фреймворк go-sqlmock. Все запросы в базу данных «ходили» не в реальную базу, а их обрабатывали написанные ранее заглушки на sqlmock.

Перед развертыванием веб-приложения обязательно запускаются тесты. И деплой приложения происходит только с условием того, что все тесты прошли. Добиться этого позволила технология Continuous Integration(CI). Когда код приложения оказывается в главной ветке репозитория, технологии Github запускают тесты и начинают собирать и развертывать приложение.

3.3. Usability-тест

Для проверки удобства и понятности интерфейса был проведен usability-тест. Юзабилити-тестирование — это метод оценки интерфейса со стороны удобства и эффективности его использования. Чтобы получить ее, нужно привлечь представителей целевой аудитории программного продукта. Был приглашен независимый пользователь, и для начала ему было задано несколько вопросов, а затем было предложено пройти на сайте два задания. Сценарий теста:

1. Представиться, рассказать, что разрабатываем

2. Сколько вам лет?
3. Как часто вы заказываете еду?
4. С помощью каких сервисов вы это делаете?
5. В каких ситуациях вы заказываете еду?
6. Опишите процесс заказа еды
7. Как часто возникают сложности при заказе еды? Какие это сложности?
8. Вы обычно заказываете, конкретно зная, что хотите? Или принимаете решения, просматривая сайт?
9. Сколько времени занимает принятие решения о том, в каком ресторане заказать?
10. Сравниваете ли вы еду из разных ресторанов? Как вы это делаете?
11. Задание: Если вы сейчас хотите что-то поесть - закажите еду в соответствии со своими пожеланиями.
12. Задание: Вспомните один из своих заказов еды и попробуйте повторить его.
13. Задание: Вы хотите заказать азиатской еды на сумму до 1500 рублей, из ресторана рейтингом 4+

Тестирование проводилось с целью определить удобство продукта и выявить места, в которых интерфейс или функционал можно изменить. В ходе тестирования было выявлено, что в целом, веб-сервис удобный и понятный, но работа с корзиной неудобна для пользователей ввиду того, что блюдо нельзя удалить из корзины в окне самой корзины. Это было взято на заметку к исправлению.

Заключение

В результате выполнения курсовой работы был получен опыт в разработке на языке Golang и JavaScript, а также опыт проектирования баз данных и архитектуры веб-сервисов в целом. Как итог – разработан веб-сервис для заказа еды «Delivery Borscht», позволяющее заказывать еду с доставкой на дом, а также зарегистрировать на сервисе свой ресторан и принимать заказы с данного веб-сервиса.

В ходе разработки была использована система контроля версий Git и платформа для хранения репозитория Github, на которой хранится исходный код программы. Для тестирования и развертывания была использована методология CI/CD.

Также в процессе разработки были созданы различные диаграммы и схемы с целью получения более продуманного продукта и понимания всех выкладок его работы.

По окончании разработки приложение было протестировано для того, чтобы удостовериться, что оно работает корректно и правильно выполняет предполагаемые функции.

Итоговый сервис соответствует заявленному ТЗ и выполняет все требования.

Список используемых источников

1. Официальная документация по языку Golang [Электронный ресурс]. – URL: <https://golang.org/doc/>
2. Официальная документация СУБД PostgreSQL, версия 13.4 [Электронный ресурс]. – URL: <https://www.postgresql.org/docs/13/index.html>
3. Конспекты и видеозаписи лекций по курсу «Разработка веб-сервисов на Golang» образовательной программы «Технопарк»
4. Иванова Г.С., Ничушкина Т.Н. Тестирование программного обеспечения: Методические указания по выполнению лабораторной работы по дисциплине "Технология разработки программных систем" [Текст]. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2019. – 13 с.
5. Методы обработки данных и оценки программ : учебное пособие / Г. С. Иванова, Т. Н. Ничушкина, Е. К. Пугачев. — Москва : Издательство МГТУ им. Н. Э. Баумана, 2020. — 72, [2] с. : ил.
6. Роберт Мартин. Чистая архитектура. Искусство разработки программного обеспечения – Издательство Питер СПб, 2018. – 352 с.
7. Backend-репозиторий проекта 2021_1_Borscht_With_Cabbage – URL: https://github.com/go-park-mail-ru/2021_1_Borscht_With_Cabbage
8. Frontend-репозиторий проекта 2021_1_Borscht_With_Cabbage – URL: https://github.com/frontend-park-mail-ru/2021_1_Borscht_With_Cabbage

Приложение А. Техническое задание.

Исходный код веб-приложения для заказа еды «Delivery Borscht» находится на Github.com:

Backend-часть приложения: https://github.com/go-park-mail-ru/2021_1_Borscht_With_Cabbage

Frontend-часть приложения: https://github.com/frontend-park-mail-ru/2021_1_Borscht_With_Cabbage