Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Инженерно-экономический факультет

Кафедра экономической информатики

Дисциплина: Современные технологии обработки экономической информации

Пояснительная записка

к курсовому проекту

на тему

**РАЗРАБОТКА АВТОМАТИЗИВАРОННОЙ СИСТЕМЫ ПООДЕРЖКИ ПРОЦЕССОВ СЛУЖБЫ ДОСТАВКИ ЕДЫ КАФЕ И РЕСТОРАНОВ**

|  |  |
| --- | --- |
| Студент  Группы 572303 | Богданова К.М. |
| Руководитель  старший преподаватель | Унучек Т.М. |

Минск 2018

СОДЕРЖАНИЕ

[Введение 4](#_Toc533531194)

[1 Процедура доставки еды кафе и ресторанов 6](#_Toc533531195)

[1.1 Сущность и перспективы создания единой системы доставки еды 6](#_Toc533531196)

[1.2 Основные компоненты агрегатора доставки еды с точки зрения потребителя 7](#_Toc533531197)

[1.2 Анализ рынка агрегаторов доставки еды 9](#_Toc533531198)

[2 Функциональное моделирование на основе стандарта IDEF0 14](#_Toc533531199)

[3 Постановка задачи 21](#_Toc533531200)

[3.1 Назначение и цели создания системы 21](#_Toc533531201)

[3.3 Требования, предъявляемые к системе 21](#_Toc533531202)

[3.3.1 Требования к надежности. 21](#_Toc533531203)

[3.3.2 Требования к графическому интерфейсу. 22](#_Toc533531204)

[4 Проектирование и разработка базы данных системы автоматизации 23](#_Toc533531205)

[5 Обоснование выбора компонентов и технологий для реализации курсового проекта 28](#_Toc533531206)

[5.1 Описание ключевых технологий и обоснование их выбора 28](#_Toc533531207)

[5.2 Описание диаграммы последовательности 30](#_Toc533531208)

[5.3 Описание диаграммы компонентов 31](#_Toc533531209)

[5.4 Описание диаграммы развертывания 32](#_Toc533531210)

[6 Спецификация вариантов использования системы 33](#_Toc533531211)

[7 Модели представления системы и их описание 34](#_Toc533531212)

[8 Руководство пользователя 36](#_Toc533531213)

[8.1 Развертывание приложения 36](#_Toc533531214)

[8.2 Работа с сайтом 37](#_Toc533531215)

[9 Результаты тестирования разработанной системы 43](#_Toc533531216)

[Заключение 45](#_Toc533531217)

[Список использования источников 46](#_Toc533531218)

[Приложение А (Листинг кода) (Обязательное) 47](#_Toc533531219)

[Приложение Б (Листинг генерации базы данных) (Обязательное) 54](#_Toc533531220)

# **ВВЕДЕНИЕ**

Вследствие развития информационных технологий появилась возможность упрощать и ускорять многие процессы в повседневной деятельности людей. Так, например, ранее для покупки билетов необходимо было ездить в билетные кассы, теперь же достаточно заказать билет через интернет, или же для расчета рентабельности предприятия раньше экономисты пересчитывали множество документированных данных вручную, в наши дни это все делается через специальные программы, в которых уже заранее хранятся все данные по предприятию. Люди не хотят тратить свое время и силы на вещи, которые с помощью технологий можно сделать за минуты.

Кроме того, в повседневной жизни у людей не всегда есть время заниматься каким-то домашними заботами или отвлекаться на долгий серфинг интернета в поисках желаемых услуг.

Перейдем больше к проблеме заказа и выбора еды на доставке. В наше время доставка существует практически у каждого ресторана, однако это не всегда удобно, так как для выбора еды необходимо заходить на каждый из них. Например, команда разработчиков выводит проект в релиз, у них нет времени искать поблизости заведения с едой, а тем более каждый хочет что-то свое: кто-то предпочтёт армянскую кухню, кто-то мексиканскую. При большом количестве людей уйдет очень много времени на формирование заказов. Кроме того, каждая доставка в таком случае привезет заказ в разное время. Это сразу создает идею: а нельзя ли создать некий сервис, который объединит в себе все доставки?

Однако не стоит рассматривать подобный сервис в качестве удобства только для клиентов, он также имеет множество преимуществ для самих кафе и ресторанов. Во-первых, подобные сервисы предоставляют свою команду доставки, таким образом, ресторану нет необходимости содержать отдельный штат сотрудников и покупать автомобили. Во-вторых, это реклама: человек не всегда знает, чего точно он хочет, когда заходит на сервис доставки, он начинает просматривать различные категории, просматривает меню ресторанов, в процессе этого он может открыть для себя множество новых заведений.

Таким образом, целью данного проекта является автоматизация работы системы доставки еды и ресторанов путем решения следующих задач:

– исследование принципов работы доставки;

– изучение существующих аналогов систем доставки;

– разработка программного средства, которое будет обеспечивать наиболее эффективную работу доставки;

– разработка uml-диаграмм и блок-схем программы;

– описание результатов тестирование программного средства;

В приложении предусмотрено роли клиента и администратора:

– администратор: определяет права пользователей, добавляет и редактирует новые категории, редактирует страницы меню заведений, редактирует зоны доставок;

– пользователь: просматривает доступные категории кухонь: азиатская, американская, армянская и т.д., просматривает меню заведений, добавляет заказы в корзину, способен осуществлять различные фильтрации заведений: по популярности, по районам, по ценовой категории и т.д., осуществляет заказ, оставляет отзывы и оценки заведениям.

# ПРОЦЕДУРА ДОСТАВКИ ЕДЫ КАФЕ И РЕСТОРАНОВ

## 1.1 Сущность и перспективы создания единой системы доставки еды

Доставка еды – это довольно новый, перспективный и быстроразвивающийся сегмент бизнеса. Также это отдельный сервис, который не всегда может быть успешным у определенных ресторанов, какой бы качественной ни была их кухня. Рестораторы нанимают сотрудников для сервиса доставки, пользуются различными маркетинговыми приемами, но все же не всегда выходят на необходимый им уровень бизнеса.

Существует наиглавнейший критерий хорошей доставки: время. Для доставки еды это имеет наибольшую важность. Во-первых, заказывая еду, клиент рассчитывает получить ее как можно скорее, а не через пару часов. Во-вторых, горячая пища и не продержится долгое время в хорошем состоянии. Кроме того, ошибкой многих заведений является добавление в меню доставки блюд, которые не выдерживают долгого ожидания в процессе доставки. Также не стоит забывать, что после того, как сотрудник доставил еду, ему еще необходимо вернуться в ресторан и забрать новый заказ, поэтому штат сотрудников на доставке должен быть соответствующего размера, чтобы не заставлять клиентов ждать.

Дополнительно, необходимо учитывать среднее количество заказов по времени суток и по праздникам. Праздники – это наиболее прибыльные дни для ресторанов, однако же если доставка не может справиться с потоком клиентов, то никакой выгоды от таких дней нет, а разочарованные клиенты могут оставить отрицательные отзывы, что повлияет на рейтинг ресторана.

Из этого стоит сделать вывод, что не каждый ресторан способен быть

успешным в области доставки еды. В таком случае ему могут помочь сервисы-агрегаторы. Подобные сервисы существуют не только в области доставки еды, но и во многих других: флористика, авиабилеты (Anywayanyday.com, Skyscanner.ru и др.), бронирование отелей и апартаментов (Booking.com) и многие другие.

Агрегатор в сфере доставке – это еще и отдельная маркетинговая система. Например, это размещение рекламных блоков на главной странице сайта, это ведение системы отзывов и оценивания ресторанов, также может осуществляться сбор данных по просмотрам и поведению клиентов. Кроме того, может вестись статистика по отмене заказов и причине этих отмен, это также может помочь выявить ресторану его слабые места. Также для выявления проблемных мест агрегаторы в сфера доставки еды предоставляют систему оценивания самих блюд, это помогает как клиенту в выборе еды, так и ресторану в выборе наиболее популярных блюд.

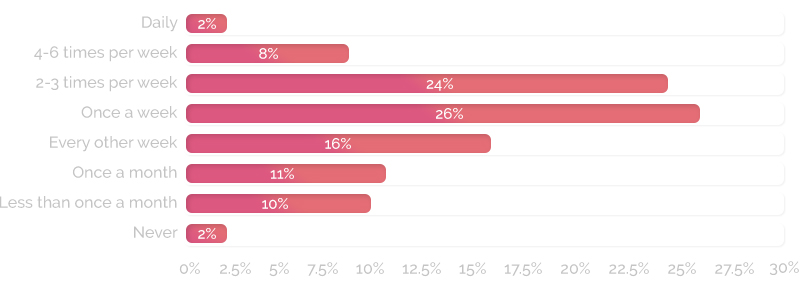
Приведем наглядную статистику по объему ранка доставки еды на 2016 год для Казахстана, Украины и России (рисунок 1.1).



Рисунок 1.1 – Статистика объема рынка доставки блюд

Из этой статистики очевиден потенциал данной области бизнеса.

Также приведем статистику частоты заказов от одного клиента (рисунок 1.2). Данное исследование было проведено в США. Видно, что очень многие заказывают хотя бы раз в неделю. Если это сопоставить с количеством клиентов, то получается довольно внушительное количество заказов. Всего 2% людей никогда не пользовались доставкой.



Никогда

Реже раза в месяц

Раз в месяц

Каждую вторую неделю

Один раз в неделю

2-3 раза в неделю

4-5 раз в неделю

Ежедневно

Рисунок 1.2 - Статистика частоты заказов от одного клиента

## 1.2 Основные компоненты агрегатора доставки еды с точки зрения потребителя

Итак, какие основные функции должны быть в приложении по заказу еды:

1 После размещения заказа, клиент получает информации о заказе со временем отправки еды из ресторана и временем доставки.

2 Подтверждение о получении заказа рестораном.

3 Добавление адреса доставки.

4 Автоматические уведомления о том, что заказ выехал из ресторана.

Отметим также основные особенности и характеристики систем доставки:

1 Каталог/меню. Это, вероятно, главная особенность, так как трудно представить приложение такого типа без меню предметов, доступных для доставки. Бакалея и собственные приложения для доставки продуктов обычно имеют каталоги с категориями и подкатегориями, чтобы облегчить поиск нужного продукта или еды. В приложениях, работающих с различными ресторанами, обычно есть сложные меню с различными фильтрами, чтобы найти блюда определенной кухни или ресторана.

2 Геолокация. Как правило, многие приложения для доставки продуктов доступны только в определенных городах или регионах, поэтому они просят пользователей ввести свой почтовый индекс или нажать кнопку «Найти меня». Кроме того, в некоторых приложениях используется геолокация, чтобы обеспечить разницу в меню для разных областей и рассчитать время транспортировки и плату.

3 Онлайн-платежи. Сегодняшний рынок предлагает различные платежные системы, специально предназначенные для продажи товаров, таких как PayPal, BrainTree, Stripe и многие другие. Каждый из них имеет свои собственные настройки, поэтому, прежде чем выбирать один, лучше связаться с их представителями. Это принесет удобство как клиенту, так и сотруднику, занимающемуся бухгалтерией заведения.

4 Трекинг заказа. Хотя это и не обязательная функция, многим пользователям нравится проверять статус их доставки. Эта опция может быть представлена ​​с помощью отдельного меню в приложении или службе обмена сообщениями, когда вы получаете SMS-оповещения с уведомлениями, сообщающими вам, когда еда готова, покидает ресторан и достигает места назначения. Расширенный способ проверки текущего состояния вашего заказа - отслеживание GPS в реальном времени, когда вы можете проверить, как ваша еда приходит к вам.

5 Запланированная доставка. Планирование доставки особенно удобно для доставки продуктов, когда люди могут заказать вперед и установить точное время, когда их товары должны быть доставлены. Преимущества заключаются в следующем:

- Во-первых, вам не нужно ждать своей еды, она придет именно тогда, когда вам это нужно.

- Во-вторых, это особенно удобно во время праздников, когда службы доставки перегружены количеством заказов.

6 Программы лояльности. Реализация некоторых бонусных программ также может помочь ресторану сохранить пользователей. Примеры программ лояльности:

- Купоны, предлагающие определенный процент скидки для людей, заказывающих на определенную сумму или имеющих большое количество заказов.

- Ваучеры, предоставленные в особых случаях, содержащие коды для использования в приложении, которые предоставляют скидки на заказы или некоторые позиции товаров в подарок.

- Очки, которые начисляются за на покупку определенных товаров или размещение каждого нового заказа. Наличие некоторого количества очков позволяет пользователям обменивать свои очки на рекламные предметы (это могут быть футболки, очки, тарелки с логотипом и т. д.) Или же с помощью них можно получить скидку.

- Специальные предложения, которые побуждают клиента делать заказы в определенные дни. Примерами являются счастливые часы скидок или предложений, таких как «1 + 1 = 3» (купить два, получить три) и т. д.

## 1.2 Анализ рынка агрегаторов доставки еды

Проведем сравнение различных агрегаторов доставки еды: menu.by, foodout.lv.

1. menu.by

Главная страница представлена на рисунке 1.3.

