МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Программной инженерии

Специальность 1-98 01 03 Программное обеспечение информационной безопасности мобильных систем

Направление специальности 1-98 01 03 Программное обеспечение информационной безопасности мобильных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Современные технологии программирования мобильных систем»

Тема Программное средство «Ресторан Klingen»

Исполнитель

студент (ка) 2 курса группы 7 Тарайкович Владислав Игоревич

(Ф.И.О.)

Руководитель работы ассистент Панченко О.Л.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Пацей Н.В.

(подпись)

Минск 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра программной инженерии

Утверждаю

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ Н.В. Пацей

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020г.

**ЗАДАНИЕ**

**к курсовому проектированию**

**по дисциплине** "Современные технологии программирования мобильных систем"

|  |  |
| --- | --- |
| Специальность: 1-98 01 03 Программное обеспечение информационной безопасности мобильных систем  Студент: Тарайкович В.И. | Группа: 7 |
| **Тема: Программное средство «Ресторан Klingen»** | |

**1. Срок сдачи студентом законченной работы**: "20 мая 2020 г."

**2. Исходные данные к проекту:**

**2.1**. Функционально ПС поддерживает:

* Функции администратора:
  + Поддерживать работу c базой данных;
  + Выполнять поисковые запросы;
  + Просматривать забронированные столики;
  + Заказ продуктов для ресторана;
  + Проверка наличия продуктов на складе.
* Функции клиента:
  + Выполнять регистрацию и авторизацию;
  + Бронировать столик;
  + Просматривать меню ресторана;
  + Просматривать забронированные столики;
  + Оставлять отзывы о ресторане.

**2.2.** При выполнении курсового проекта необходимо использовать принципы и приемы ООП. Приложение разрабатывается под ОС Windows и представляет собой настольное приложение (desktop). Отображение, бизнес логика должны быть максимально независимы друг от друга для возможности расширения. Диаграммы вариантов использования, классов реализации задачи, взаимодействия разработать на основе UML. Язык разработки проекта – C#. Управление программой должно быть интуитивно понятным и удобным. При разработке использовать несколько наиболее подходящих шаблонов проектирования ПО.

**3. Содержание расчетно-пояснительной записки**

(перечень вопросов, подлежащих разработке)

* Введение
* Постановка задачи и обзор литературы (алгоритмы решения, обзор прототипов, актуальность задачи)
* Проектирование архитектуры проекта (структура модулей, классов).
* Разработка функциональной модели и модели данных ПС (выполняемые функции)
* Тестирование
* Заключение
* Список используемых источников
* Приложения

**4. Форма представления выполненного курсового проекта:**

* + Теоретическая часть курсового проекта должна быть представлена в формате docx. Оформление записки должно быть согласно выданным правилам.
  + Листинги программы представляются частично в приложении.
  + Пояснительную записку, листинги, проект (инсталляцию проекта) необходимо загрузить на диск, указанный преподавателем.

#### Календарный план

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 19.02.2020 |  |
| 2 | Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования | 12.03.2020 |  |
| 3 | Анализ и проектирование архитектуры приложения (построение диаграмм, проектирование бизнес-слоя, представления и данных) | 26.03.2020 |  |
| 4 | Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса | 2.04.2020 |  |
| 5 | Кодирование программного средства | 23.04.2020 |  |
| 6 | Тестирования и отладка программного средства | 30.04.2020 |  |
| 7 | Оформление пояснительной записки | 7.05.2020 |  |
| 9 | Сдача проекта | 20.05.2020 |  |

**5. Дата выдачи задания** 12.02.2020 г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *О. Л. Панченко*

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата и подпись студента)

Содержание

[ВВЕДЕНИЕ 6](#_Toc41541454)

[1 Аналитический обзор прототипов и литературных источников 7](#_Toc41541455)

[2 Анализ требований к программному средству и разработка функциональных требований 10](#_Toc41541456)

[3 Проектирование программного средства 12](#_Toc41541457)

[3.1 Проектирование архитектуры приложения 12](#_Toc41541458)

[3.2 Проектирование базы данных 15](#_Toc41541459)

[3.3 Проектирование доступа к базе данных 18](#_Toc41541460)

[3.4 Проектирование логики сценариев использования 18](#_Toc41541461)

[3.5 Проектирование доступа к сервисам приложения 20](#_Toc41541462)

[4 Реализация программного средства 21](#_Toc41541463)

[4.1 Реализация сущностей 21](#_Toc41541464)

[4.2 Реализация архитектуры MVVM 22](#_Toc41541465)

[4.3 Реализация доступа к сервисам приложения 23](#_Toc41541466)

[4.4 Реализация представления 24](#_Toc41541467)

[5 Тестирование, проверка работоспособности и анализ полученных результатов 25](#_Toc41541468)

[6 Руководство по использованию 32](#_Toc41541469)

[ЗАКЛЮЧЕНИЕ 36](#_Toc41541470)

[СПИСОК ЛИТЕРАТУРЫ 37](#_Toc41541471)

[ПРИЛОЖЕНИЕ А 38](#_Toc41541472)

[ПРИЛОЖЕНИЕ Б 39](#_Toc41541473)

[ПРИЛОЖЕНИЕ В 43](#_Toc41541474)

[ПРИЛОЖЕНИЕ Г 44](#_Toc41541475)

[ПРИЛОЖЕНИЕ Д 47](#_Toc41541476)

ВВЕДЕНИЕ

Данный курсовой проект посвящён разработке программного средства «Ресторан Klingen», основной целью которого является автоматизация работы ресторана. В настоящее время существует малое количество аналогов, реализованных в качестве десктопных приложений. Большинство таких сервисов представлено в виде Web-сайтов.

Приложение «Ресторан Klingen» предназначено для комплексной автоматизации работы ресторана. Данная программа создана для решения всего спектра задач, связанного с оформлением заказов, просмотром и добавлением пунктов меню, новостей ресторана, учетом продуктов на складе и заказ необходимых продуктов, бронирования столиков и многим другим. Исходя из этого, повышается производительность труда, скорость оформления заказов и общая прибыль фирмы. Программа интуитивно проста для пользователей, так как не требует квалифицируемых ИТ-ресурсов.

Также, большое внимание я уделил качеству UI. Так как в настоящее время, чтобы выделится среди конкурентов, необходимо давать пользователям что-то новое и интересное, ведь большинство приложений имеют схожий функционал.

В качестве интерфейса приклaдного программирования был выбран обширный API-интерфейс — Windows Presentation Foundation (WPF), предназначенный для создания настольных программ с графически насыщенным пользовательским интерфейсом.

Для работы с WPF использовался объектно-ориентированный язык программирования с С-подобным синтаксисом — С#, разработанный для создания приложений на платформе Microsoft .NET Framework.

Хранение данных осуществляется в Microsoft SQL Server.

1 Аналитический обзор прототипов и литературных источников

Немаловажным этапом в разработке программного продукта является аналитический обзор прототипов и литературных источников.

На сегодняшний день можно встретить достаточно большое количество программных решений, разработанных для автоматизации работы ресторана, но довольно малое количество программ имеет интуитивно понятный интерфейс. Я рассмотрел несколько программ для автоматизации, но их нельзя назвать аналогами моего ПС. Также все программы разделяются на desktop-приложения и web-приложения.

Desktop-приложения для автоматизации:

* «Presto»;
* «Restoplace».

Web-приложения:

* «LeClick»

«Presto» - облачная система для автоматизации небольших кафе, столовых, ресторанов и целых сетей. В одном окне: работа зала и кухни, учет продуктов, калькуляция блюд и себестоимость, графики смен и мотивация персонала, система лояльности и служба доставки.

Интерфейс «Presto» представлен на рисунке 1.1.

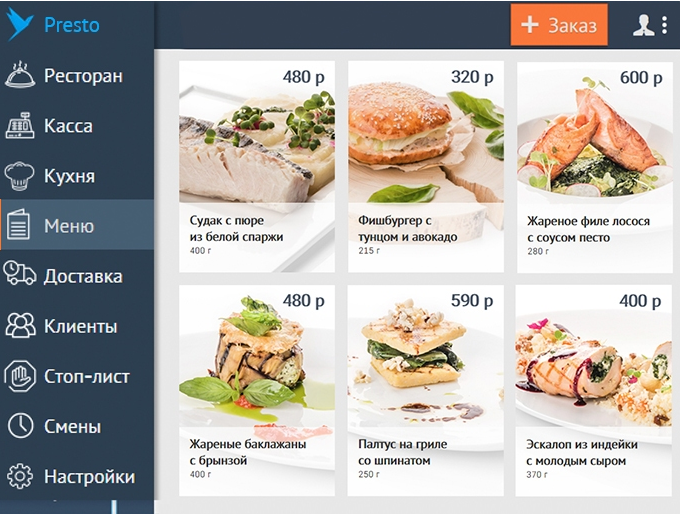


Рисунок 1.1 − Интерфейс «Presto»

Проанализировав «Presto», можно выделить основные минусы данного программного сред.

Основные минусы:

* Загруженный интерфейс;
* Отсутствие возможности работы со стороны клиента;
* Стоимость.

Основные плюсы:

* Поддерживается 1С;
* Неплохой UI;
* Надежность;

Продукт «Restoplace» - cервис бронирования столов для сайтов и соцсетей ресторанов и кафе. Онлайн приём броней, депозиты, статистика, база гостей, банкеты. Настраивается за 15 минут, работает в облаке. Стоит от 0 руб. в месяц.

Интерфейс «Restoplace» представлен на рисунке 1.2.

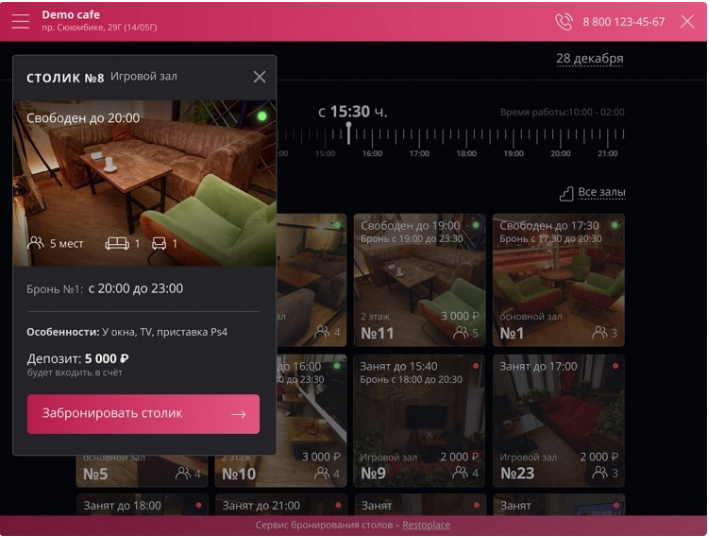


Рисунок 1.2 − Интерфейс «Restoplace»

Проанализировав ПО «Restoplace», можно выделить её основные минусы и плюсы.