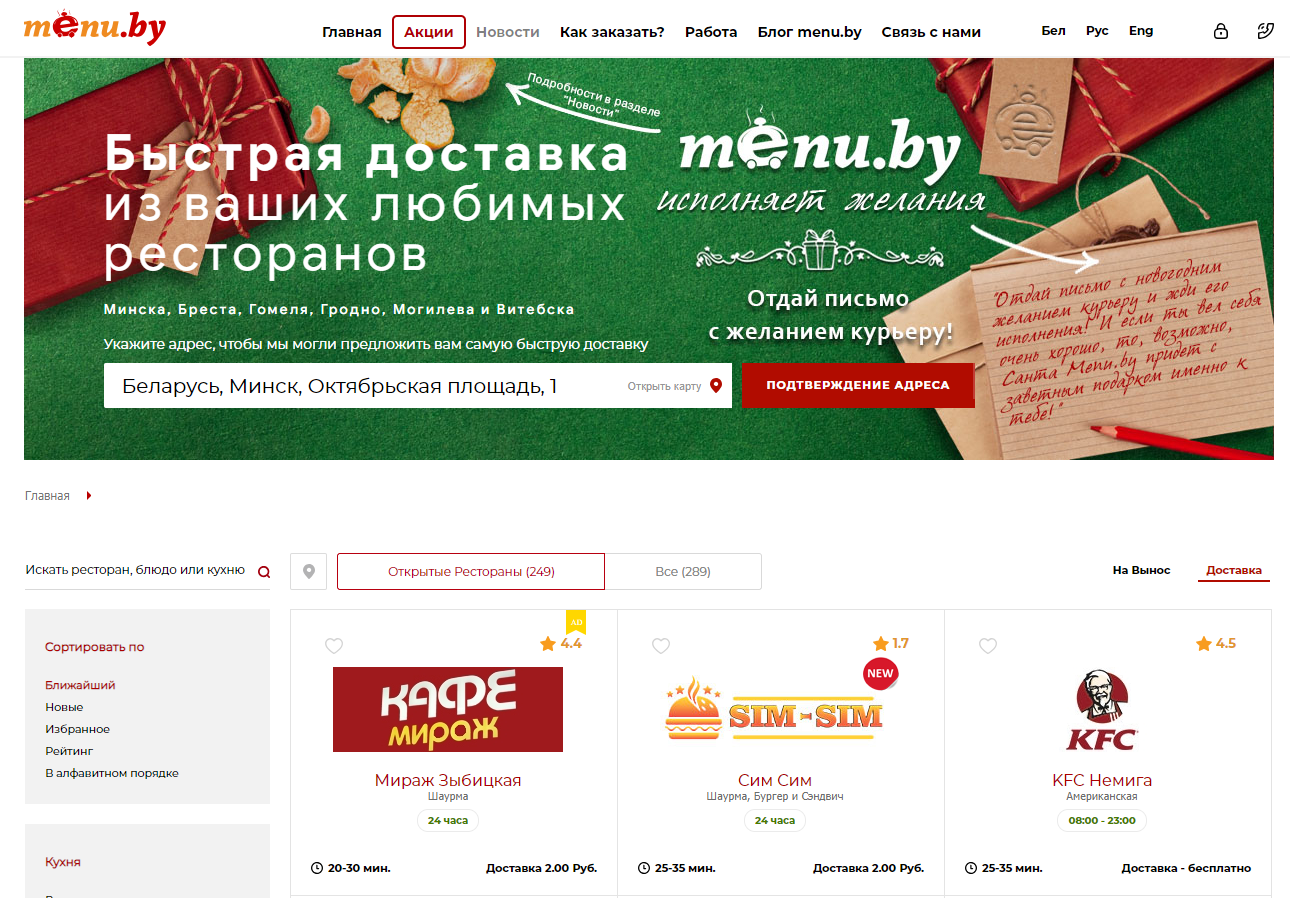


Рисунок 1.3. – Главная страница menu.by

Сайт menu.by является наиболее популярным на рынке Беларуси. В данном сайте отражен наиболее полный функционал, который возможно ожидать от агрегатора доставок. В его основной функционал входит:

1. Фильтрации ресторанов по типу кухни.
2. Сортировка по локации, новизне и рейтингу.
3. Добавление ресторана в избранное.
4. Добавление блюд в корзину.
5. Оформление заказа.
6. Выставление рейтинга и написание отзывов на ресторан.

Интерфейс данного сайта не нагроможден, является user-friendly, дизайн непосредственно страниц также является приятным и располагает к себе.

При выборе ресторана отображается информация о самом ресторане, отзывы, а также его меню (рисунок 1.4).

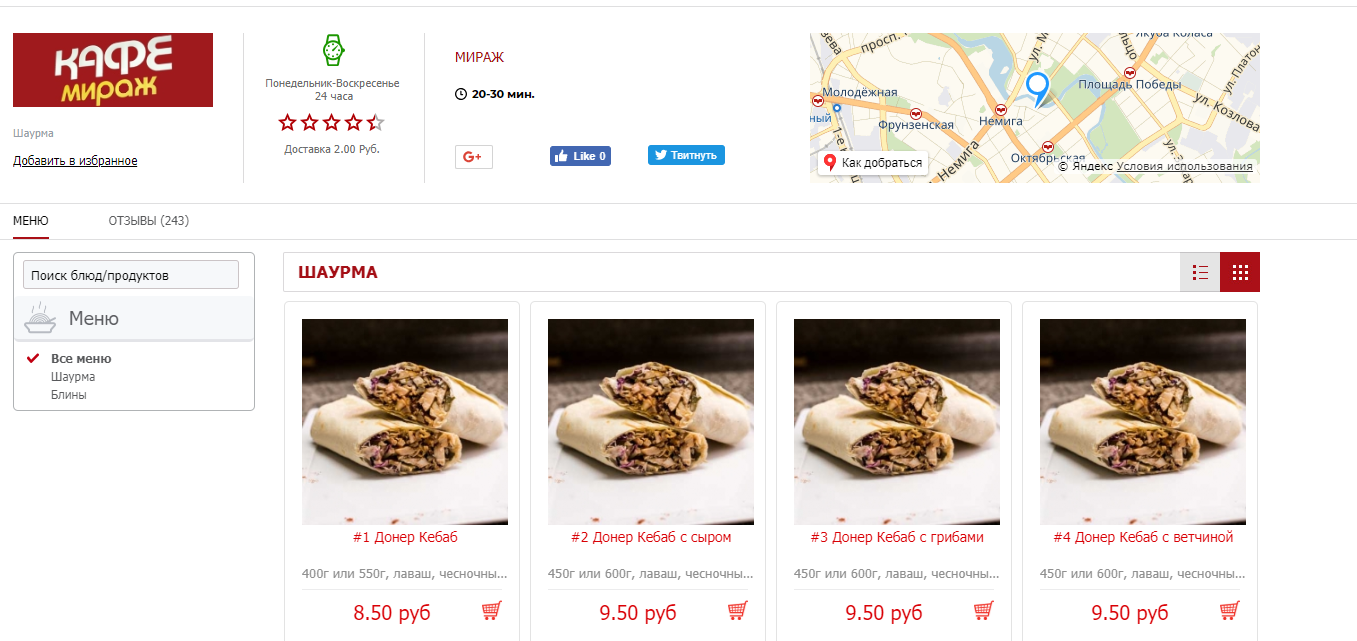


Рисунок 1.4 – Меню ресторана

Из этой страницы производится заказ, корзина отображается справа (рисунок 1.5).

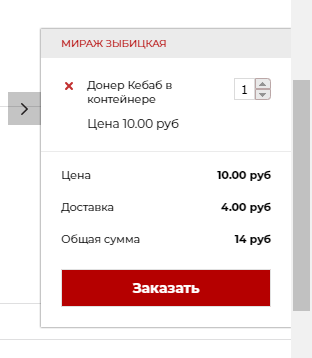


Рисунок 1.5 – Корзина заказа

На рисунке 1.6 представлена форма оформления заказа:

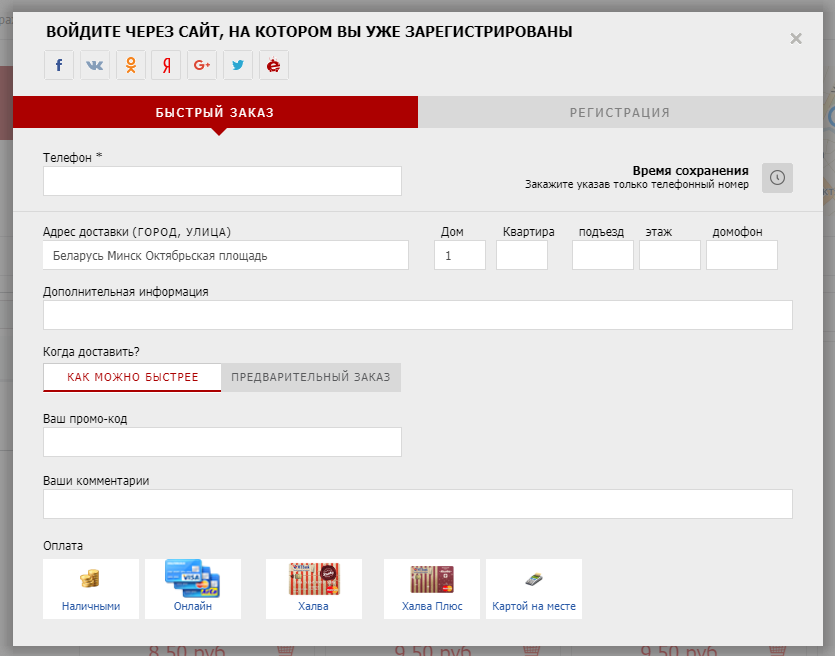


Рисунок 1.6 – Форма оформления заказа

Из недостатков данного сайта можно отметить отсутствие изображений в корзине или же описание выбранного блюда. При заказе, например, большого количество разных видов суши, имеется трудности в их идентификации.

Второй агрегатор доставок еды кафе и ресторанов – это foodout.lv. Это наиболее популярный сайт доставки Латвии.

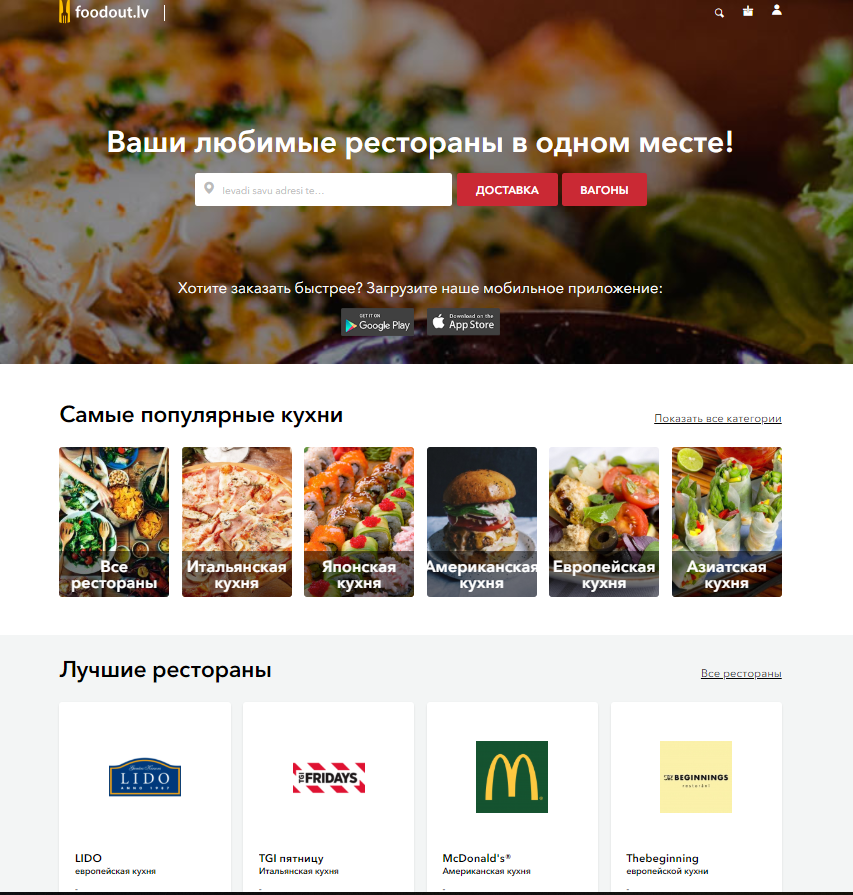


Рисунок 1.7 – Главная страница сайта foodout.lv

Аналогично с menu.by, в начале заказа можно выбрать свое местонахождение (рисунок 1.8).

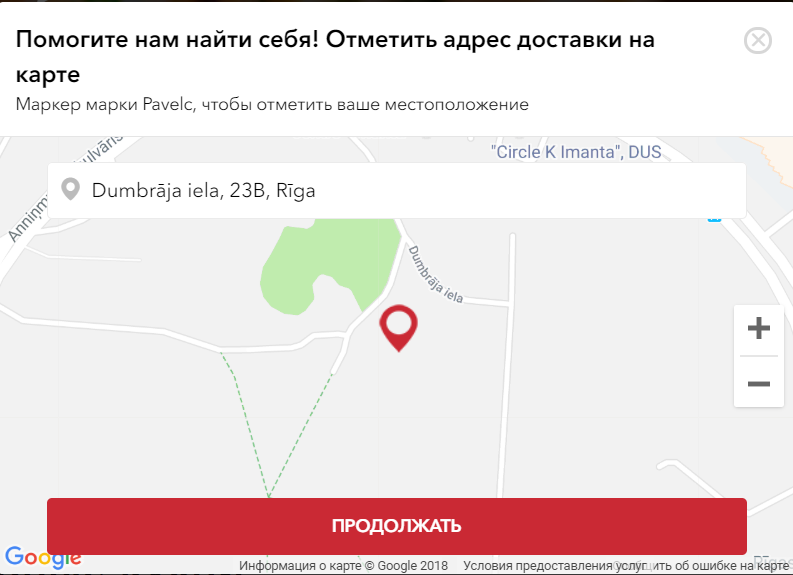


Рисунок 1.8 – Выбор локации доставки

На данном сайте значительно меньше фильтраций и отсутствуют сортировки, однако на карточке ресторана отображен средний ценник меню (рисунок 1.9).

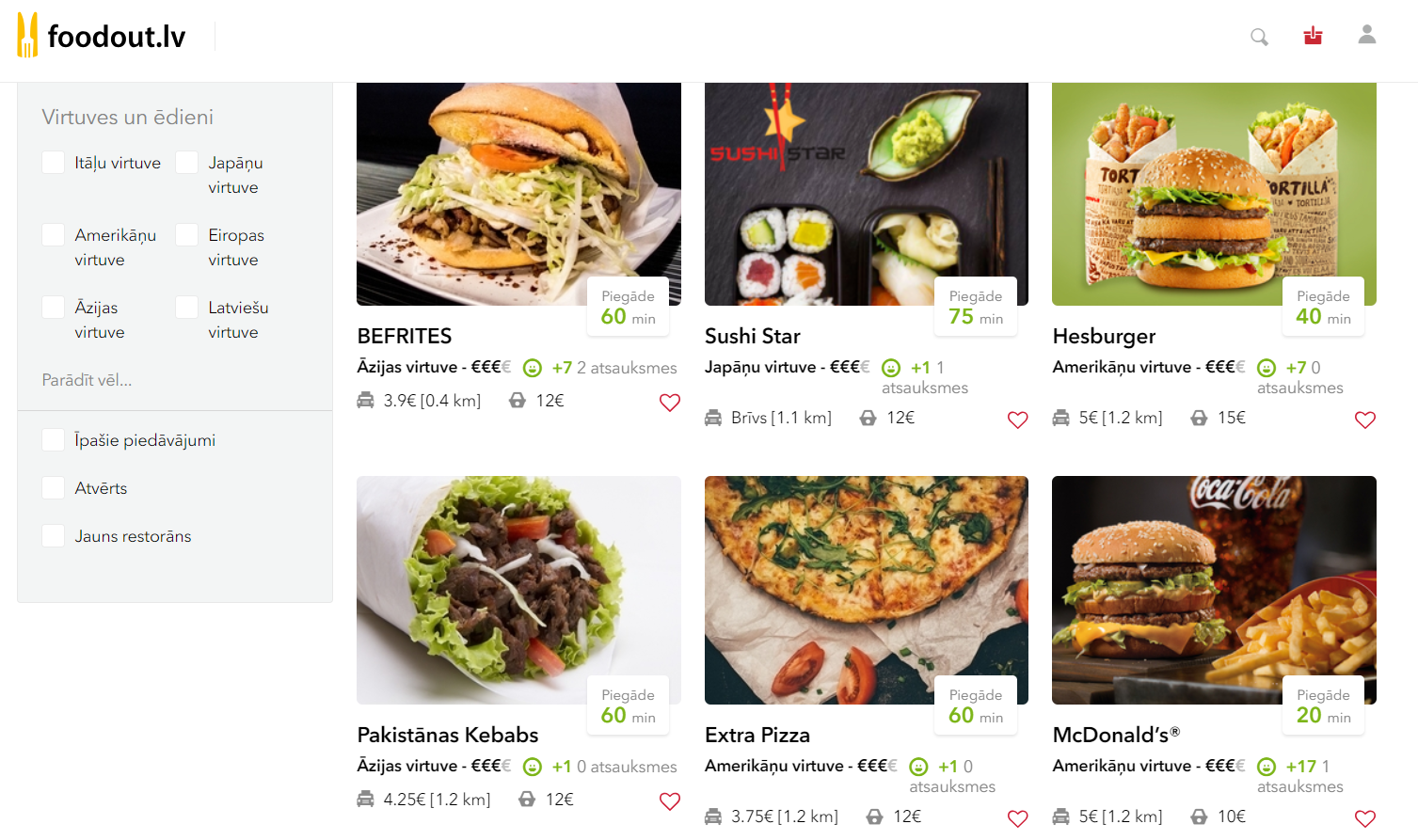


Рисунок 1.9 – Карточки ресторанов

# 2 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ НА ОСНОВЕ СТАНДАРТА IDEF0

Рассмотрим диаграмму организации доставки в ресторане. На контекстной диаграмме (рисунок 2.1) расположен главный процесс, а именно организация доставки в ресторане, этот процесс имеет на входе необходимость поставки оборудования, желающие заказывать еду, необходимость создания нового штата сотрудников и необходимость создания сайта. Результатом этого процесса является установленное оборудование, принятые заказы от клиентов, новые работники и сайт ресторана. Контролируются все процессы законодательством РБ, требованиями к кандидатам на должность сотрудника call-центра и требованиям владельца. Все процессы выполняет персонал ресторана, поставщиками и программистами (нанятыми или из штата, если имеются).



Рисунок 2.1 – Контекстная диаграмма «Организовать доставку еды ресторана»

При декомпозиции контекстной диаграммы (рисунок 2.2) получаем четыре процесса: закупка необходимого оборудования (место оператора call-центра, компьютер для отслеживания доставки, телефон, переносные терминалы, контейнеры для перевозки еды), создания штата работников доставки (чем крупнее ресторан, тем больше работников необходимо) и создание сайта ресторана, проработка скрипта продажи. Механизмом работы каждого из них является персонал ресторана, а также для процесса закупки оборудования добавляется поставщик, а для создания сайта - программист. В первом блоке происходит изменение состояния наличия необходимого оборудования, получая на выходе установленное и готовое к использованию оборудование. Этот процесс контролируется законодательством и требованиями владельца ресторана. Создание штата работников выполняется администратором ресторана с утверждением кандидата владельцем ресторана. На выходе процесса имеются новые работники. Далее необходимо создать сайт, через который клиенты будут осуществлять заказ доставки еды. Главный механизм этого процесса – это программисты. Скрипт продажи – это принцип поведения оператора call-центра. Какие заранее установленные фразы и вопросы необходимо сказать при звонке клиенту. Итогом данного процесса являются принятые заказы от клиентов (которые в последующем приносят прибыль).



Рисунок 2.2 – Декомпозиция процесса «Организовать доставку еды»

Рассмотрим декомпозицию процесса закупки необходимого оборудования (рисунок 2.3). Данный процесс состоит из следующих подпроцессов:

1. Заключить договор поставки оборудования.
2. Составить документ закупки.
3. Получить оборудование.
4. Установить оборудование.
5. Протестировать оборудование.
6. Ввести оборудование в эксплуатацию.

Изменение состояний входных параметров, а также управляющие элементы можно увидеть на рисунке 2.3.



Рисунок 2.3 – Декомпозиция процесса «Закупить необходимое оборудование»

Процесс создания штата работников (рисунок 2.4) доставки включает в себя:

1. Распаространение информации о вакансии.
2. Собеседование в службе персонала.
3. Проверка кандидата.
4. Принятие решения о приеме на работу.

Ключевым механизмом данных процессов является администратор, а также владелец на последнем этапе. Контролируется этот процесс законодательством, требованиям к кандидатам в выставленном объявлении о вакансии и требованиями владельца ресторана.



Рисунок 2.4 – Декомпозиция процесса «Создать штат работников доставки»

Далее следует процесс создания сайта (рисунок 2.5). Его этапы:

1. Определить основную задачу сайта.
2. Проанализировать и исследовать задачи сайта.
3. Создать дизайн макета веб-сайта.
4. Написать программную реализацию сайта.
5. Протестировать сайт.

Выходным параметром после окончания процесса является готовый сайт ресторана. Также в данном процессе, помимо необходимости в создании сайта, входным потоком является установленное оборудование.

Контролируется процесс требованиями владельца, а выполняется программистами.



Рисунок 2.5 – Декомпозиция процесса «Создать сайт ресторана»

Процесс «Проанализировать и исследовать задачи сайта» также можно рассмотреть, как совокупность более мелких процессов:

1. Проанализировать структуру сайта.
2. Проанализировать работу динамических модулей.
3. Проработать пожелания по дизайну.
4. Определение технических требований к сайту.

Данная декомпозиция представлена на рисунке 2.6.

Контролируют процесс требования владельца. Механизмом исполнения являются программисты.



Рисунок 2.6 – Декомпозиция процесса «Проанализировать и исследовать задачи сайта»

И последнее – это декомпозиция процесса написания программной реализации сайта. Последовательность написания программы следующая:

1. Сверстать страницы сайта.
2. Спрограммировать сайт.
3. Наполнить сайт контентом.
4. Расположить сайт в сети интернет.

Механизм: программисты. Контролируется техническим заданием.

Детальное изменение входящего потока можно увидеть на рисунке 2.7.



Рисунок 2.7 – Декомпозиция процесса «Написать программную реализацию сайта»

# 3 ПОСТАНОВКА ЗАДАЧИ

## 3.1 Назначение и цели создания системы

При разработке автоматизированной системы преследуются следующие цели:

− разработка комплексной системы, ориентированной на предоставление ресторанам и кафе готовой к интеграции системы доставки еды;

− предоставление клиентам возможности заказа блюд из нескольких ресторанов в рамках одной доставки;

− предоставление систематизированного хранение данных;

− предоставление механизма поиска нужного ресторана по цене и категории;

− предоставление возможности выставлению рейтинга ресторану, а также написанию отзывов на них;

− предоставление посетителю возможности управления своими заказами;

− обеспечение удобного и понятного интерфейса пользователя;

− обеспечение безопасности хранимых данных.

3.2 Общее описание системы

Система автоматизации включает в себя программное обеспечение, состоящее из ряда двух основных частей:

1 Веб-приложение, реализующее:

- возможности просмотра имеющихся в системе ресторанов

- их фильтрация

- просмотр меню

- написание отзывов на ресторан

- выставление рейтинга ресторану

- формирование заказа (с сохранением сессии).

2 База данных

Представляет собой реляционную базу, в которой хранится следующая информация:

- информация о всех ресторанах

- информация об отзывах и оценках ресторанов

- меню каждого ресторана

- информация о заказах

- временно хранящаяся информация о текущих корзинах

## 3.3 Требования, предъявляемые к системе

### 3.3.1 Требования к надежности.

Система должна оставаться работоспособной и обеспечивать восстановление функционирования при возникновении следующих ситуаций:

* при ошибках, связанных с программным обеспечением;
* при сбоях аппаратных средств;
* при сбоях в системе электроснабжения, приводящих к перегрузке операционной системы.

Так же система должна выдерживать большой поток клиентов, одновременно запрашивающих доступ к одному и тому же ресурсу.

### 3.3.2 Требования к графическому интерфейсу.

Интерфейс должен быть user-friendly, быть не нагроможденным и иметь проработанный дизайн. Все страницы должны иметь единый стиль оформления.

Интерфейс приложения должен быть:

1. Доходчивым. Пользователь не должен тратить большое количество времени на поиск способа выполнения желаемых действий, интерфейс должен быть интуитивным.
2. Отзывчивым. Система не должна тормозить или же долго загружаться. В случае, если избежать этого невозможно, необходимо уведомлять пользователя об этом в виде индикатора загрузки или другого отображения процесса загрузки.
3. Лаконичным и последовательным. Интерфейс должен быть адаптирован для контекста и не должен быть перегружен информацией.

# 4 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА БАЗЫ ДАННЫХ СИСТЕМЫ АВТОМАТИЗАЦИИ

В разрабатываемой системе используется реляционная база данных MySQL.

4.1 Логическое проектирование

Первым этапом проектирования базы данных является логическое проектирование. На данном этапе осуществляется преобразования семантических требований к данным в конкретные логические структуры данных, определяются связи между сущностями. Логическая модель разрабатываемой базы представлена на рисунке 3.1.



Рисунок 4.1 – Логическая модель базы данных

4.2 Физическое проектирование

Физическое проектирование служит для переноса логической модели базы в СУБД. Во время данного проектирования определяются особенности различных атрибутов, методы доступа, индексы и типы связей между таблицами.

Таблица “user” (пользователь) используется для хранения данных о всех зарегистрированных в системе пользователях. Она состоит из полей, приведенных в таблице 4.1.

Таблица 4.1 – Описание таблицы «User»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| id | int | Уникальный идентификатор |
| login | varchar | Логин пользователя |
| password | varchar | Пароль |
| created\_at | varchar | Время создания записи |

Таблица «Restaurants» (Рестораны) используется для хранения данных о всех имеющихся в системе ресторанах. Она состоит из полей, приведенных в таблице 4.2.

Таблица 4.2 – Описание таблицы «Restaurants»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| id | int | Уникальный идентификатор |
| category\_id | varchar | Логин пользователя |
| name | varchar | Пароль |
| image | varchar | Путь к логотипу ресторана |
| schedule | varchar | Расписание работы ресторана |
| average\_rating | int | Рейтинг ресторана. Формируется на основе отзывов как среднее арифметическое. |
| is\_active | varchar | Является ли ресторан действующим |

Таблица «Restaurant\_Feedbacks» (отзывы на ресторан) хранит информацию об отзывах на рестораны. Там находится отзыв, оценка и имя пользователя, оставившего его. Она состоит из полей, приведенных в таблице 4.3.

Таблица 4.3 – Описание таблицы «Restaurant\_Feedbacks»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| id | int | Уникальный идентификатор |
| restaurant\_id | varchar | Внешний ключ на ресторан, на который пишется отзыв |
| user\_id | varchar | Внешний ключ на пользователя |
| rate | varchar | Рейтинг ресторана |
| comment | varchar | Комментарий к ресторану |
| created\_at | int | Время создания записи |

Таблица «Locations» (местоположения) хранит адреса. Она состоит из полей, приведенных в таблице 4.4.

Таблица 4.4 – Описание таблицы «Locations»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| id | int | Уникальный идентификатор |
| street | varchar | Улица |
| house | varchar | Дом |
| room | varchar | Квартира |
| is\_active | varchar | Является ли ресторан действующим |

Таблица «Categories» (категории) хранит категории ресторанов, например, суши или армянская кухня. Она состоит из полей, приведенных в таблице 4.5.

Таблица 4.5 – Описание таблицы «Categories»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| id | int | Уникальный идентификатор |
| title | varchar | Улица |
| created\_at | int | Время создания записи |

Таблица «Cart\_Items» (Элемент корзины) хранит блюда, которые находятся в данный момент в корзине с привязкой к сессии. После окончания сессии или же оформления заказа таблица очищается. Она состоит из полей, приведенных в таблице 3.6.

Таблица 4.6 – Описание таблицы «Cart\_Items»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| id | int | Уникальный идентификатор |
| session\_id | varchar | Идентификатор сессии, к которой привязана корзина |
| product\_id | varchar | Внешний ключ продукта, добавленного в корзину |
| quantity | int | Количество блюд одного типа |

Таблица «Restaurant\_Dishes» (Блюда ресторана) хранит блюда из меню соответствующего ресторана. Она состоит из полей, приведенных в таблице 4.7.

Таблица 4.7 – Описание таблицы «Restaurant\_Dishes»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| id | int | Уникальный идентификатор |
| restaurant\_id | int | Внешний ключ на ресторан, на который пишется отзыв |
| name | varchar | Название блюда |
| description | varchar | Описание блюда |
| price | int | Цена |
| image | varchar | Путь к изображению блюда |
| is\_active | int | Существует ли данное блюдо на данный момент в меню |

Таблица «Orders\_Form» (Заказ) информацию о заказе. Она состоит из полей, приведенных в таблице 4.8.