Основные минусы:

* Примитивный интерфейс и плохие цвета;
* Очень маленький функционал;
* Отсутствие поиска

Основные плюсы:

* Наличие мобильного приложения;
* Простота в использовании;
* Фильтрация;

«[Fusion](https://www.livebusiness.ru/out.php?url=https://leclick.ru/for-partners.html) POS» - Облачное решение для автоматизации ресторана, кафе, магазина или кальянной. Рабочее место кассира и официанта. Управление меню, складом, лояльностью, отчеты. Интеграция с принтерами чеков и фискальными регистраторами.

Интерфейс «[Fusion](https://www.livebusiness.ru/out.php?url=https://leclick.ru/for-partners.html) POS» представлен на рисунке 1.3.

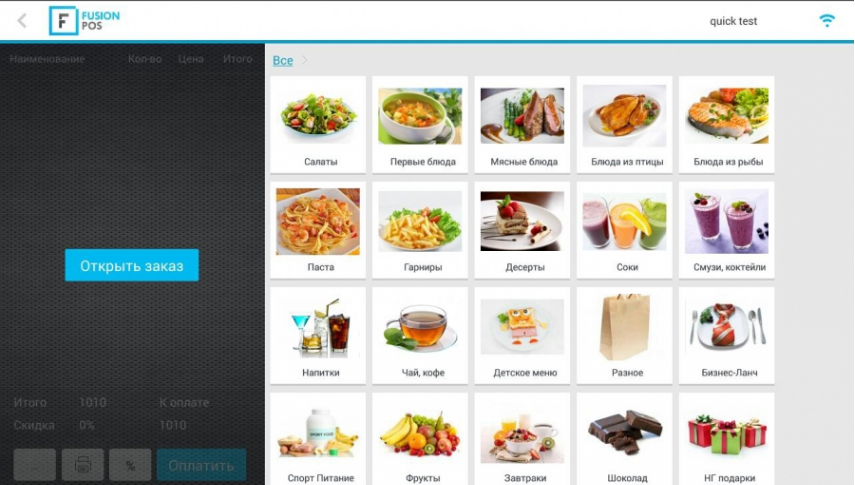


Рисунок 1.3 – Интерфейс «[Fusion](https://www.livebusiness.ru/out.php?url=https://leclick.ru/for-partners.html) POS»

Данное ПС грамотно спроектировано и выполняет большое количество функций.

Основные минусы:

* Возможность работы только со стороны администратора;
* Много лишних категорий.

Основные плюсы:

* Может интегрироваться с кассовой системой;
* Вывод чека на печать;

2 Анализ требований к программному средству и разработка функциональных требований

Анализ требований — это процесс сбора требований к программному обеспечению, их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки программного обеспечения.

Цель анализа требований в проектах — получить максимум информации о заказчике и специфике его задач, уточнить рамки проекта, оценить возможные риски. На этом этапе происходит идентификация принципиальных требований методологического и технологического характера, формулируются цели и задачи проекта, а также определяются критические факторы успеха, которые впоследствии будут использоваться для оценки результатов внедрения. Определение и описание требований — шаги, которые во многом определяют успех всего проекта, поскольку именно они влияют на все остальные этапы.

Различают три уровня требований к проекту:

* бизнес-требования;
* пользовательские требования;
* функциональные требования.

Бизнес-требования содержат высокоуровневые цели организации или заказчиков системы. Как правило, их высказывают те, кто финансируют проект, покупатели системы, менеджер реальных пользователей, отдел маркетинга. Курсовой проект не подразумевает наличие заказчика, который мог бы выдвинуть бизнес-требования, поэтому в качестве таких высокоуровневых требований можно рассматривать общие требования к разрабатываемому средству. К их числу относятся:

* простота и лёгкость интерфейса;
* использование принципов объектно-ориентированного программирования;
* использование архитектурных шаблонов проектирования;
* использование системы управления базами данных (СУБД);

Весь дальнейший процесс проектирования и разработки программного средства должен находиться в очерченных бизнес-требованиями границах.

Следующими требованиями являются требования пользователей. Данные требования описывают цели и задачи, которые пользователям позволит решить система. Таким образом, в пользовательских требованиях указано, что клиенты смогут делать с помощью системы. Пользователь данного программного решения должен иметь возможность:

* регистрировать себя в системе;
* входить в приложение, после ввода данных, необходимых для аутентификации;
* просматривать новости ресторана на главной;
* просматривать блюда в меню;
* добавление блюд в избранное;
* удаление блюд из избранного;
* оставлять отзывы о ресторане.

Повар имеет возможность:

* выполнять действия пользователя;
* просматривать наличие продуктов на складе;
* заказывать новые продукты на склад;
* заказывать имеющиеся продукты;
* использовать имеющиеся продукты.

Администратор имеет возможность:

* выполнять действия пользователя;
* вносить новую информацию о новостях, заказах, блюдах;
* удалять информацию о заказах;
* просматривать информацию о заказах;
* прикреплять изображения к информации о блюдах;
* прикреплять изображения к новостному блоку;
* осуществлять поиск по забронированным столикам;
* сортировать забронированные столики;

После проведения анализа были выявлены следующие функциональные требования:

* архитектура приложения должна соответствовать шаблонам проектирования, таким как MVVM, Command;
* вся информация должна храниться в базе данных;
* приложение должно производить валидацию вводимых пользователем данных;
* приложение должно корректным образом обрабатывать возникающие исключительные ситуации: отображать понятное для пользователя сообщение о возникшей ошибке;
* приложение должно предоставлять пользователям возможность создания нового аккаунта в виде регистрационной формы;
* приложение должно предоставлять возможность пользователям проходить аутентификацию и входить в систему под соответствующим введенным данным пользовательским именем;
* приложение должно предоставлять возможность поиска блюд по следующим критериям: название.

Таким образом, был проведен тщательный анализ требований к программному средству, который позволил разработать список функциональных требований. Разработка данной программной системы должна проводиться в соответствии с сформированными списком.

3 Проектирование программного средства

Проектирование программного средства — процесс создания проекта программного обеспечения. Целью проектировaния является определение внутренних свойств системы и детализации её внешних свойств на основе исходных условий задачи. Исходные условия задачи уже были сформулированы во втором разделе данной пояснительной записки. Этап проектирования подразумевает их анализ.

## **3.1 Проектирование архитектуры приложения**

Архитектура программного обеспечения — совокупность вaжнейших решений об организации программной системы. Архитектура включает:

* выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;
* соединение выбрaнных элементов структуры и поведения во всё более крупные системы;
* архитектурный стиль, который направляет всю организацию — все элементы, их интерфейсы, их сотрудничество и их соединение.

Для удовлетворения проектируемой системы различным атрибутам качества применяются различные архитектурные шаблоны (паттерны). В разрабатываемом приложении используется архитектурный шаблон Model-View-ViewModel (MVVM).

Шаблон MVVM имеет три основных слоя: модель, которая представляет бизнес-логику приложения, представление пользовательского интерфейса, и представление-модель, в котором содержится вся логика построения графического интерфейса и ссылка на модель, поэтому он выступает в качестве модели для представления.

На рисунке 3.1 представлена диаграмма, которая показывает общую структуру приложения в рамках шаблона MVVM.



Рисунок 3.1 – Структура шаблона MVVM

View или представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Так как пользовательский интерфейс и качество его реализации играет далеко не последнее место в конечном результате, разработка эффективного интерфейса, приятного и удобного для конечного пользователя, является важной задачей. Поэтому для хорошего проектирования View необходимо понять, как пользователь будет взаимодействовать с приложением. Для этого была составлена схема на рисунке 3.2, на которой представлен принцип работы приложения с точки зрения пользователя.

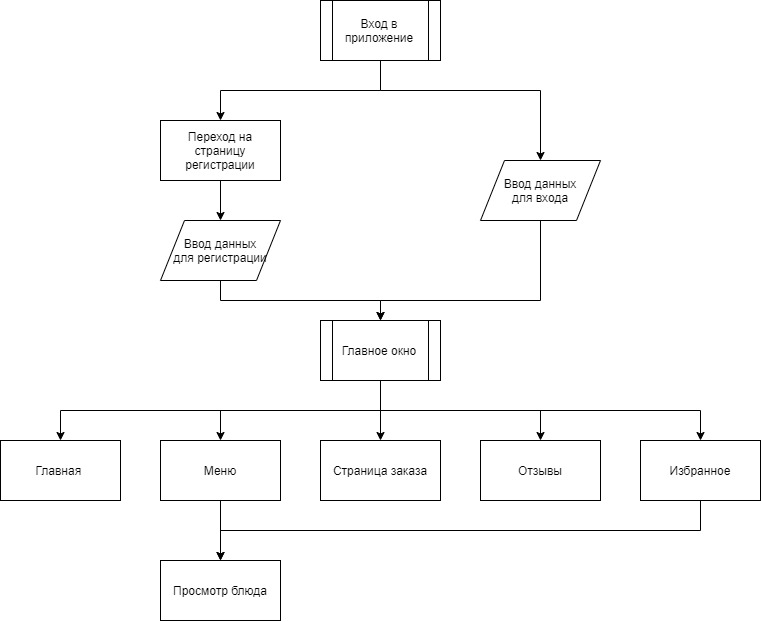


Рисунок 3.2 – Схема навигации по страницам

На диаграмме 3.2 видно, что пользователь при запуске приложения будет попадать на страницу входа, на которой чтобы перейти дальше необходимо ввести соответствующие данные или перейти на страницу регистрации и ввести данные для регистрации.

После входа в приложение пользователю откроется окно приложения с 5 вкладками:

* Главная;
* Меню;
* Заказать;
* Отзывы;
* Избранное.

На странице «Главная» будет отображаться сетка с добавленными администратором новостями. На этой странице пользователь сможет просмотреть последние новости о ресторане.

В разделе «Меню» отображаются карточки с блюдами из меню, на которых указано название, описание и цена. Можно осуществлять поиск по названию. При нажатии на кнопку «Подробнее» открывается страница «Просмотр блюда».

На странице «Просмотр блюда» можно увидеть вес, категорию и более подробное описание блюда, а также добавить его в избранное.