Таблица 4.8 – Описание таблицы «Orders\_Form»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| id | int | Уникальный идентификатор |
| session\_id | int | Идентификатор сессии, к которой привязан заказ |
| name | varchar | Имя клиента |
| phone | varchar | Телефон клиента |
| email | varchar | Почта клиента |
| location\_id | int | Внешний ключ адреса заказа |
| total\_price | int | Итоговая стоимость заказа |
| created\_at | timestamp | Время создания записи |

Таблица «Orders\_Items» (Элементы заказа) информацию о содержимом заказа. Она состоит из полей, приведенных в таблице 3.9.

Таблица 4.9 – Описание таблицы «Orders\_Items»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| id | int | Уникальный идентификатор |
| order\_id | int | Внешний ключ заказа |
| product\_id | int | Внешний ключ заказанного блюда |
| quantity | int | Количество заказанных блюд одного наименования |
| created\_at | timestamp | Время создания записи |

4.4 Описание связей между таблицами

Существует четыре типа связи между таблицами:

– один - к - одному: каждой записи из одной таблицы соответствует одна запись в другой таблице;

– один - ко - многим: каждой записи из одной таблицы соответствует несколько записей другой таблицы;

– многие - к - одной: нескольким записям из одной таблице соответствует одна запись другой таблицы;

– многие - ко - многим: нескольким записям из одной таблицы соответствует несколько записей из другой таблице

В данной базе данных существует только связи один-ко-многим (она же многие-к-одной). Данные связи можно увидеть на рисунке 4.1 на логической модели данных.

4.5 Индексация

Индексы существуют для улучшения производительности чтения данных. Их необходимо создавать на те таблицы и по тем полям, по которым наиболее часто производится выборка. Если же чаще происходит операция вставки данных, то вводить индексы нежелательно, так как при добавлении индекса происходит перестроение бинарного дерева, в котором упорядоченно хранятся ссылки на записи в таблицах, что значительно ухудшит производительность.

В данной базе используется индекс на таблице ресторанов по полю категории ресторана, так как данный фильтр является наиболее часто используемым клиентами. Также индексы автоматически устанавливаются по ключам таблиц.

4.6 Нормализация базы данных

Данная база приведена к 3 нормальной форме, так как соблюдены все следующие условия:

1. Первая нормальная форма

Переменная отношения находится в первой нормальной форме (1НФ) тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов.

1. Вторая нормальная форма

Переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме и каждый не ключевой атрибут неприводимо (функционально полно) зависит от её потенциального ключа.

1. Третья нормальная форма (3NF)

Переменная отношения находится в третьей нормальной форме тогда и только тогда, когда она находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости не ключевых атрибутов от ключевых.

# 5 ОБОСНОВАНИЕ ВЫБОРА КОМПОНЕНТОВ И ТЕХНОЛОГИЙ ДЛЯ РЕАЛИЗАЦИИ КУРСОВОГО ПРОЕКТА

## Описание ключевых технологий и обоснование их выбора

Для написания программной реализации сайта доставки еды был выбран язык JavaScript.

На 2018 год JavaScript является наиболее популярным языком программирования. На данный момент JS покрывает не только фронт-энд, но и бэк-энд за счет большого разнообразия фреймворков.

График самых популярных языков согласно сайту Stack Overflow представлен на рисунке 5.1.

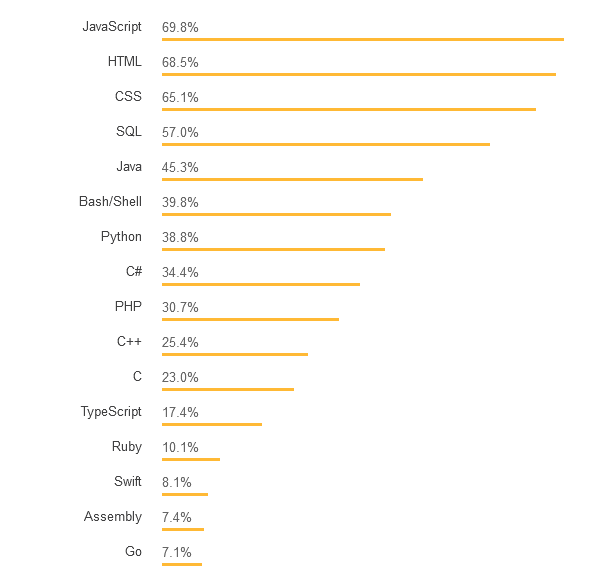


Рисунок 5.1 – Статистика популярности языков программирования

В процессе первичного проектирования системы были выбраны следующие технологии и фреймворки:

1. Node.js – серверная платформа, работающая на движке V8. Она является средой выполнения JavaScript. C Node проще масштабироваться. При одновременном подключении к серверу тысяч пользователей Node работает асинхронно, то есть ставит приоритеты и распределяет ресурсы грамотнее. На Node можно писать полноценные приложения.
2. npm  (аббр. node package manager)  – менеджер пакетов, входящий в состав Node.js. С помощью npm можно устанавливать пакеты локально или глобально. В локальном режиме пакеты устанавливаются в каталог node modules родительского каталога. Владельцем каталога является текущий пользователь. Он позволяет разработчику найти уже готовые модули, а не заниматься длительной разработкой того, что уже существует.
3. Express – быстрый, гибкий и минималистичный веб-фрейморк для приложений Node.js, который предоставляет следующие возможности:

* Понятная маршрутизация;
* помощники перенаправления (redirect helpers);
* динамические помощники представлений;
* опции представлений уровня приложения;
* взаимодействие с контентом;
* монтирование приложений;
* ориентированность на высокую производительность;
* рендеринг шаблонов и поддержка фрагментных шаблонов (view partials);
* конфигурации, быстро переключаемые под разные задачи (development, production, и т.д.);
* хранимые в сессии всплывающие сообщения (flash messages);
* скрипт express для быстрой генерации каркаса приложения;
* высокое покрытие тестами.

1. Vue.js – прогрессивный фреймворк для создания пользовательских интерфейсов. Vue создан для пригодным для постепенного внедрения. Также он подходит и для создания сложных одностраничных приложений. Один из основных элементов Vue.js – это директивы. Ниже представлен список некоторых из них:

V-bind — динамически связывается с одним или несколькими атрибутами.

V-cloak — прячет “усатые” выражения, пока не подтянулись данные

v-if — условие для рендера элемента

V-else — обозначает “else блок” для v-if

V-for — циклично проходит массив объектов

V-model — связывает состояние с input элементом

V-on — связывает слушателя события с элементом

V-once — рендерит элемент только вначале и больше не следит за ним

V-pre — не компилирует элемент и его дочерние элементы

V-show — переключает видимость элемента, изменяя свойство CSS display

V-text — обновляет textContent элемента

Простой пример использования Vue.js:

app.js:

var demo = new Vue({

el: '#demo',

data: {

message: 'Hello, Singree!'

}

})

app.html:

<div id="demo">

<p>{{message ? message : "no content"}}</p>

<input v-model="message">

</div>

В результате будет отображена фраза, указанная в поле message в app.js.

1. Axious – это JavaScript библиотека, предназначенная для выполнения HTTP-запросов в Node.js, либо XMLHttpRequests в браузере. Одна из его особенностей – это автоматическое преобразование JSON-данных. Она использует промисы по умолчанию, а так же работает как на сервере(что делает его подходящим для получения данных при рендеринге на сервере), так и на клиенте. Axios очень легко начать использовать с vue.

## 5.2 Описание диаграммы последовательности

Рассмотрим диаграмму последовательности в рамках варианта использования «Оформление заказа» (Рисунок 4.2)



Рисунок 5.2 – Диаграмма последовательности процесса оформления заказа

Главным актером (действующим лицом) данного процесса является желающий оформить доставку еды клиент. Он заполняет форму заказа в окне браузера и подтверждает введенную информацию. После этого клиент формирует post-запрос и отправляет его на сервер, используя библиотеку axious. Сервер получает запрос используя фреймфорк express и отправляет данные для записи в базу данных. После этого клиенту возвращается результат выполненной операции в виде сообщения в окне формы заказа.

## 5.3 Описание диаграммы компонентов

Диаграмма компонентов показывает разбиение программной системы на структурные компоненты и связи между компонентами. Цель разработки диаграммы компонентов заключается в следующем:

* Спецификация общей структуры исходного кода системы;
* Спецификация исполнимого варианта системы.



Рисунок 5.3 – Диаграмма компонентов системы

Система состоит из трех основных компонент:

1. Клиентская часть.
2. Серверная часть.
3. База данных.

Клиент в свою очередь состоит из нескольких пакетов:

1. public – в нем содержатся все статические элементы системы. К ним относятся стили css, ресурсы в виде изображений, а также статические js файлы.
2. views – в данном пакете находятся файлы с версткой страниц в виде шаблонизаторов twig.

К серверу относятся пакеты:

1. routes – скрипт, вызывает необходимые методы API, а также перенаправляет на нужные страницы.
2. Services – API системы.
3. node\_modules – пакеты с готовыми модулями, загруженные с помощью npm.

## 5.4 Описание диаграммы развертывания

Диаграмма развертывания моделирует физическое развертывание артефактов на узлах. Например, чтобы описать веб-сайт диаграмма развертывания должна показывать, какие аппаратные компоненты («узлы») существуют (например, веб-сервер, сервер базы данных, сервер приложения), какие программные компоненты («артефакты») работают на каждом узле (например, веб-приложение, база данных), и как различные части этого комплекса соединяются друг с другом (например, JDBC, REST, RMI). [5]



Рисунок 5.4 – Диаграмма развертывания системы

На диаграмме отражено, что системы состоит из трех основных элементов база данных, сервер и клиент, а также показаны их компоненты.

# СПЕЦИФИКАЦИЯ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ СИСТЕМЫ

Спецификация вариантов использования системы представлена на рисунке 6.1.



Рисунок 6.1 – Диаграмма «Use case» системы

В системе имеется 2 роли: Администратор и Клиент. Администратор отвечает за управление содержимым сайта: список ресторана, их меню, категории ресторанов в системе.

К вариантам использования клиента относятся:

1. Просмотр новостей, связанных с системой доставки или партнерами.
2. Просматривать рестораны.
3. Фильтровать рестораны по категории и ценнику.
4. Искать рестораны по названию.
5. Оценивать заведения.
6. Оставлять отзывы на заведения.
7. Работать с корзиной: добавлять блюда, удалять, менять количество порций.
8. Изучать инструкцию по оформлению доставки.
9. Получать консультацию от работников доставки.

# 7 МОДЕЛИ ПРЕДСТАВЛЕНИЯ СИСТЕМЫ И ИХ ОПИСАНИЕ

Рассмотрим изменение состояния заказа в процессе его формирования и оформления. Он представлен на рисунке 7.1.

На данной диаграмме рассмотрен основной процесс сайта доставки еды: сбор и оформление заказа со стороны клиента. При оформлении заказа клиентом сначала происходит добавление желаемых блюд в корзину. Если блюда выбраны, то клиент проверяет свой заказ и открывает корзину. Там, в свою очередь, можно изменить количество выбранных блюд или же удалить полностью из корзины. Если весь заказ верен, то появляется форма заказа для заполнения. В ней находятся имя клиента, его телефон и адрес, а также потовый ящик (опциональное поле). Если все необходимые поля заполнены и вся информация верна, то заказ отправляется на сервер и записывается в базу данных для последующей обработки. Если нет, то клиент либо исправляет неверные данные, либо заканчивает сессию.



Рисунок 7.1 – Диаграмма состояний системы

На блок схеме отражен алгоритм действий пользователя на сайте. После фхода в систему перед ним открывается главная страница, на ней располагаются все действующие рестораны. Основной фильтр – это выборка по категории кухни. Например, это может быть кухня азии, пицца или же белорусская кухня. Дополнительно к категории, например, можно выбрать фильтр по цене. Далее клиенту необходимо выбрать ресторан, который ему наиболее нравится. Сдлеать он может по личным предпочтениям, или же по отзывам или рейтингу, оставленными другими пользователями. После завершения выбора блюд клиент открывает корзину, перепроверяет свой заказ и заполняет форму с личными данными: имя, телефон, адрес и почтовый ящик.

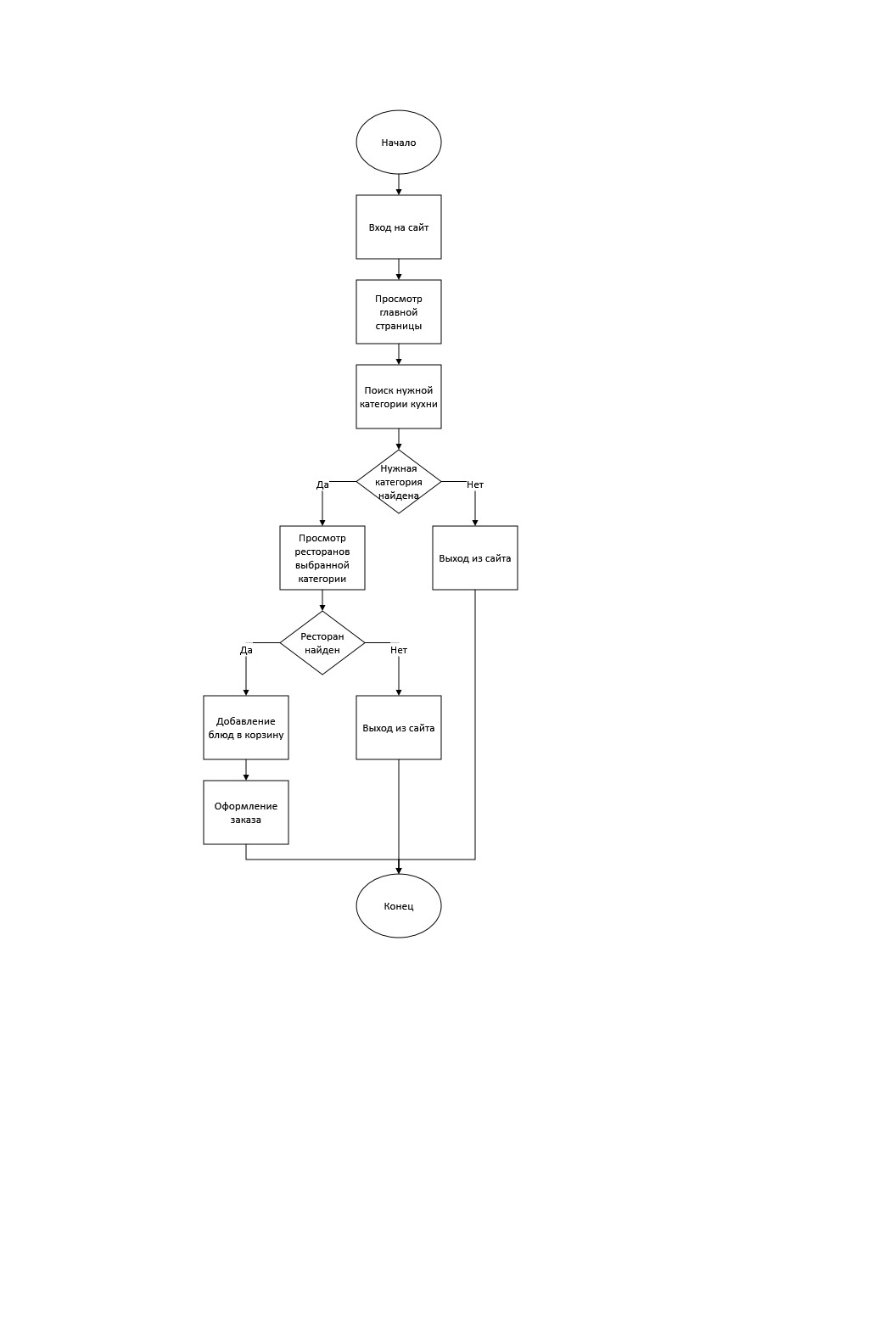


Рисунок 7.2 – Блок схема процесса заказа блюд на доставку

Таким образом, мы рассмотрели основной принцип работы с сайтом доставки еды кафе и ресторанов.

# 8 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## 8.1 Развертывание приложения

Для запуска сервера необходимо выполнить несколько шагов:

1. Импорт базы данных.

Для этого необходимо запустить в базе данных MySQL скрипт, который находится в корневом каталоге в файле «foodpanda-18-12-2018-16-30.sql».

1. В файле config.js, который также находится в корневом каталоге изменить соответствующие настройкам базы данных параметры: host, user, password, database.



Рисунок 8.1 – Содержимое файла config.js

1. Установить Node.js. Его установка доступна с официального сайта.
2. Запустить файл «launch server.bat».

Теперь сайт доступен по адресу localhost:3030 (рисунок 8.2).

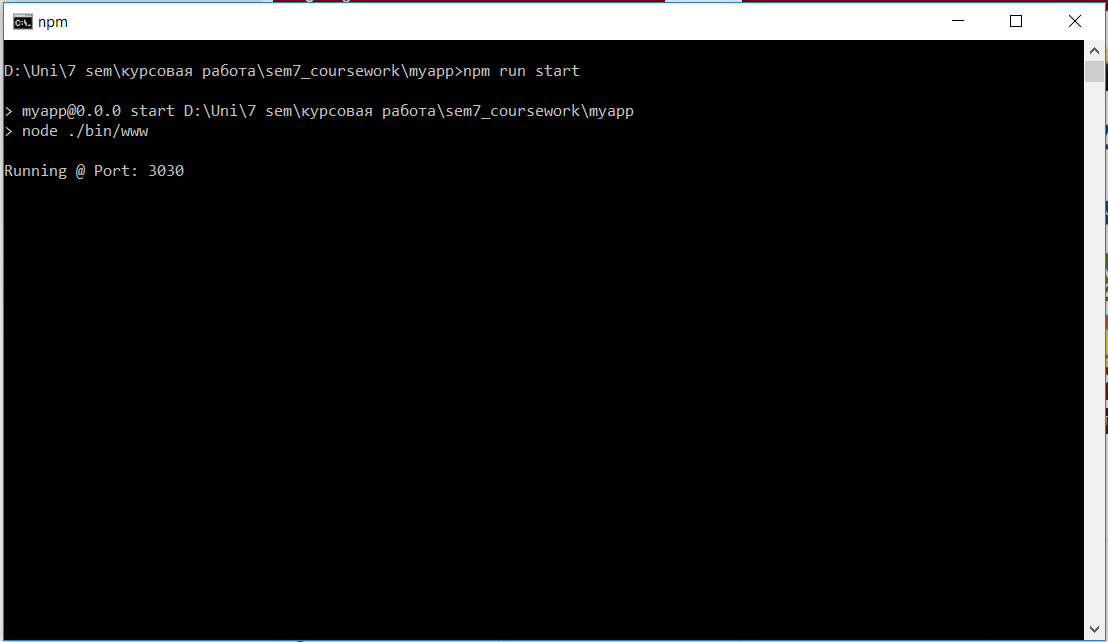


Рисунок 8.2 – Сервер запущен

## 8.2 Работа с сайтом

После запуска сервера и открытия сайта по выше указанному адресу пользователю открывается главная страница сайта (рисунок 8.3)

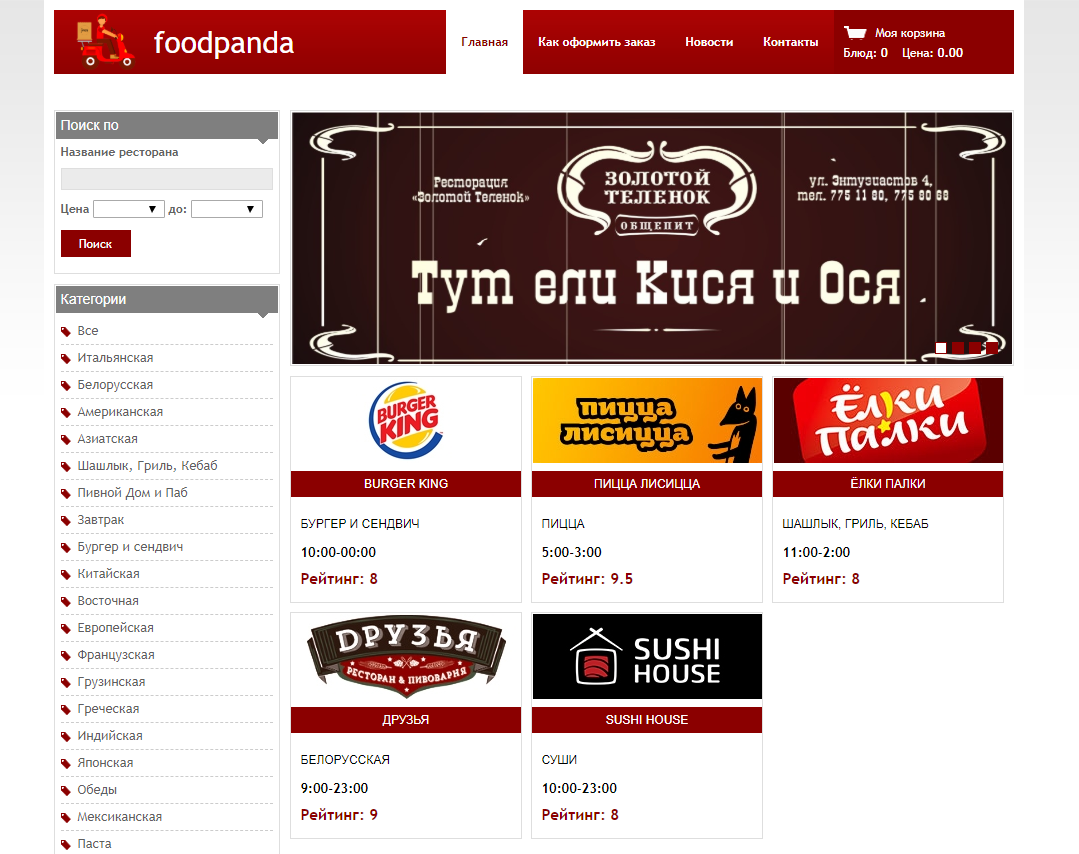


Рисунок 8.3 – Главная страница сайта

На сайте имеется 4 страницы:

* Главная;
* Как оформить заказ;
* Новости;
* Контакты.

Последние три страницы являются чисто информационными. Первая содержит подробную инструкцию об оформлении заказа, на второй располагаются новости портала, а также партнеров, на странице Контакты находится местоположение офиса и способ связаться с консультантами. Содержимое этих страниц представлено на рисунках 8.4 и 8.5.

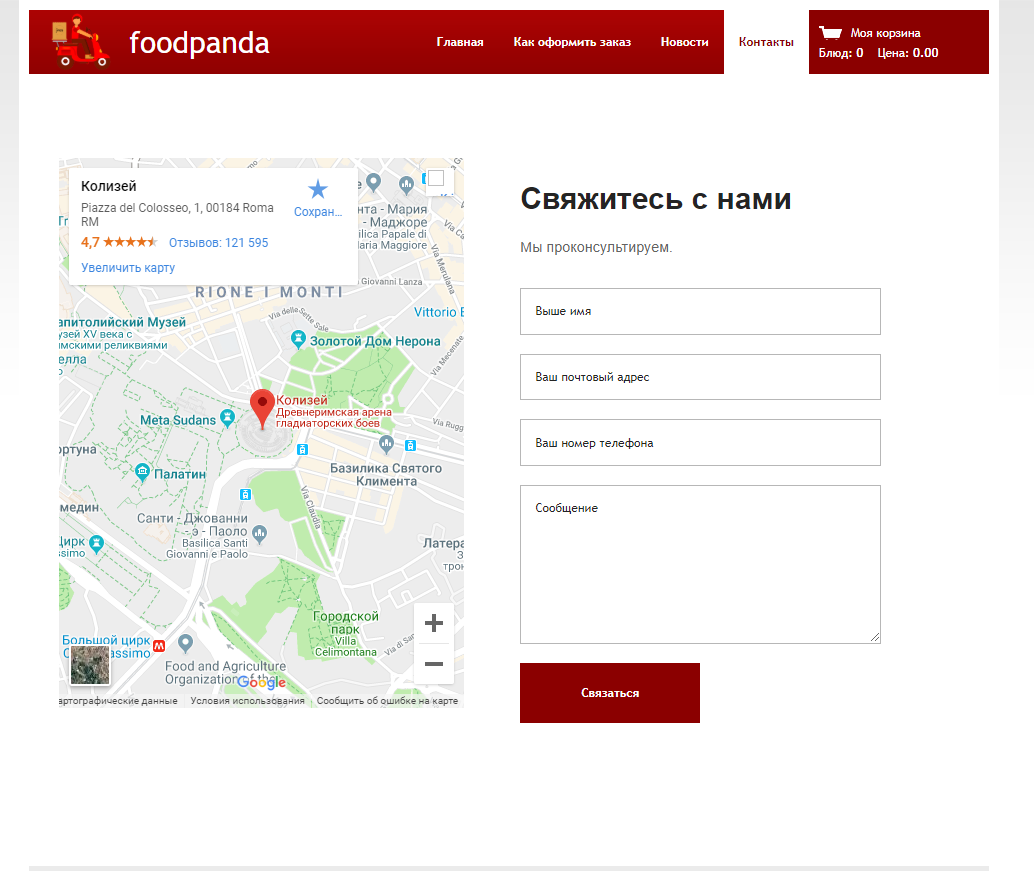


Рисунок 8.4 – Страница «Контакты»



Рисунок 8.5 – Страница «Как оформить заказ»

На главной странице доступна фильтрация ресторанов по категории и по цене. Также есть поиск по названию ресторана.

Для применения фильтра необходимо просто выбрать соответствующие значения. Пример показан на рисунке 8.6.

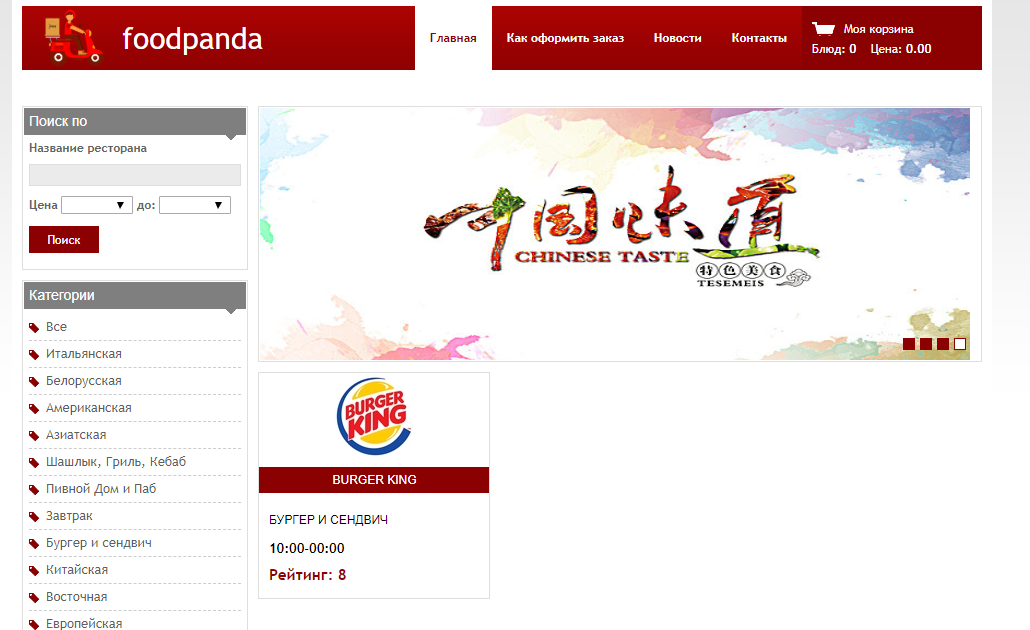


Рисунок 8.6 – Результат фильтрации по категории «Бургер и сэндвич»

Аналогично можно фильтровать по цене и названию (рисунок 7.7).