На странице «Заказать» размещена схема расположения столиков и форма для заполнения.

В разделе «Избранное» пользователь сможет найти блюда, которые он отметил как избранные на странице «Просмотр блюда» и перейти к информации о них.

При авторизации пользователя с ролью «Администратор» или «Повар» появляется еще один соответствующий пункт меню.

Для администратора появляется доступ к страницам «Добавить новость», «Добавить блюдо» и «Забронированные столики».

Для повара появляется доступ к страницам «Заказ продуктов» и «Склад».

ViewModel или модель представления связывает модель и представление через механизм привязки данных. Она содержит Модель, преобразованную к Представлению, а также команды, которыми может пользоваться Представление, чтобы влиять на Модель.

Model или Модель представляет собой логику работы с данными и описание фундаментальных данных, необходимых для работы приложения.

Для разработки приложения в качестве сервиса для хранения данных была выбрана СУБД Microsoft SQL Server. На диаграмме размещение на рисунке 3.3 показано графическое представление инфраструктуры приложения.

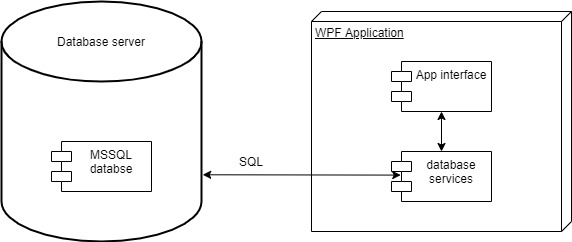


Рисунок 3.3 – Диаграмма размещения

## **3.2 Проектирование базы данных**

Проектирование баз данных — процесс создания схемы базы данных и определения необходимых ограничений целостности.

Основные задачи проектирования базы данных:

* обеспечение хранения в БД всей необходимой информации;
* обеспечение возможности получения данных по всем необходимым запросам;
* сокращение избыточности и дублирования данных;
* обеспечение целостности базы данных.

Проектирование базы данных проводится в два этапа: концептуальное (инфологическое) и логическое (даталогическое) проектирование.

Концептуальное (инфологическое) проектирование — построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня aбстракции. В результате этого этапа создаётся ER-модель. Такая модель создаётся без ориентации на какую-либо конкретную СУБД и модель данных.

Основными понятиями ER-модели являются: сущность, связь и атрибут

Сущность – это реальный или представляемый объект, информация о котором должна сохраняться и быть доступна.

Связь – это графически изображаемая ассоциация, устанавливаемая между двумя сущностями. Эта ассоциация обычно является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь).

Атрибут сущности − это любая детaль, которая служит для уточнения, идентификации, классификации, числовой характеристики или выражения состояния сущности.

В рамках этого этапа была создана ER-модель, которая включает 8 сущностей:

* пользователь;
* блюдо;
* новость;
* резервация;
* стол;
* отзыв;
* избранное блюдо;
* продукт.

Также в ER-модели были определены необходимые связи. Например, между сущностями пользователь и резервация была установлена связь один-ко-многим. Для каждой сущности были выделены aтрибуты. Например, для резервации в качестве атрибутов были выделены такие характеристики, как идентификатор резервации, идентификатор пользователя, имя клиента, номер телефона, номер столика, дата и время резервации и пожелания.

Логическое (даталогическое) проектирование — создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для реляционной модели дaнных логическая модель — набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи. На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД.

Логическая модель базы данных представлена на рисунке 3.4

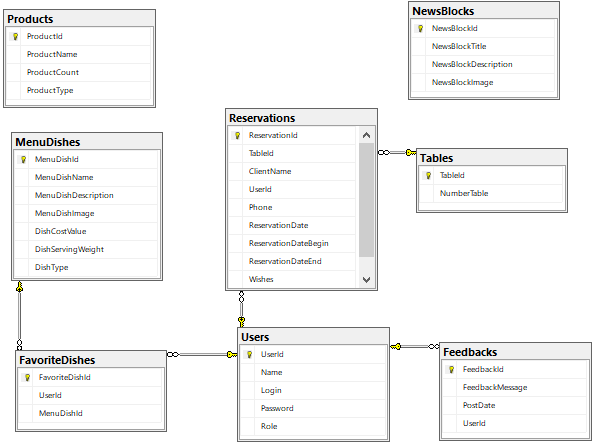


Рисунок 3.4 - Логическая модель базы данных

Всего в базе данных содержится 8 таблиц. В таблице Users хранятся все пользователи, зарегистрированные в приложении. Таблица MenuDishes содержит информацию о загруженных пунктах меню. Таблица Reservations хранит оставленные пользователями заявки на бронирование. Таблица Tables содержит номера столиков. Таблица FavoriteDishes хранит информацию о блюдах, которые пользователь отметил как избранные. Таблица Products хранит данные об имеющихся товарах на складе. Таблица NewsBlocks содержит информацию о загруженных новостях. Таблица Feedbacks хранит в себе информацию об оставленных отзывах.

Таблица Users состоит из 5 столбцов:

* UserId;
* Name;
* Login;
* Password;
* Role.

В столбце Name хранится информация о имени зарегистрированного пользователя, Login – логин, Password – пароль пользователя, Role – роль пользователя.

Таблица MenuDishes состоит из 7 столбцов:

* MenuDishId;
* MenuDishName;
* MenuDishDescription;
* MenuDishImage;
* DishCostValue;
* DishServingWeight;
* DishType;

В столбце MenuDishName хранится информация о названии блюда, в столбце MenuDishDescription – описание блюда. MenuDishImage – фото блюда, DishType – тип блюда. DishServingWeight – вес порции блюда. DishCostValue – цена блюда.

Таблица Feedbacks состоит из 5 столбцов:

* FeedbackId;
* FeedbackMessage;
* PostDate;
* UserId;

В столбце FeedbackMessage хранится отзыв о ресторане, в столбце PostDate – дата публикации комментария. UserId – идентификатор пользователя оставившего комментарий.

Таблица Tables состоит из 2 столбцов:

* TableId;
* NumberTable;

В столбце NumberTable хранится информация о номере столика.

Таблица FavoriteDishes состоит из 3 столбцов:

* FavoriteDishId;
* UserId;
* MenuDishId.

В столбце UserId – идентификатор пользователя, добавившего блюдо в избранное. MenuDishId – идентификатор блюда которое добавил пользователь в избранное.

Таблица Reservations состоит из 6 столбцов:

* ReservationId;
* TableId;
* UserId;
* ClientName;
* Phone
* ReservationDate;

В столбце TableId хранится информация о выбранном столике, UserId – идентификатор клиента, ClientName – имя клиента, Phone – номер телефона, ReservationDate – дата резервации.

Таблица Products состоит из 4 столбцов:

* ProductId;
* ProductName;
* ProductType;
* ProductCount.

В столбце ProductName хранится название продукта, ProductType – тип продукта, ProductCount – количество продукта.

Таблица NewsBlocks состоит из 4 столбцов:

* NewsBlockId;
* NewsBlockTitle;
* NewsBlockDescription;
* NewsBlockImage.

В столбце NewsBlockTitle храниться информация о заголовке новости, NewsBlockDescription – описание новости, NewsBlockImage – фото новости.

## **3.3 Проектирование доступа к базе данных**

Для доступа к базе данных используется Entity Framework 6. Этот подход предоставляет ряд существенных преимуществ: нам не нужно беспокоиться о коде доступа к данным и не нужно знать деталей работы СУБД SQL Server и синтаксиса языка запросов T-SQL, вместо этого мы работаем с таблицами базы данных как с классами C#, с полями этих таблиц - как со свойствами классов, а синтаксис SQL-запросов заменен на  [LINQ](https://professorweb.ru/my/LINQ/base/level1/info_linq.php). Entity Framework берет на себя обязанности по преобразованию кода C# в SQL-инструкции.

Существуют 3 подхода по проектированию базы данных:

* Database-First;
* Model-First;
* Code-First.

В своей работе я использовал подход Code-First, при данном подходе модель EDMX не используется, и мы вручную настраиваете классы C# объектной модели (данный подход поддерживает как генерацию сущностных классов из существующей базы данных, так и создание базы данных из созданной вручную модели объектов C#).

## **3.4 Проектирование логики сценариев использования**

При проектировании приложения были рассмотрены различные сценарии использования. На рисунке 3.5 представлена диаграмма последовательности добавления блюда в избранное.



Рисунок 3.5 − Диаграмма последовательности добавления блюда в избранное

Из диаграммы видно, что сначала пользователю необходимо нажать на кнопку «Просмотр» под блюдом, которое его интересует. Затем приложение отправит запрос в базу данных на получение информации об этом блюде. Дальше, когда база данных вернет информацию о выбранном блюде, эта информация будет показана пользователю на странице «Просмотр блюда». На этой странице присутствует кнопка добавления в избранное. По нажатию этой кнопки, приложение отправит запрос на добавление этого блюда в соответствующую таблицу. После добавления блюда в избранное, пользователю будет выведено сообщение об успешном добавлении.

## **3.5 Проектирование доступа к сервисам приложения**

Приложение использует принцип Inversion of Control. Inversion of Control (инверсия управления) – это некий абстрактный принцип, набор рекомендаций для написания слабо связанного кода. Суть которого в том, что каждый компонент системы должен быть как можно более изолированным от других, не полагаясь в своей работе на детали конкретной реализации других компонентов.

Для доступа к сервисам навигации в приложении используется IoC-контейнер.

В ViewModel средства навигации будут встраиваться через Dependency Injection(внедрение зависимостей), а именно через паттерн внедрения зависимости через конструктор ([Constructor Injection](http://sergeyteplyakov.blogspot.com/2012/12/di-constructor-injection.html)).

Внедрение зависимостей (DI, Dependency Injection) – это механизм передачи классу его зависимостей. Суть паттерна [Constructor Injection](http://sergeyteplyakov.blogspot.com/2012/12/di-constructor-injection.html) сводится к тому, что все зависимости, требуемые некоторому классу передаются ему в качестве параметров конструктора, представленных в виде интерфейсов или абстрактных классов.

Для доступа к IoC-контейнерам и отдельным ViewModel используется паттерн Service Locator. Основная идея паттерна Service Locator заключается в том, чтобы иметь объект, который знает, как получить все сервисы, которые, возможно, потребуются. Главной особенностью Service Locator является то, что он не создаёт объекты, а знает, как получить тот или иной объект.

MD5 - это алгоритм, позволяющий получить своего рода "отпечаток" исходной строки. Он спроектирован таким образом, чтобы из получившегося отпечатка нельзя было восстановить исходную строку (т.е. зашифровать можно, а расшифровать нет). Алгоритм широко используется при авторизации.

Также у MD5, как у любой хэш-функции, существует такое понятие как коллизии — это получение одинаковых хэшей для разных исходных строк.

алгоритм, позволяющий находить коллизии на обычном компьютере с любым начальным вектором (A,B,C,D) при помощи метода, названного им «туннелирование».

4 Реализация программного средства

Следующим этапом разработки приложения является непосредственная реализация программного решения в соответствии с уже сформированными требованиями и шаблонами.

4.1 Реализация сущностей

В соответствии с требованиями в качестве хранилища данных программного средства должна быть база данных, поэтому первым шагом в реализации программы является выбор технологии, позволяющей это осуществить. Выбор остановился на ORM технологии Entity Framework. Она предоставляет три подхода по проектированию базы данных. В данном программном решении был использован подход Code-First. При данном подходе модель EDMX не используется. Создание базы данных происходит из созданной вручную модели объектов C#. Созданные модели объектов совпадают с сущностями, которые были сформированы раннее в разделе 3.2.

Диаграмма классов UML для сущностных классов представлена на рисунке 4.1

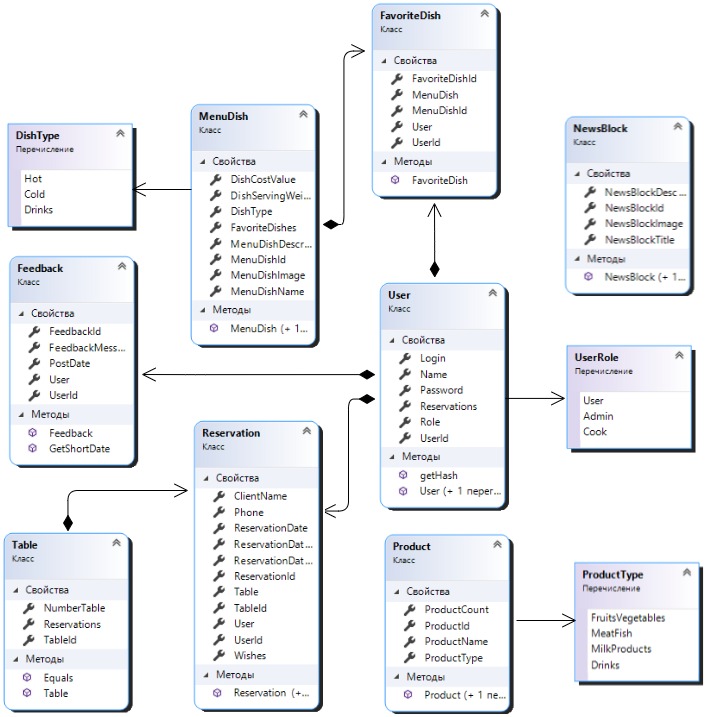


Рисунок 4.1 − Диаграмма классов

На диаграмме классов видно, как связаны между собой сущности. Сущность User связана с классами Feedback, Reservation, FavoriteDish отношением композиции т.е. при уничтожении объекта User в области памяти вместе с ним будут уничтожены и объекты Feedback, Reservation, FavoriteDish связанные с ним. И в этом плане объект User является главным, а объекты Feedback, Reservation, FavoriteDish – зависимыми. Также на диаграмме можно заметить, что классы MenuDish и FavoriteDish, и классы Table и Reservation также связаны отношением композиции.

Из диаграммы видно, что класс User и перечисление UserRole, MenuDish и DishType, а также Product и ProductType связаны между собой отношением ассоциации. Перечисление UserRole показывает роль текущего пользователя: 0 – Пользователь, 1 – Администратор, 2 - Повар. Перечисление DishType показывает тип блюда: 1 - Горячее, 2 – Холодное, 3 - Напитки. А перечисление ProductType показывает тип продукта: 1 – Овощи и фрукты, 2 – Мясо и рыба, 3 – Молочные продукты, 4 – Напитки и другое.

4.2 Реализация архитектуры MVVM

Для ускорения создания и разработки приложений с архитектурой MVVM существует ряд различных библиотек. В качестве такой библиотеки в данном решении был выбран инструментарий MVVM Light Toolkit.

Для реализации паттерна MVVM файлы программы были распределены по соответствующим директориям и реализовали соответствующие функции. Разделение проекта на логические модули представлено на рисунке 4.3.

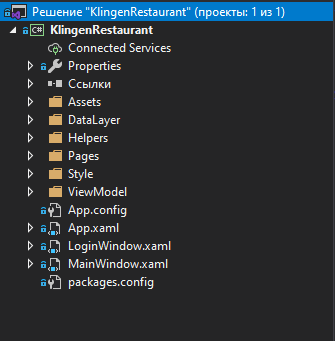


Рисунок 4.3 – Логические модули проекта

В папке Assets находятся картинки, использующиеся в приложении.

В папке DataLayer содержатся сущностные классы, которые используются для создания БД. Их описание представлено в разделе 4.1.

В папке Helpers находятся хранятся различные AttachedProperty, вспомогательные классы перечислений, конвертеры, пользовательские классы валидации, классы для передачи сообщений с помощью библиотеки MVVM Light Toolkit. Также в эту папку вынесены классы для работы с навигацией по страницам и определение для команды.

Навигация по страницам представлена классом FrameNavigationService который реализует интерфейс IFrameNavigationService и INotifyPropertyChanged. Интерфейс INotifyPropertyChanged позволяет уведомить систему об изменениях свойств модели. Интерфейс IFrameNavigationService обязует класс навигации иметь свойство Parameter и реализовать интерфейс INavigationService который находится в пространстве имен GalaSoft.MvvmLight.Views (библиотеки MVVM Light). Интерфейс INavigationService обязует иметь свойство CurrentPageKey которое отвечает за текущую страницу в окне. Метод GetBack – позволяет вернуться назад на 1 страницу. Метод NavigateTo открывает страницу, в окне в котором произошел вызов этого метода. Код класса навигации FrameNavigationService представлен в приложении Б.

В приложении используется паттерн Command который позволяет инкапсулировать запрос на выполнение определенного действия в виде отдельного объекта. В WPF команды представлены интерфейсом ICommand. В приложении он представлен в виде RelayCommand и собственной команды RelayCommandParametr код которой приведен в приложении В. Класс реализует два метода:

* CanExecute: определяет, может ли команда выполняться
* Execute: собственно, выполняет логику команды

А также реализует событие CanExecuteChanged вызывается при изменении условий, указывающий, может ли команда выполняться. Для этого используется событие CommandManager.RequerySuggested.

В папке Styles определены стили для представлений.

В папке ViewModels находятся все классы ViewModel для каждого представления. Также там содержится класс ViewModelLocator код которого предоставлен в приложении Г, а описание предоставлено в следующем подразделе.

В папке Views хранятся все используемые представления.

**4.3 Реализация доступа к сервисам приложения**

В ViewModelLocator реализуется паттерн Service Locator, с помощью статического класса ServiceLocator, который использует в качестве провайдера IoC-контейнер SimpleIoC (библиотеки MVVM Light). В контейнере регистрируются 4 ViewModel: MainViewModel, AdminViewModel, CookViewModel, LoginWindowViewModel. А также регистрируется сервис навигации страниц FrameNavigationService которому передается Uri необходимых page и ключи по которым можно обратиться к ним. Сам класс FrameNavigationService описан в пункте 4.2.

Реализацию паттерна [Constructor Injection](http://sergeyteplyakov.blogspot.com/2012/12/di-constructor-injection.html) можно посмотреть на примере MainViewModel код класса приведён в приложении Д. В классе определенно поле IFrameNavigationService \_navigationService значение которому присваивается через конструктор.

**4.4 Реализация представления**

Для разработки графической части приложения была выбрана технология WPF.

Windows Presentation Foundation (WPF) — это библиотека для создания пользовательских интерфейсов для интеллектуальных клиентских приложений. В основе WPF лежит мощная инфраструктура, основанная на DirectX — API-интерфейсе графики с аппаратным ускорением, который обычно используется в современных компьютерных играх. Это означает, что применение развитых графических эффектов не приведёт к снижению производительности.

Одной из важных особенностей WPF является использование языка декларативной разметки интерфейса XAML, основанного на XML. Разработка с использованием XAML позволяет отделить графический интерфейс от логики приложения, а также создавать насыщенный интерфейс, используя или декларативное объявление интерфейса, или код на управляемых языках C#.

В конечном итоге в приложении было реализовано два окна:

* стартовое окно (регистрация и вход в систему);
* главное окно (содержит различные страницы);

А также пятнадцать страниц:

* страница для авторизации;
* стартовая для регистрации;
* главная страница с новостями;
* страница для просмотра меню;
* страница просмотра блюда;
* страница для резервации столика;
* страница для отображения избранных блюд;
* страница для отзывов;
* страница администратора (для администратора);
* страница просмотра зарезервированных столиков (для администратора);
* страница для добавления блюда в меню (для администратора);
* страница для добавления новости на главную (для администратора);
* страница для повара (для повара);
* страница для заказа продуктов (для повара);
* страница для просмотра склада (для повара).

В результате выполнения данного этапа было создано функционирующее программное средство.

5 Тестирование, проверка работоспособности и анализ полученных результатов

Прежде всего были проведены тесты авторизации: проверка на пустые и неверные данные. Валидация выполнена таким образом, что кнопка для входа или регистрации будет недоступна до тех пор, пока введённые данные не будут корректными. При таком способе валидации шанс ввести невалидные данные минимален. На рисунке 5.1 видно, что валидация на странице регистрации не допускает ввода некорректных полей. Запрещается вводить пробелы в поля имение, логина и пароля. В поле «Имя» можно ввести только буквы латинского и русского алфавитов длинною в 2 символа минимум и 40 символов максимум, и это поле является обязательным, так же как логин и пароль. Логин может содержать только латинские буквы и цифры с специальными символами ‘.’ и ‘\_’. Длина поля от 5 до 40 символов. Пароль должен содержать только латинские буквы, минимум 1 букву и цифру, а также спец символы.