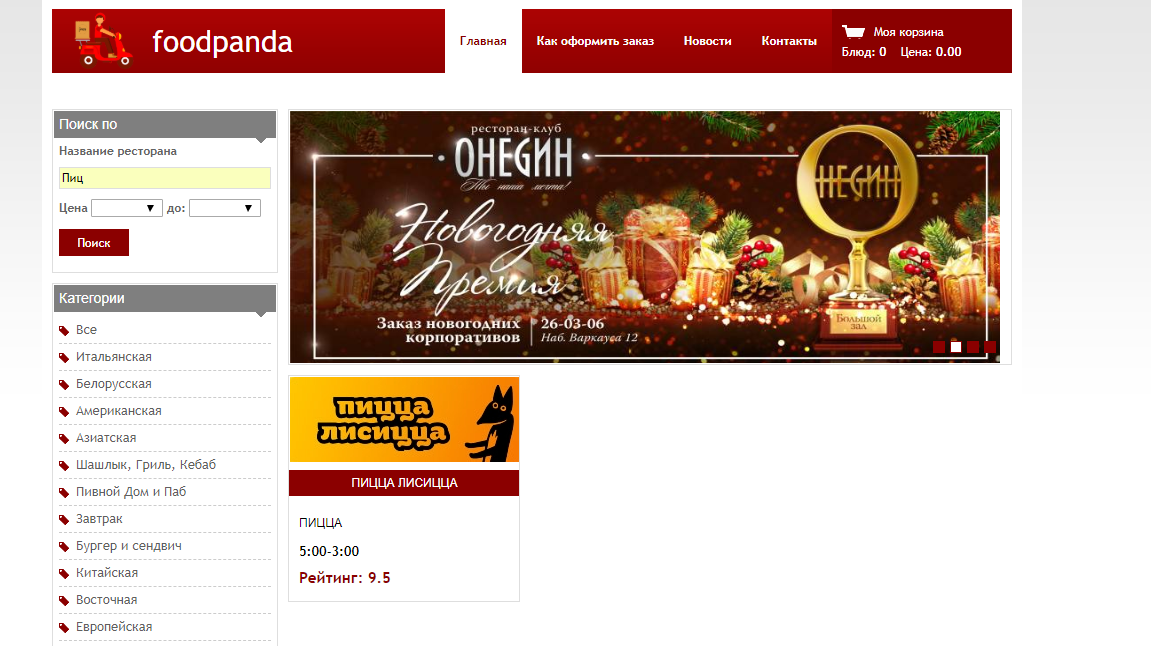


Рисунок 8.7 – Поиск по названию

При нажатии по логотипу ресторана отображается его меню, а также отзывы (рисунок 8.8).

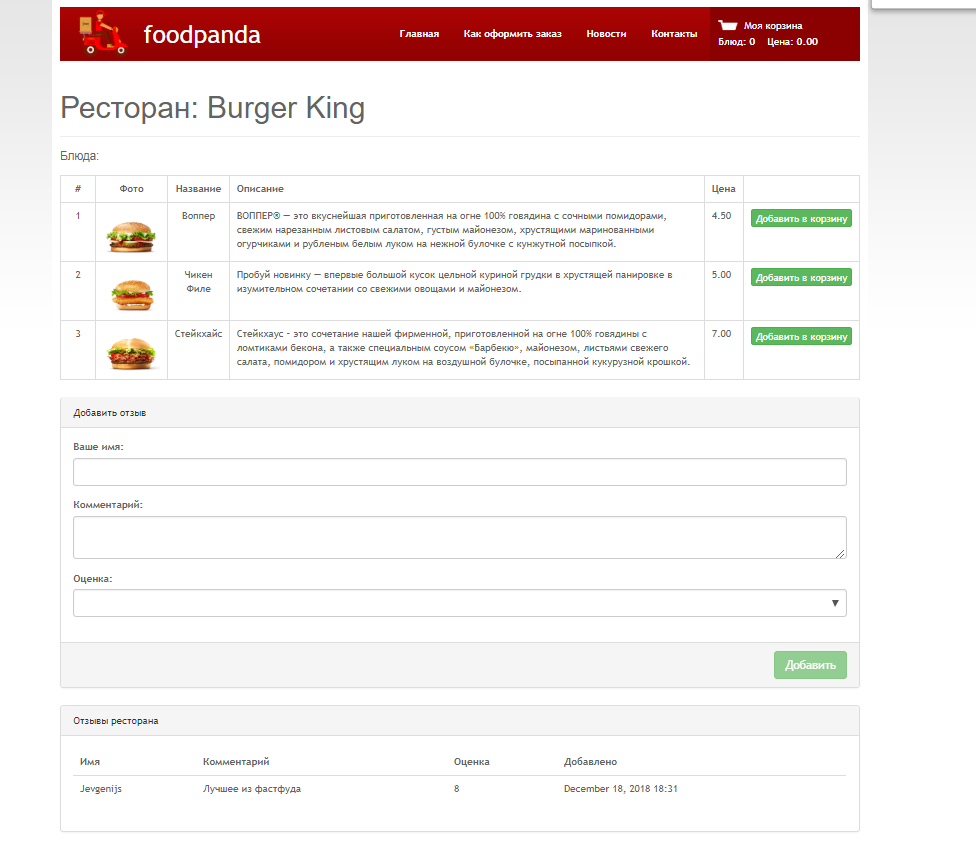


Рисунок 8.8 – Страница ресторана «Burger King»

Для добавления блюда в корзину достаточно нажать на кнопку «Добавить в корзину».

Для того, чтобы добавить новый отзыв, достаточно ввести свое имя, выбрать оценку и, если есть желание, написать комментарий к ресторану (рисунок 8.9). На основе этих отзывов формируется оценка ресторана, которая потом отображена возле названия ресторана на главной странице.



Рисунок 8.9 – Заполнение формы отзыва на ресторан

После добавления блюда в корзину, кнопка добавления автоматически меняет свой вид (рисунок 8.10). В то же время в правом верхнем углу изменяются краткие сведения о наполненности корзины.

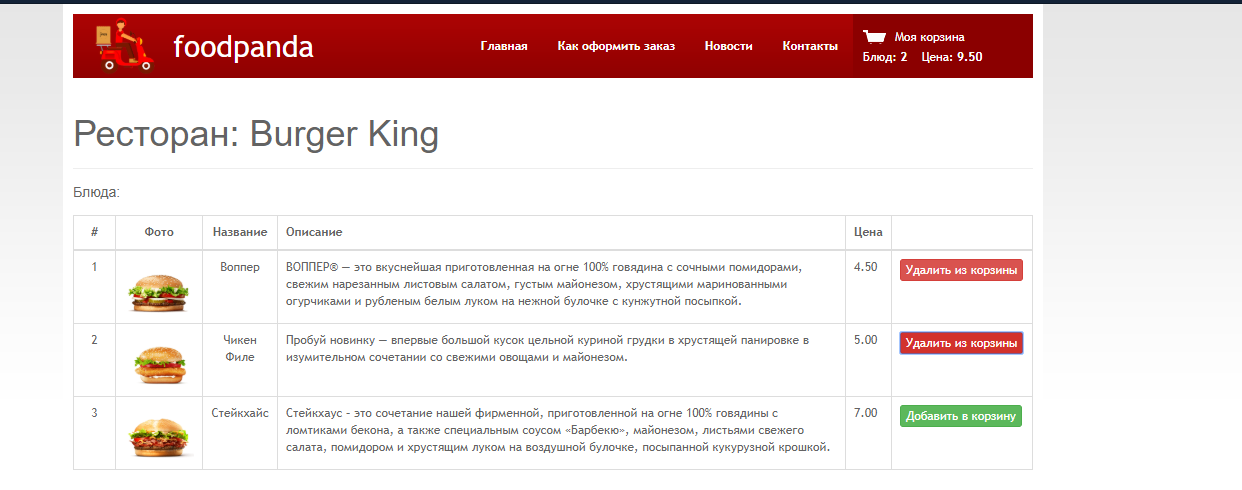


Рисунок 8.10 – Добавление блюд в корзину

После выполнения заказа необходимо открыть корзину путем нажатия на соответствующую вкладку в верхнем правом углу (рисунок 8.11).

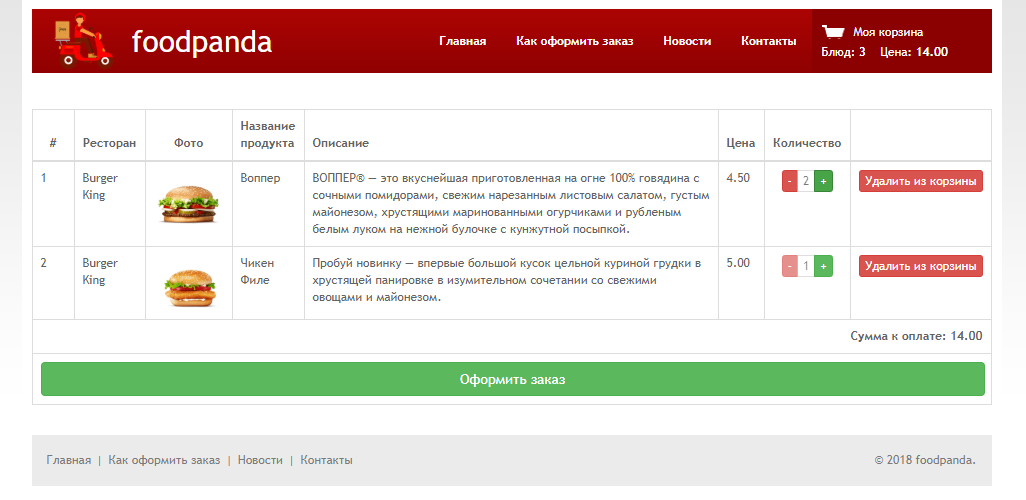


Рисунок 8.11 - Корзина

В корзине видны не только названия блюд, но и ресторан, к которому они относятся, изображения блюд и их описание. Также отсюда можно изменять количество заказанных блюд одного типа или же полностью удалять их из корзины.

Далее необходимо нажать кнопку «Оформить заказ» (рисунок 8.12). В форме необходимо заполнить имя, номер телефона и адрес, также опциональным полем является поле адреса электронной почты. После окончания заказа выведется сообщение об успехе.

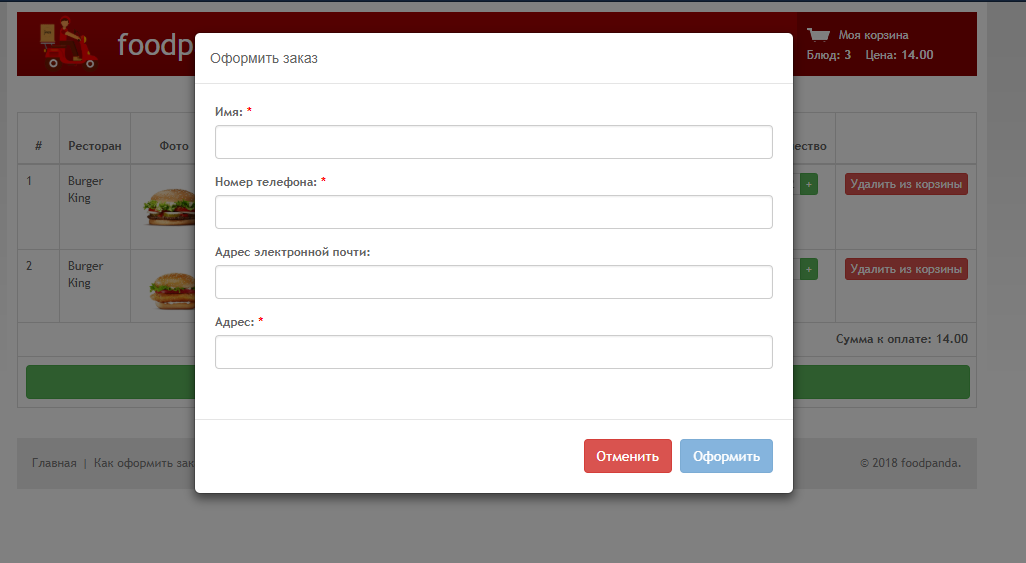


Рисунок 8.12 – Заполнение формы заказа

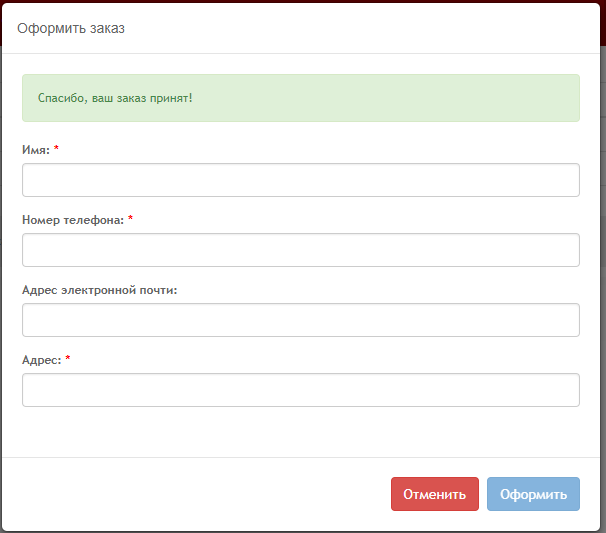


Рисунок 8.13 – Успешное оформление заказа

# 9 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ

В системе осуществлены различные способы предупреждение ошибок по невнимательности пользователя.

В большинстве случаев пользователю не дается возможность продолжить свои действия до того, как он проведет заполнение форм в верном формате. Например, во время написания отзыва (рисунок 9.1, рисунок 9.2) или заполнения формы заказа (рисунок 9.3, рисунок 9.4) кнопка является заблокированной до того момента, пока все данные не будут введены корректно.