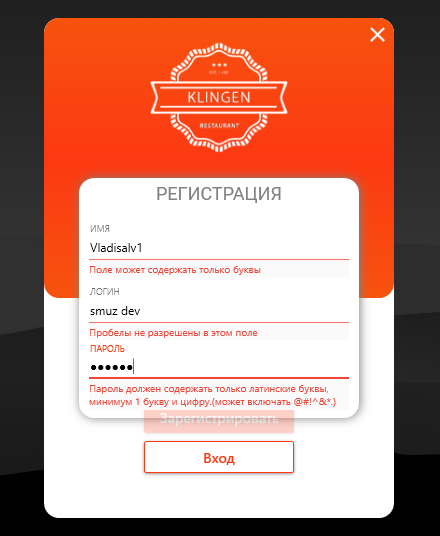


Рисунок 5.1 – Валидация регистрационной формы

На рисунке 5.2 представлена валидация формы входа

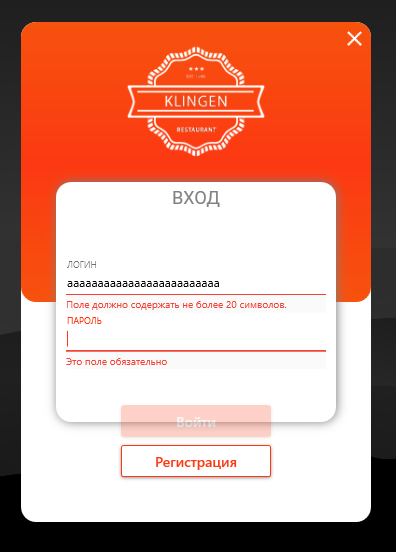


Рисунок 5.2 – Валидация формы входа

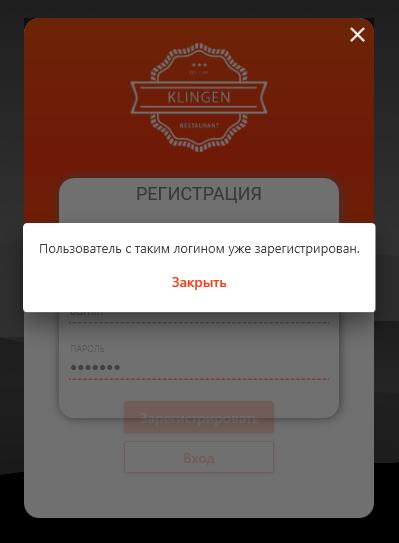


Рисунок 5.3 – Регистрация уже существующего пользователя

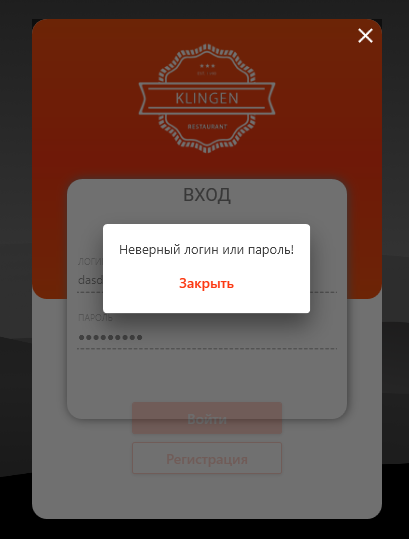


Рисунок 5.4 – Вход с неправильным логином или паролем

Также были проведены тесты на работу валидации внутри приложения. Наличие валидации является обязательным в связи с тем, что в приложении постоянно ведётся работа с базой данных.

Валидация организованна таким же образом, как и на форме авторизации, т.е. кнопки добавить и обновить будут недоступны до тех пор, пока не будут введены валидные данные. Все ошибки, возникшие при валидации, доступны и отображены удобным образом, что позволяет пользователю легко понять какие данные не валидны и быстро исправить их.

Пример результата валидации данных и обработки различных экстремальных ситуаций внутри приложения приведены на рисунках 5.5-5.10.

На странице бронирования присутствуют различные способы валидациии и защиты от ошибок пользователя. Так, например, если пользователь задает некорректную дату (раньше текущей), то автоматически выставляется текущая дата и время. В combobox заносятся лишь те номера столиков, которые свободны в указанное пользователем время. Для этого установлено начальная продолжительность бронирования, равная двум часам. В этот период никто не сможет заказть забронированный столик

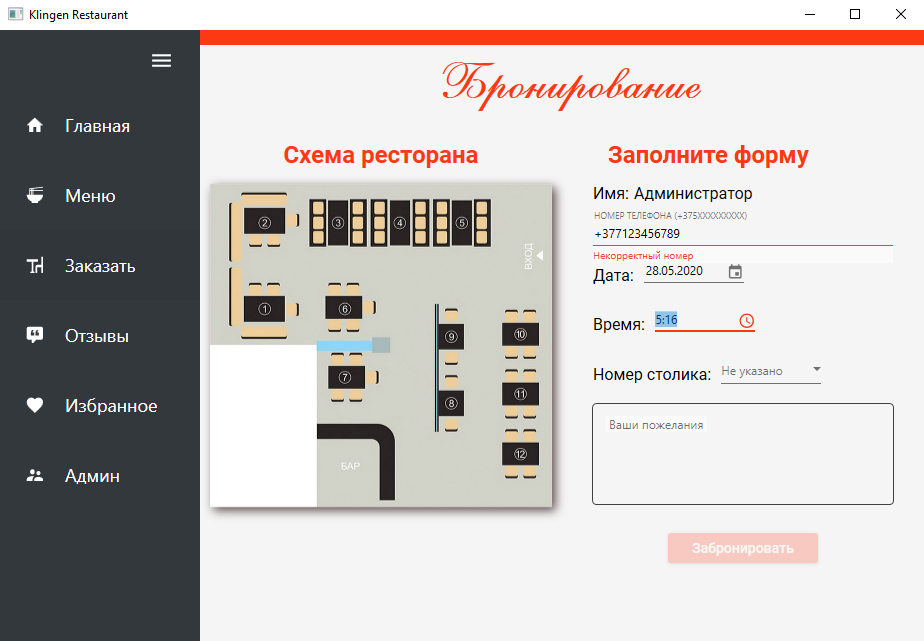


Рисунок 5.5 – Валидация поля ввода номера телефона

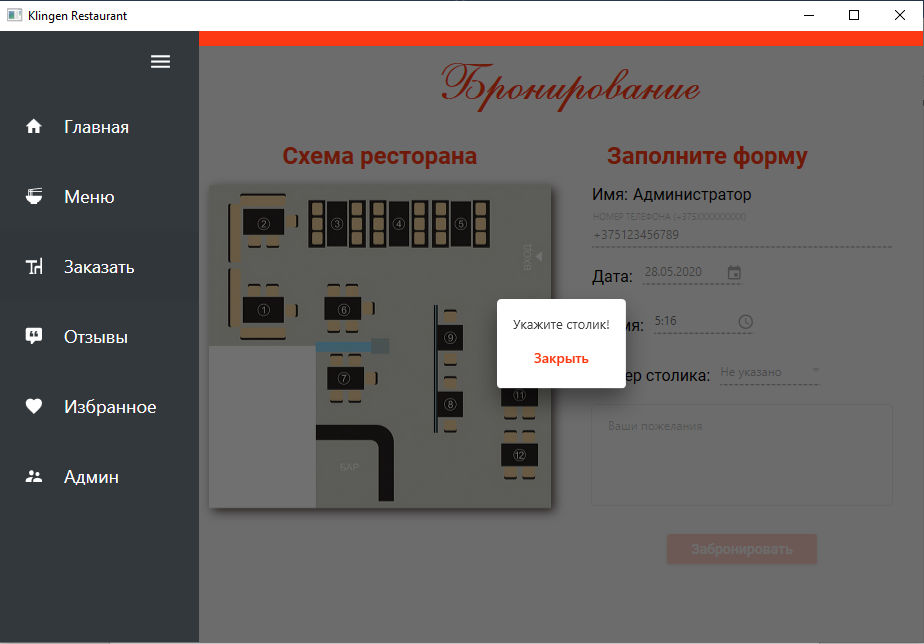


Рисунок 5.6 – Бронирование без указания столика

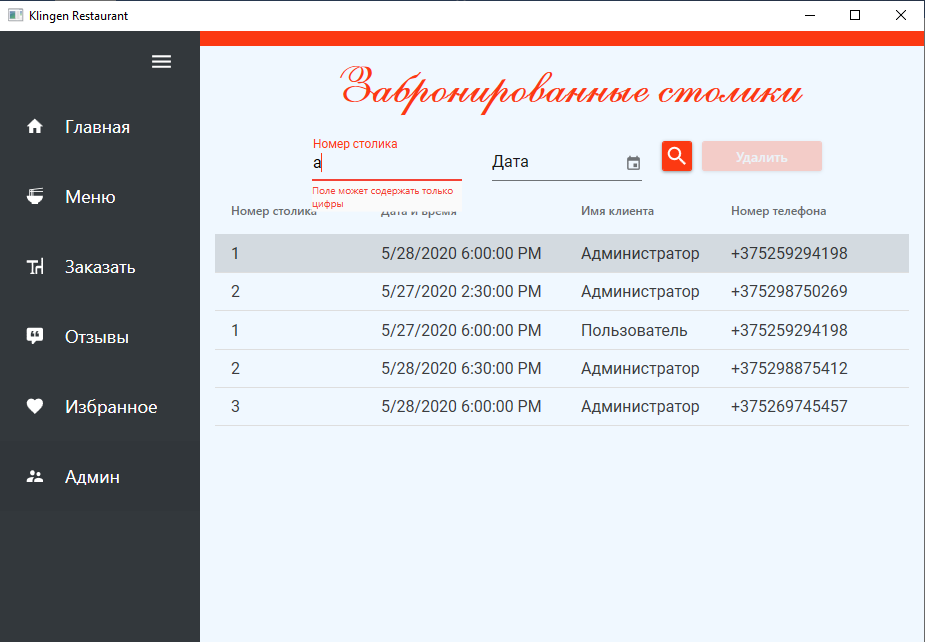


Рисунок 5.7 – Поле ввода по номеру столика

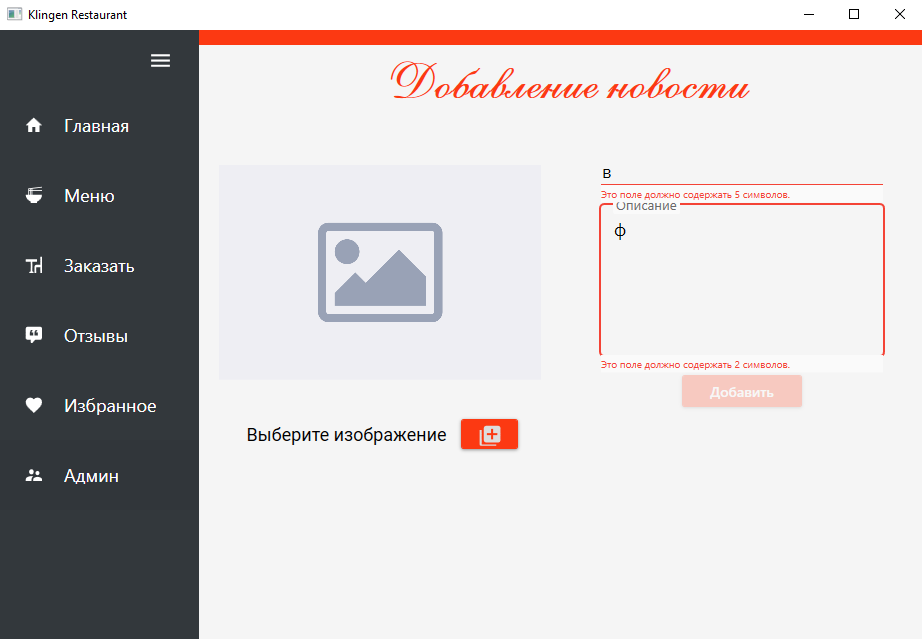


Рисунок 5.7 – Добавление новости

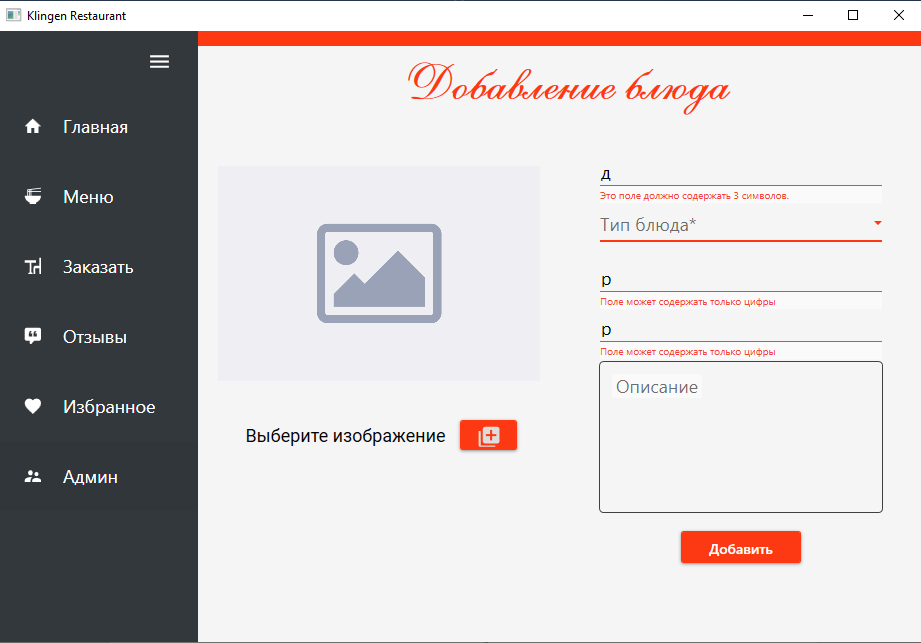


Рисунок 5.8 – Добавление блюда

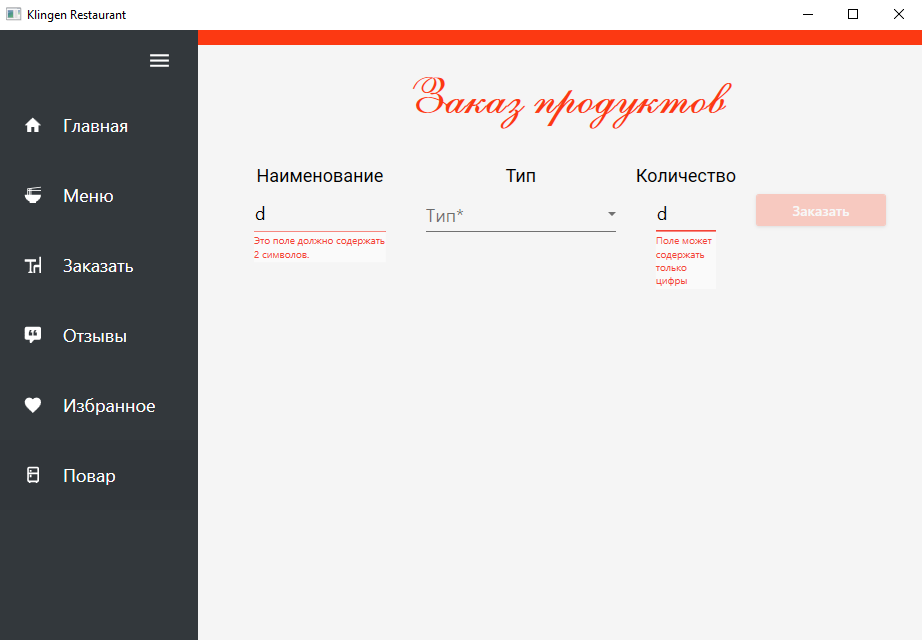


Рисунок 5.9 – Заказ продуктов

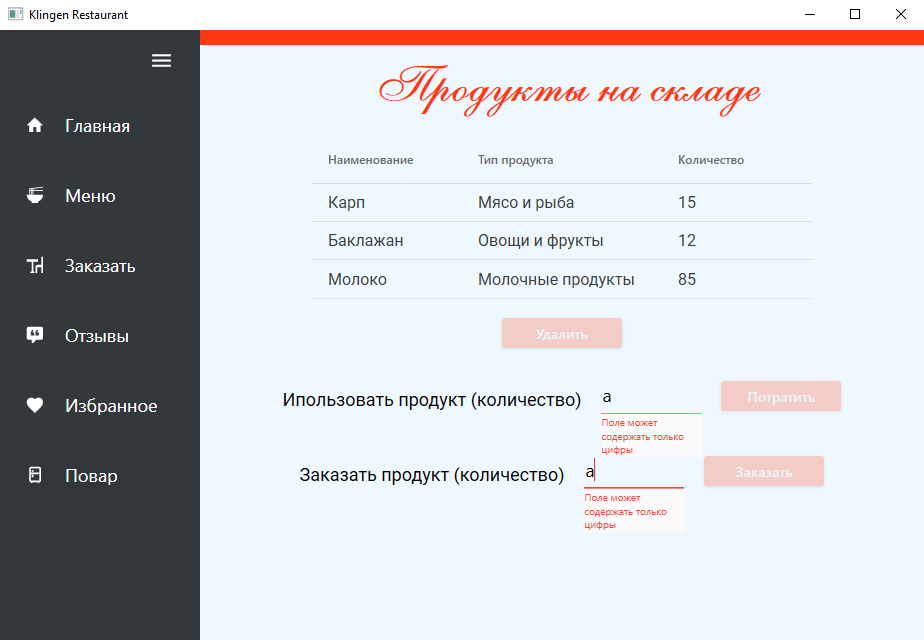


Рисунок 5.10 – Продукты на складе

На данном этапе были выполнены тесты на проверку работоспособности приложения, а именно на проверку валидации. Были проведены анализы результатов, которые показали, что валидация в приложении работает в соответствии с тем, как она задумывалась.

6 Руководство по использованию

После запуска приложения в первую очередь открывается окно входа и регистрации, на которой мы регистрируемся и авторизируемся

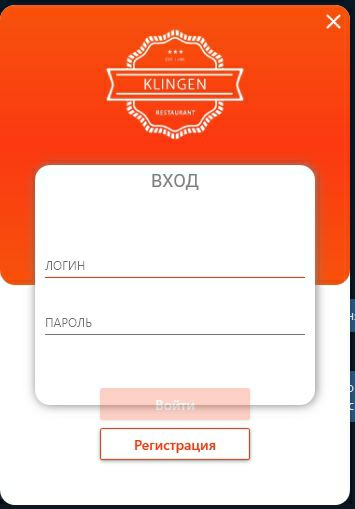


Рисунок 6.1 – Окно входа

После того, как мы вошли в аккаунт, открывается главная страница с последними новостями

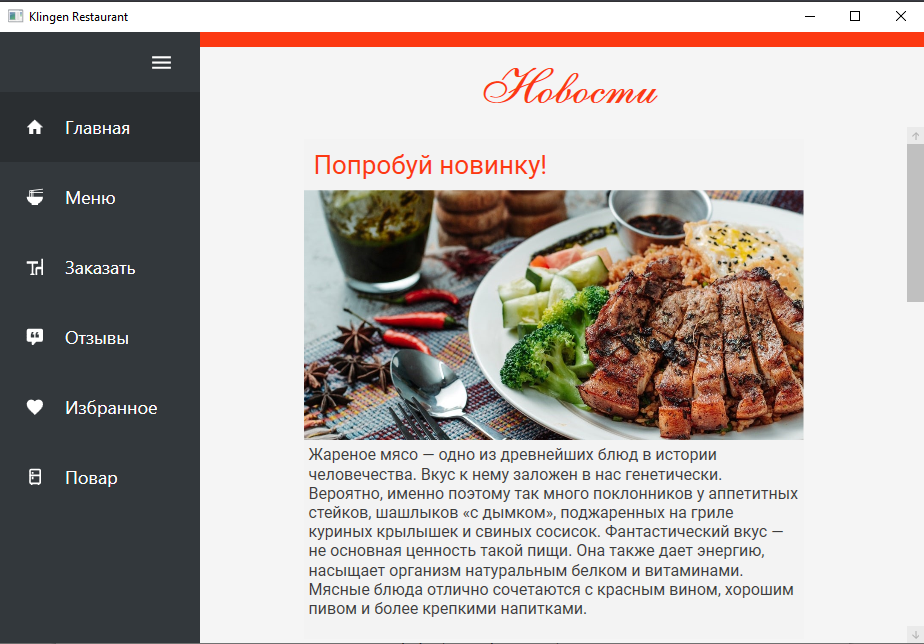


Рисунок 6.2 – Главная страница

На странице меню пользователь может просмотреть пункты меню, воспользоваться поиском, а также узнать более подробную информацию о блюде. Для этого ему нужно нажать на кнопку подробнее и оттуда он перенаправится на страницу просмотра блюда (рис 6.3)