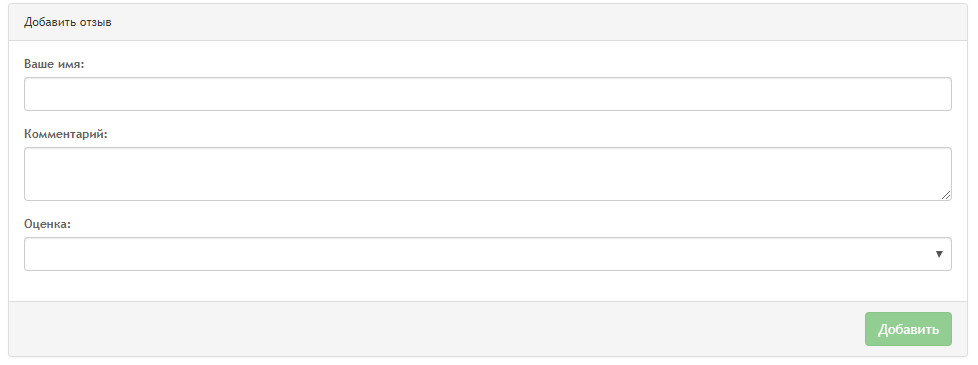


Рисунок 9.1 – Кнопка «Добавить» заблокирована

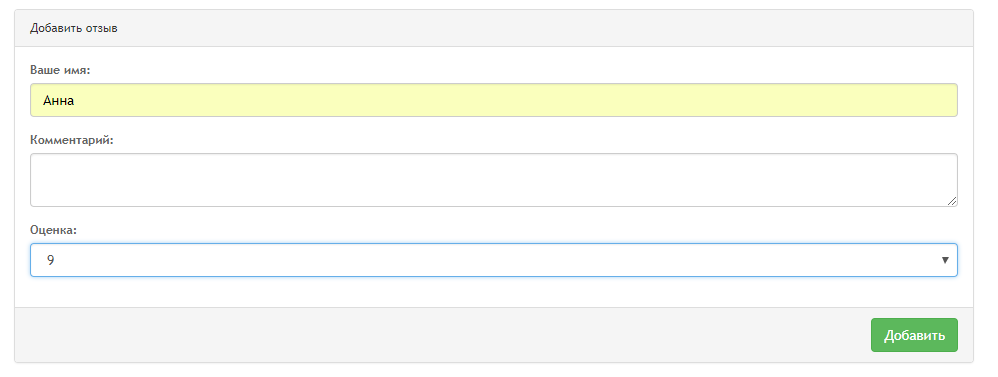


Рисунок 9.2 – Кнопка «Добавить» доступна

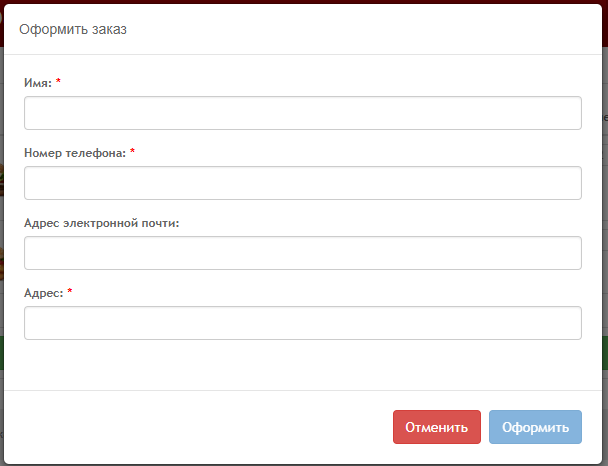


Рисунок 9.3 – Кнопка «Оформить» заблокирована

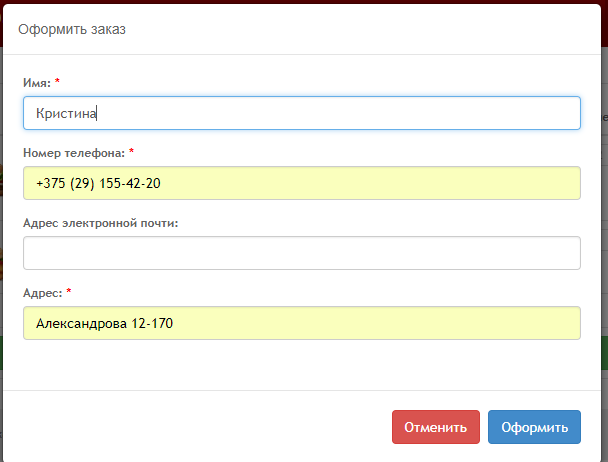


Рисунок 9.4 – Кнопка «Оформить» доступна

Пустой заказ оформить также нельзя, в таком случае в окне корзины будет выведена надпись, что корзина пуста (рисунок 9.5).

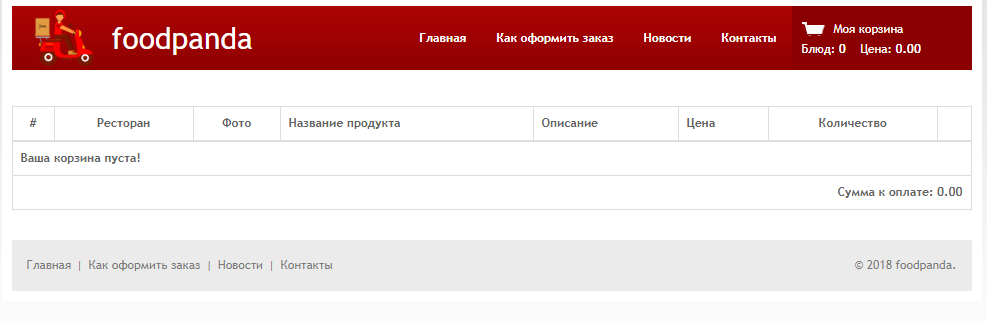


Рисунок 9.5 – Пустая корзина

Корзина привязана к сессии, поэтому при перезагрузках страниц корзина очищаться не будет.

При выборе фильтрации таким образом, что к ней не будет относиться ни один ресторан, также выведется сообщение о том, что ничего не было найдено (рисунок 9.6).

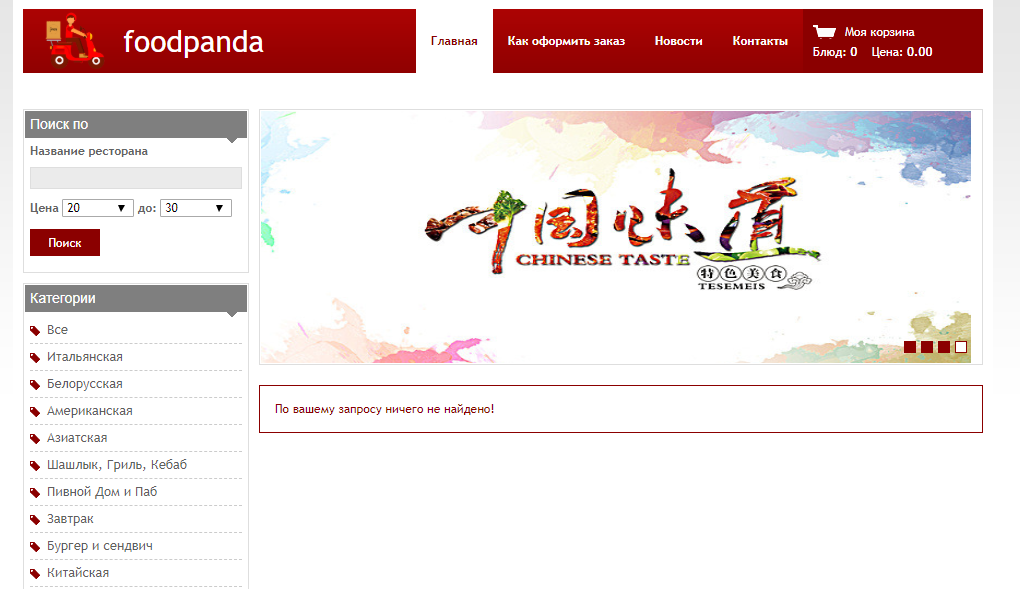


Рисунок 9.6 – Сообщение об отсутствии подходящих ресторанов

# **ЗАКЛЮЧЕНИЕ**

В результате разработки курсового проекта была создана система автоматизации доставки еды различных кафе и ресторанов.

Были изучены новейшие технологии, фреймворки и библиотеки JavaScript: Node.js, express, Vue.js, TWIG, axious, также была разработана база данных и изучены «best practices» в сфере разработки дизайна сайта.

База данных была приведена к 3 нормальной форме, все данные в ней связаны, каждый неключевой атрибут функционально зависит от ключа соответствующей таблицы, отсутствуют транзитивные функциональные зависимости ключевых и неключевых атрибутов.

Была тщательно изучена предметная область, а также уже существующие решения создания агрегаторов доставки еды. Был спроектирован принцип работы приложения и позже реализован средствами JavaScript.

В соответствии со всеми требованиями к интерфейсу, он не является перегруженным и имеет приятный дизайн, что позволяет с первых секунд расположить к себе клиента. Все использованные технологии позволяют сделать систему гибкой, легко модифицируемой. Например, в будущем возможна функция добавления программ лояльности постоянным клиентам.

Таким образом, данное приложение позволяет различным ресторанам значительно упростить процесс создания системы доставки еды из их ресторана путем предоставления услуг уже сформированной команды доставки, а также уже размещения их ресторана на сайте. Кроме того, есть возможность размещения рекламы новых ресторанов.

Все поставленные цели и задачи этого проекта были выполнены с учетом требований к системе.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Как работают агрегаторы по доставке еды [Электронный ресурс]. – Электронные данные. – Режим доступа: https://rb.ru/opinion/obed/

[2] Development of Mobile Apps for Food Delivery Services [Электронный ресурс]. – Электронные данные. – Режим доступа: https://mlsdev.com/blog/71-development-of-mobile-apps-for-food-delivery-services

[3] Кейс Dinarys №2 E-commerce сервисы по доставке еды [Электронный ресурс]. – Электронные данные. – Режим доступа: https://dinarys.com/ru/blog/Food\_Delivery\_Apps

[4] Три самых популярных языка программирования в 2018 году [Электронный ресурс]. – Электронные данные. – Режим доступа: https://rb.ru/story/top-3-programming-languages/

[5] Диаграмма развёртывания [Электронный ресурс]. – Электронные данные. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0\_%D1%80%D0%B0%D0%B7%D0%B2%D1%91%D1%80%D1%82%D1%8B%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F

[6] Богданова К.М. “Автоматизация работы сети ресторанов с модулем складского и бухгалтерского учета”: курсовой проект – Минск, 2017.

[7] Богданова К.М. “Автоматизация работы сети ресторанов с модулем бухгалтерского учёта и онлайн бронированием”: курсовой проект – Минск, 2018.

[8] Express. Руководство [Электронный ресурс]. – Электронные данные. – Режим доступа: https://expressjs.com/ru/guide/routing.html

[9] Начало работы с Node.js [Электронный ресурс]. – Электронные данные. – Режим доступа: https://medium.com/devschacht/node-hero-chapter-1-239f7afeb1d1

# Приложение А (Листинг кода) (Обязательное)

app.js – запуск проекта

const path = require('path')

const logger = require('morgan')

const routes = require('./routes')

const express = require('express')

const session = require('express-session')

const cookieParser = require('cookie-parser')

const app = express()

const port = 3030

app.use(cookieParser())

app.use(session({

resave: false,

saveUninitialized: true,

secret: 'kVrcDSFlL4pBFHgY3EWX'

}))

app.use(logger('dev'))

app.use(express.json())

app.use(express.urlencoded({

extended: false

}))

app.use(routes)

app.use(express.static(path.join(\_\_dirname, 'public')))

app.use((err, req, res, next) => {

console.error(err.stack)

res.status(500).send('Something broke!')

})

app.use((req, res, next) => {

res.status(404).send('Sorry can\'t find that!')

})

app.disable('etag')

app.listen(port)

console.log('Running @ Port: ' + port)

module.exports = app

db.js – подключение к базе данных

const mysql = require('mysql')

const config = require('./config')

const connection = mysql.createConnection(config.db)

// Use PDO style query statements.

connection.config.queryFormat = function (query, values) {

if (!values) return query

return query.replace(/\:(\w+)/g, function (text, key) {

if (values.hasOwnProperty(key)) {

return this.escape(values[key])

}

return text

}.bind(this))

}

connection.connect(function (err) {

if (err) throw err

})

module.exports = connection

routes.js – фрагмент обработки запросов с клиента.

// ----------------------------

// API routes - place order

// ----------------------------

router.post('/api/order', (req, res) => {

OrderService.placeOrder(req.session.id, req.body).then(() => {

res.json({

status: 'success'

})

}).catch(error => {

console.error('Unable to create an order: ', error)

res.status(500).json({

message: error.message

})

})

})

RestaurantService – API работы с рестораном (вывод блюд, фильтрации и т.д.)

const db = require('../db')

const perPage = 20

const RestaurantService = {

getQuery(params, forCount = false) {

let fields = forCount ? 'COUNT(r.id) as count' : 'r.\*, c.title as category'

let whereString = 'WHERE is\_active = 1 '

if (params.search) {

params.search = `%${params.search}%`

whereString += `AND name LIKE ${db.escape(params.search)} `

}

if (params.category\_id) {

whereString += `AND category\_id = :category\_id `

}

if ((params.price\_from && !isNaN(parseInt(params.price\_from))) || (params.price\_to && !isNaN(parseInt(params.price\_to)))) {

let subQuery

if (params.price\_from && params.price\_to) {

subQuery = `BETWEEN ${parseInt(params.price\_from)} AND ${parseInt(params.price\_to)}`

} else {

subQuery = params.price\_from ? `> ${parseInt(params.price\_from)}` : `< ${parseInt(params.price\_to)}`

}

whereString += `AND EXISTS (

SELECT 1 FROM restaurant\_dishes WHERE restaurant\_dishes.restaurant\_id = r.id AND restaurant\_dishes.price ${subQuery}

) `

}

if (params.page && !forCount) {

let offset = parseInt((params.page - 1) \* perPage)

whereString += `LIMIT ${perPage} OFFSET ${offset} `

}

return `

SELECT

${fields}

FROM

restaurants r

LEFT JOIN

categories c ON (c.id = r.category\_id)

${whereString}

`

},

getAllRestaurants(params) {

let countPromise = new Promise((resolve, reject) => {

let query = this.getQuery(params, true)

db.query(query, params, function (error, results) {

error ? reject(error) : resolve(results && results.length ? results[0].count : 0)

})

})

return countPromise.then(count => {

let data = {

pages: 0,

perPage,

total: count,

items: [],

}

data.pages = Math.ceil(count / perPage)

return new Promise((resolve, reject) => {

let query = this.getQuery(params, false)

db.query(query, params, function (error, results) {

data.items = results ? results : []

error ? reject(error) : resolve(data)

})

})

})

},

findById(id) {

let queryRestaurant = `SELECT r.\*, c.title as category FROM restaurants r LEFT JOIN categories c ON (c.id = r.category\_id) WHERE r.id = :id`

let queryDishes = `SELECT \* FROM restaurant\_dishes rd WHERE rd.restaurant\_id = :id`

let queryFeedbacks = `SELECT \* FROM restaurant\_feedbacks rf WHERE rf.restaurant\_id = :id ORDER BY created\_at DESC`

let basePromise = new Promise((resolve, reject) => {

db.query(queryRestaurant, {id}, function (error, results) {

error ? reject(error) : results.length ? resolve(results[0]) : reject()