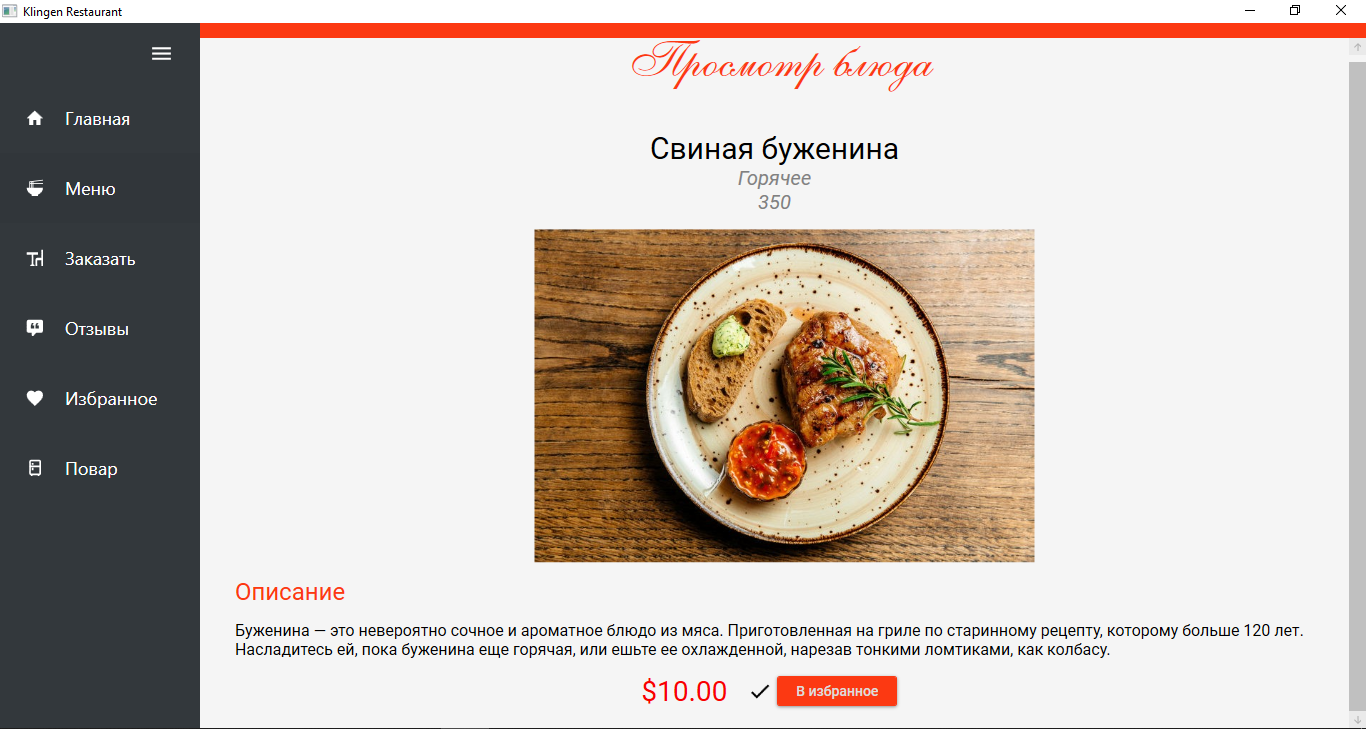


Рисунок 6.3 – Просмотр блюда

Нажав на кнопку избранное, выбранный товар поместиться в отдельную вкладку с избранными блюдами. При повторном нажатии блюдо будет удалено из избранного.

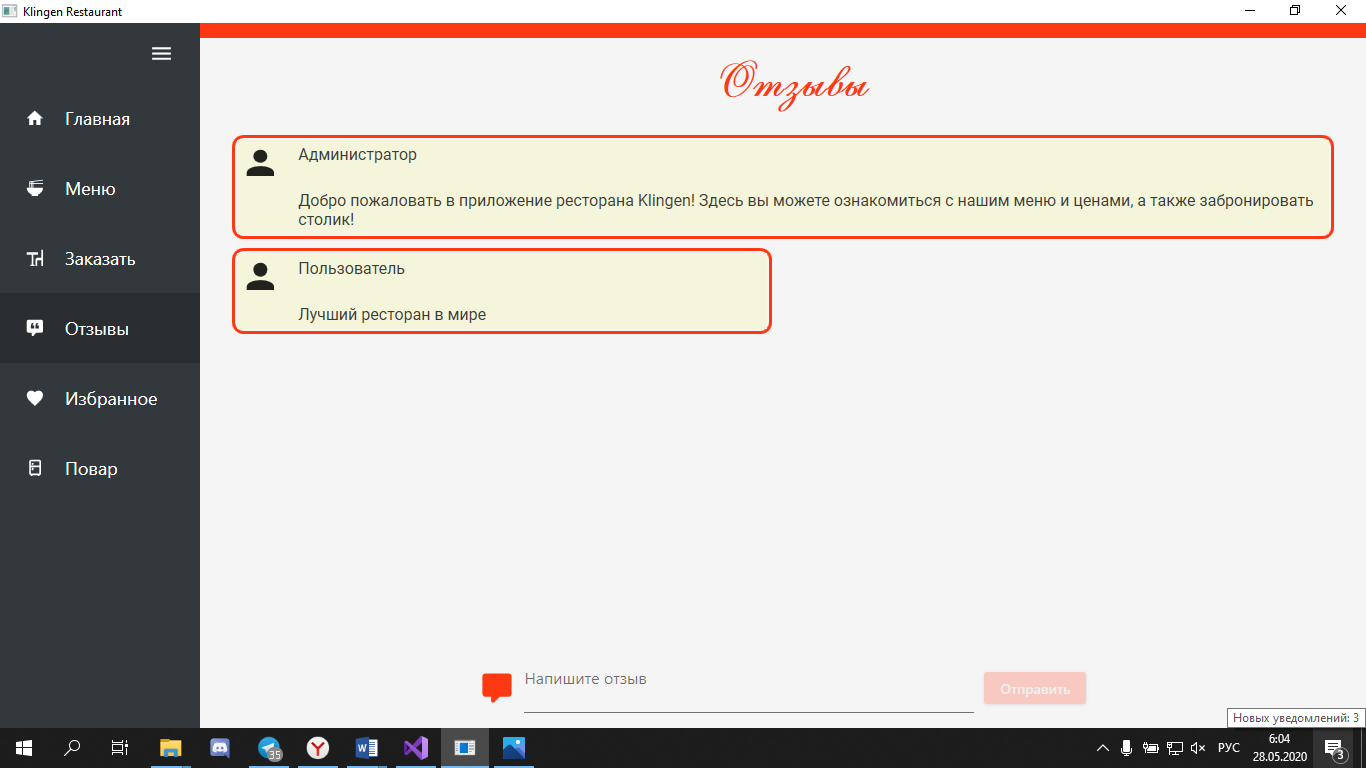


Рисунок 6.4 – Страница отзывов

На странице отзывов пользователь может прочитать имеющиеся отзывы и при желании оставить свой.

На странице бронирования, заполнив форму, пользователь может забронировать столик на указанную дату и время.

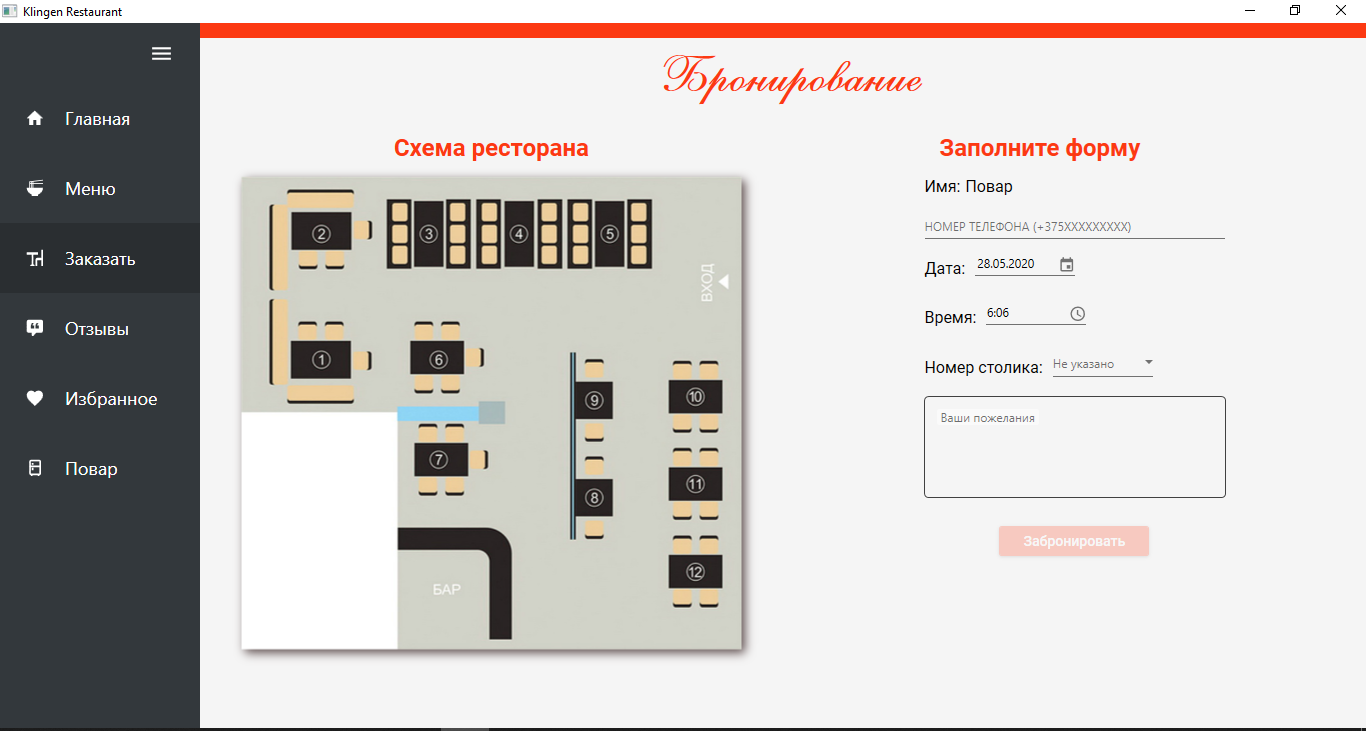


Рисунок 6.5 – Страница бронирования

Для повара создана отдельная страница, на которой он может узнать информацию об имеющихся продуктах на складе, а также использовать их и заказать.