})

})

let dishesPromise = new Promise((resolve, reject) => {

db.query(queryDishes, {id}, function (error, results) {

error ? reject(error) : resolve(results)

})

})

let feedbacksPromise = new Promise((resolve, reject) => {

db.query(queryFeedbacks, {id}, function (error, results) {

error ? reject(error) : resolve(results)

})

})

return Promise.all([basePromise, dishesPromise, feedbacksPromise]).then(values => {

let restaurant = values[0]

restaurant.dishes = values[1]

restaurant.feedbacks = values[2]

return restaurant

})

},

addFeedback(id, form) {

let query = `INSERT INTO restaurant\_feedbacks (restaurant\_id, first\_name, rate, comment) VALUES (:id, :first\_name, :rate, :comment)`

return new Promise((resolve, reject) => {

db.query(query, {

id: id,

rate: form.rate,

comment: form.comment,

first\_name: form.name

}, function (error, results) {

let rateQuery = `

UPDATE

restaurants r

LEFT JOIN (

SELECT

rfinner.restaurant\_id, AVG(rfinner.rate) as avgrate

FROM

restaurant\_feedbacks rfinner

GROUP BY

rfinner.restaurant\_id

) rf ON (rf.restaurant\_id = r.id)

SET

r.average\_rating = rf.avgrate

WHERE

r.id = :id

`

// Update average rating for given restaurant.

db.query(rateQuery, {id: id}, function (\_error, \_results) {

if (\_error) console.error(\_error)

error ? reject(error) : resolve(results.affectedRows)

})

})

})

}

}

module.exports = RestaurantService

app.js – статический js, относящийся к работе клиента. Здесь происходит отправка запросов на сервер, проверки данных и изменение состояний элементов..

new Vue({

el: '#app',

components: {

'order-form': orderForm

},

data: {

categories: [],

restaurants: {

items: [],

pages: 0,

total: 0,

perPage: 0,

},

restaurantFilters: {

page: 1,

search: null,

price\_from: null,

price\_to: null

},

restaurantCommentForm: {

name: '',

comment: '',

rate: null

},

cartItems: []

},

created: function () {

if (foodPanda && foodPanda.isHomepage) {

this.getCategories()

this.getRestaurants()

}

this.getCartItems()

},

mounted: function() {

$("#slider-holder ul").jcarousel({

// scroll: 1,

// auto: 6,

// wrap: 'both',

//initCallback: \_init\_slider,

//itemFirstInCallback: \_active\_slide,

//buttonNextHTML: null,

//buttonPrevHTML: null

});

},

computed: {

totalCartItems: function () {

return this.cartItems.reduce(function (acc, item) {

return acc + item.quantity

}, 0)

},

totalCartSum: function () {

return this.cartItems.reduce(function (acc, item) {

return acc + (item.quantity \* item.price)

}, 0).toFixed(2)

},

canLeaveComment: function () {

return this.restaurantCommentForm.name.trim().length && this.restaurantCommentForm.rate

}

},

methods: {

getCategories: function () {

axios.get('/api/categories').then(function (response) {

this.categories = response.data

}.bind(this)).catch(function (error) {

console.error(error)

})

},

getRestaurants: function () {

axios.get('/api/restaurants', {

params: this.restaurantFilters

}).then(function (response) {

this.restaurants = response.data

}.bind(this)).catch(function (error) {

console.error(error)

})

},

getCartItems() {

axios.get('/api/cart').then(function (response) {

this.cartItems = response.data

}.bind(this)).catch(function (error) {

console.error(error)

})

},

filterRestaurants: function (filter, value) {

if (filter !== 'page') {

this.restaurantFilters.page = 1

}

this.restaurantFilters[filter] = value

this.getRestaurants()

},

searchRestaurants: function () {

this.restaurantFilters.page = 1

this.$nextTick(this.getRestaurants)

},

isInCart: function (id) {

var inCart = false

$.each(this.cartItems, function (index, item) {

if (parseInt(id) === item.product\_id) {

inCart = true

}

})

return inCart

},

addToCart: function (id, quantity = 1) {

axios.post('/api/cart/' + id, {quantity: quantity}).then(function () {

this.getCartItems()

}.bind(this)).catch(function (error) {

console.error(error)

alert('Неизвестная ошибка')

})

},

removeFromCart: function (id) {

axios.delete('/api/cart/' + id).then(function () {

this.getCartItems()

}.bind(this)).catch(function (error) {

console.error(error)

alert('Неизвестная ошибка')

})

},

placeOrder: function () {

console.log('asd')

}

}

})

# Приложение Б (Листинг генерации базы данных) (Обязательное)

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Sequel Pro SQL dump

# Version 4541

#

# http://www.sequelpro.com/

# https://github.com/sequelpro/sequelpro

#

# Host: 192.168.99.100 (MySQL 5.7.24)

# Database: foodpanda

# Generation Time: 2018-12-18 14:30:42 +0000

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

# Dump of table cart\_items

# ------------------------------------------------------------

DROP TABLE IF EXISTS `cart\_items`;

CREATE TABLE `cart\_items` (

`id` int(11) unsigned NOT NULL AUTO\_INCREMENT,

`session\_id` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL DEFAULT '',

`product\_id` int(11) NOT NULL,

`quantity` int(11) NOT NULL DEFAULT '1',

PRIMARY KEY (`id`),

UNIQUE KEY `session\_id\_2` (`session\_id`,`product\_id`),

KEY `session\_id` (`session\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

# Dump of table categories

# ------------------------------------------------------------

DROP TABLE IF EXISTS `categories`;

CREATE TABLE `categories` (

`id` int(11) unsigned NOT NULL AUTO\_INCREMENT,

`title` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL DEFAULT '',

`created\_at` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

LOCK TABLES `categories` WRITE;

/\*!40000 ALTER TABLE `categories` DISABLE KEYS \*/;

INSERT INTO `categories` (`id`, `title`, `created\_at`)

VALUES

(1,'Итальянская','2018-12-15 19:35:21'),

(2,'Белорусская','2018-12-15 19:35:21'),

(3,'Американская','2018-12-15 19:35:21'),

(4,'Азиатская','2018-12-15 19:35:21'),

(5,'Шашлык, Гриль, Кебаб','2018-12-15 19:35:21'),

(6,'Пивной Дом и Паб','2018-12-15 19:35:21'),

(7,'Завтрак','2018-12-15 19:35:21'),

(8,'Бургер и сендвич','2018-12-15 19:35:21'),

(9,'Китайская','2018-12-15 19:35:21'),

(10,'Восточная','2018-12-15 19:35:21'),

(11,'Европейская','2018-12-15 19:35:21'),

(12,'Французская','2018-12-15 19:35:21'),

(13,'Грузинская','2018-12-15 19:35:21'),

(14,'Греческая','2018-12-15 19:35:21'),

(15,'Индийская','2018-12-15 19:35:21'),

(16,'Японская','2018-12-15 19:35:21'),

(17,'Обеды','2018-12-15 19:35:21'),

(18,'Мексиканская','2018-12-15 19:35:21'),

(19,'Паста','2018-12-15 19:35:21'),

(20,'Пироги','2018-12-15 19:35:21'),

(21,'Пицца','2018-12-15 19:35:21'),

(22,'Шаурма','2018-12-15 19:35:21'),

(23,'Стейки','2018-12-15 19:35:21'),

(24,'Суши','2018-12-15 19:35:21'),

(25,'Тайская','2018-12-15 19:35:21'),

(26,'Хот-дог','2018-12-15 19:35:21'),

(27,'Драники, Блины','2018-12-15 19:35:21'),

(28,'Здоровая еда','2018-12-15 19:35:21'),

(29,'Узбекская','2018-12-15 19:35:21'),

(30,'Турецкая','2018-12-15 19:35:21'),

(31,'Органическая еда','2018-12-15 19:35:21'),

(32,'Специальное Предложение','2018-12-15 19:35:21'),

(33,'Домашняя еда','2018-12-15 19:35:21'),

(34,'ВОК','2018-12-15 19:35:21');

/\*!40000 ALTER TABLE `categories` ENABLE KEYS \*/;

UNLOCK TABLES;

# Dump of table order\_items

# ------------------------------------------------------------

DROP TABLE IF EXISTS `order\_items`;

CREATE TABLE `order\_items` (

`id` int(11) unsigned NOT NULL AUTO\_INCREMENT,

`order\_id` int(11) unsigned NOT NULL,

`product\_id` int(11) unsigned NOT NULL,

`quantity` int(11) unsigned NOT NULL DEFAULT '1',

`created\_at` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (`id`),

KEY `order\_items\_order\_id\_foreign` (`order\_id`),

KEY `order\_items\_product\_foreign` (`product\_id`),

CONSTRAINT `order\_items\_order\_id\_foreign` FOREIGN KEY (`order\_id`) REFERENCES `orders` (`id`) ON DELETE CASCADE ON UPDATE NO ACTION,

CONSTRAINT `order\_items\_product\_foreign` FOREIGN KEY (`product\_id`) REFERENCES `restaurant\_dishes` (`id`) ON UPDATE NO ACTION

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

# Dump of table orders

# ------------------------------------------------------------

DROP TABLE IF EXISTS `orders`;

CREATE TABLE `orders` (

`id` int(11) unsigned NOT NULL AUTO\_INCREMENT,

`session\_id` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL DEFAULT '',

`name` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`phone` varchar(100) COLLATE utf8mb4\_unicode\_ci NOT NULL DEFAULT '',

`email` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`address` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`total\_price` double(8,2) unsigned NOT NULL DEFAULT '0.00',

`created\_at` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

# Dump of table restaurant\_dishes

# ------------------------------------------------------------

DROP TABLE IF EXISTS `restaurant\_dishes`;

CREATE TABLE `restaurant\_dishes` (

`id` int(11) unsigned NOT NULL AUTO\_INCREMENT,

`restaurant\_id` int(11) unsigned NOT NULL,

`name` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL DEFAULT '',

`description` text COLLATE utf8mb4\_unicode\_ci NOT NULL,

`price` double(8,2) unsigned NOT NULL,

`image` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT '',

`is\_active` tinyint(1) unsigned NOT NULL DEFAULT '1',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

LOCK TABLES `restaurant\_dishes` WRITE;

/\*!40000 ALTER TABLE `restaurant\_dishes` DISABLE KEYS \*/;

INSERT INTO `restaurant\_dishes` (`id`, `restaurant\_id`, `name`, `description`, `price`, `image`, `is\_active`)

VALUES

(1,1,'Воппер','ВОППЕР® — это вкуснейшая приготовленная на огне 100% говядина с сочными помидорами, свежим нарезанным листовым салатом, густым майонезом, хрустящими маринованными огурчиками и рубленым белым луком на нежной булочке с кунжутной посыпкой.',4.50,'/images/menu/bk\_vopper.png',1),

(2,1,'Чикен Филе','Пробуй новинку — впервые большой кусок цельной куриной грудки в хрустящей панировке в изумительном сочетании со свежими овощами и майонезом.',5.00,'/images/menu/chiken\_file.png',1),

(3,1,'Стейкхайс','Стейкхаус – это сочетание нашей фирменной, приготовленной на огне 100% говядины с ломтиками бекона, а также специальным соусом «Барбекю», майонезом, листьями свежего салата, помидором и хрустящим луком на воздушной булочке, посыпанной кукурузной крошкой.',7.00,'/images/menu/stakehouse.png',1);

/\*!40000 ALTER TABLE `restaurant\_dishes` ENABLE KEYS \*/;

UNLOCK TABLES;

# Dump of table restaurant\_feedbacks

# ------------------------------------------------------------

DROP TABLE IF EXISTS `restaurant\_feedbacks`;

CREATE TABLE `restaurant\_feedbacks` (

`id` int(11) unsigned NOT NULL AUTO\_INCREMENT,

`restaurant\_id` int(11) unsigned NOT NULL,

`first\_name` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL DEFAULT '',

`rate` tinyint(2) unsigned NOT NULL,

`comment` text COLLATE utf8mb4\_unicode\_ci,

`created\_at` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (`id`),

KEY `restaurant\_feedbacks\_restaurant\_foreign` (`restaurant\_id`),

CONSTRAINT `restaurant\_feedbacks\_restaurant\_foreign` FOREIGN KEY (`restaurant\_id`) REFERENCES `restaurants` (`id`) ON DELETE CASCADE ON UPDATE NO ACTION

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

# Dump of table restaurants

# ------------------------------------------------------------

DROP TABLE IF EXISTS `restaurants`;

CREATE TABLE `restaurants` (

`id` int(11) unsigned NOT NULL AUTO\_INCREMENT,

`category\_id` int(11) unsigned DEFAULT NULL,

`name` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL DEFAULT '',

`is\_active` tinyint(1) NOT NULL DEFAULT '1',

`image` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`schedule` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`average\_rating` double(8,2) NOT NULL DEFAULT '0.00',

PRIMARY KEY (`id`),

KEY `restaurants\_category\_id\_foreign` (`category\_id`),

CONSTRAINT `restaurants\_category\_id\_foreign` FOREIGN KEY (`category\_id`) REFERENCES `categories` (`id`) ON DELETE SET NULL ON UPDATE NO ACTION

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

LOCK TABLES `restaurants` WRITE;

/\*!40000 ALTER TABLE `restaurants` DISABLE KEYS \*/;

INSERT INTO `restaurants` (`id`, `category\_id`, `name`, `is\_active`, `image`, `schedule`, `average\_rating`)

VALUES

(1,8,'Burger King',1,'/images/rest1.jpeg','10:00-00:00',6.00),

(2,21,'Пицца Лисицца',1,'/images/rest2.jpeg','5:00-3:00',0.00),

(3,5,'Ёлки Палки',1,'/images/rest3.jpeg','11:00-2:00',1.00),

(4,2,'Друзья',1,'/images/rest4.jpeg','9:00-23:00',0.00),

(5,24,'Sushi House',1,'/images/rest5.jpeg','10:00-23:00',0.00);

/\*!40000 ALTER TABLE `restaurants` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/;

/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;