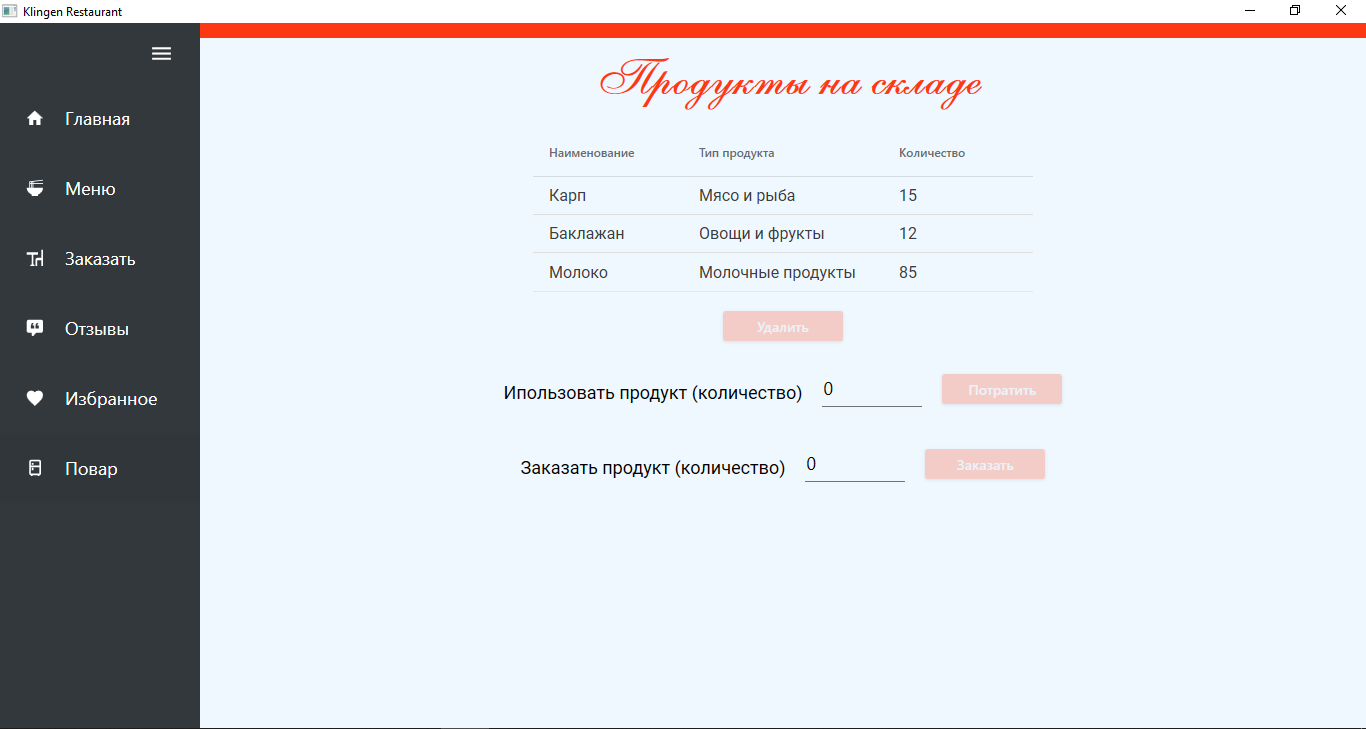


Рисунок 6.6 – Страница просмотра склада

Для администратора также имеется отдельная страница, на которой он может добавлять блюда, новости и просматривать и информацию о забронированных столиках, а также удалять имеющиеся резервации.

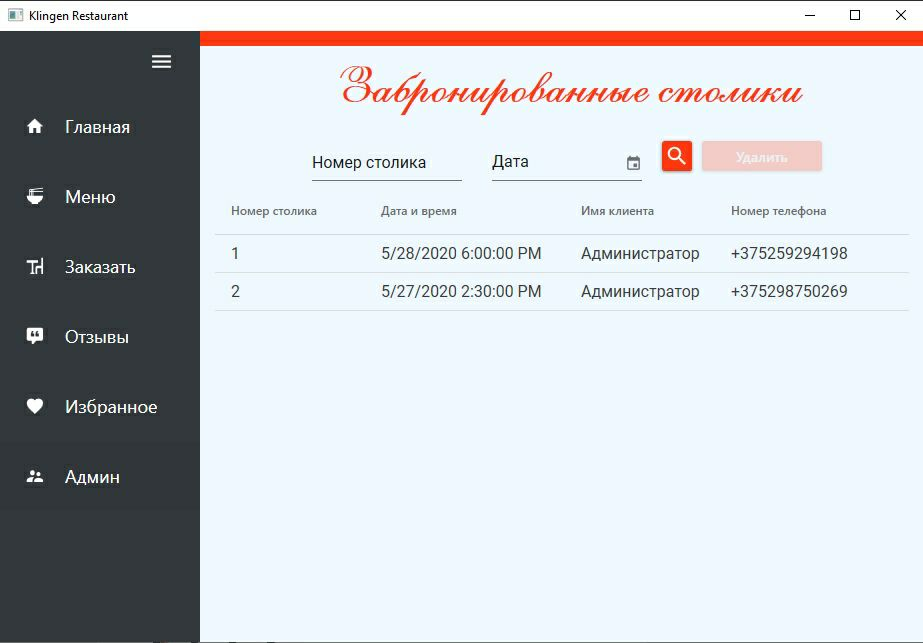


Рисунок 6.7 – Забронированные столики

ЗАКЛЮЧЕНИЕ

В процессе решения поставленной задачи была достигнута поставленная цель по созданию программного средства «Ресторан Klingen». При разработке были выполнены все пункты из указанного списка предполагаемого основного функционала приложения.

В программном средстве были реализованы следующие функции:

* регистрация пользователя в системе;
* вход пользователя в систему;
* просмотр новостей и меню;
* добавление блюд в избранное;
* бронирование столика;
* страница отзывов;
* добавление новостей и блюд;
* просмотр склада;
* заказ и использование продуктов;
* просмотр, поиск и сортировка текущих резерваций.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает, верно, а требования технического задания выполнены в полном объеме.

СПИСОК ЛИТЕРАТУРЫ

1. Пацей, Н.В. Курс лекций по языку программирования С# / Н.В. Пацей. – Минск: БГТУ, 2016. – 175 с.

2. MSDN сеть разработчиков в Microsoft [Электронный ресурс] / Режим доступа: <http://msdn.microsoft.com/library/rus/> . Дата доступа: 20.04.2020

3. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: <https://metanit.com> . Дата доступа: 20.04.2020

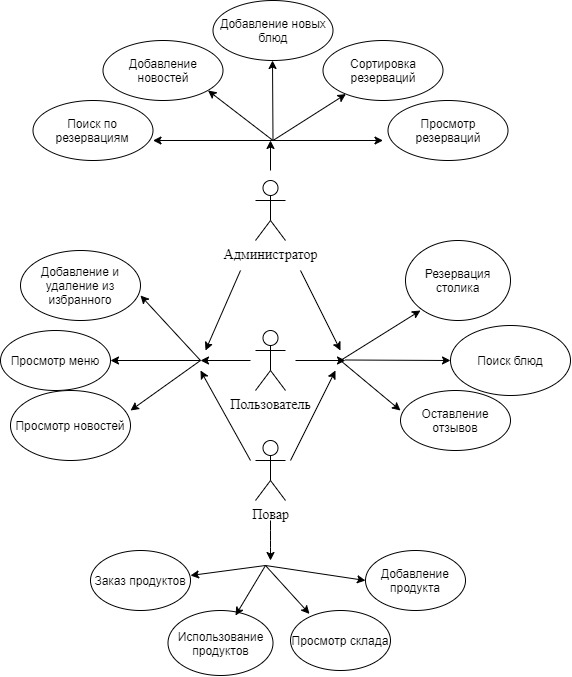
4. ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: <https://professorweb.ru> Дата доступа: 13.04.2020

5 Microsoft Docs Archived Content [Электронный ресурс]/ Режим доступа: <https://docs.microsoft.com/en-us/archive/> Дата доступа: 26.04.2020

6 Форум для программистов или разработчиков [Электронный ресурс] – <https://stackoverflow.com/> – Дата доступа: 5.05.2020

ПРИЛОЖЕНИЕ А

Диаграмма использования



ПРИЛОЖЕНИЕ Б

Листинг 1. Код класса FrameNavigationService

using System.Linq;

using System.Runtime.CompilerServices;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

namespace KlingenRestaurant

{

public class FrameNavigationService : IFrameNavigationService, INotifyPropertyChanged

{

#region Fields

private readonly Dictionary<string, Uri> \_pagesByKey;

private readonly List<string> \_historic;

private string \_currentPageKey;

#endregion

#region Properties

public string CurrentPageKey

{

get

{

return \_currentPageKey;

}

private set

{

if (\_currentPageKey == value)

{

return;

}

\_currentPageKey = value;

OnPropertyChanged("CurrentPageKey");

}

}

public object Parameter { get; private set; }

#endregion

#region Ctors and Methods

public FrameNavigationService()

{

\_pagesByKey = new Dictionary<string, Uri>();

\_historic = new List<string>();

}

public void GoBack()

{

if (\_historic.Count > 1)

{

\_historic.RemoveAt(\_historic.Count - 1);

NavigateTo(\_historic.Last(), null);

}

}

public void NavigateTo(string pageKey)

{

NavigateTo(pageKey, null);

}

public virtual void NavigateTo(string pageKey, object parameter)

{

lock (\_pagesByKey)

{

if (!\_pagesByKey.ContainsKey(pageKey))

{

throw new ArgumentException(string.Format("No such page: {0} ", pageKey), "pageKey");

}

if (Application.Current.Windows.Count > 1)

{

if(!string.IsNullOrWhiteSpace(Application.Current.Windows[1].Name))

Application.Current.Windows[0].Close();

}

Frame frame = GetDescendantFromName(Application.Current.Windows[0], "MainFrame") as Frame;

if (frame != null)

{

frame.Source = \_pagesByKey[pageKey];

}

Parameter = parameter;

\_historic.Add(pageKey);

CurrentPageKey = pageKey;

}

}

public void Configure(string key, Uri pageType)

{

lock (\_pagesByKey)

{

if (\_pagesByKey.ContainsKey(key))

{

\_pagesByKey[key] = pageType;

}

else

{

\_pagesByKey.Add(key, pageType);

}

}

}

private static FrameworkElement GetDescendantFromName(DependencyObject parent, string name)

{

var count = VisualTreeHelper.GetChildrenCount(parent);

if (count < 1)

{

return null;

}

for (var i = 0; i < count; i++)

{

var frameworkElement = VisualTreeHelper.GetChild(parent, i) as FrameworkElement;

if (frameworkElement != null)

{

if (frameworkElement.Name == name)

{

return frameworkElement;

}

frameworkElement = GetDescendantFromName(frameworkElement, name);

if (frameworkElement != null)

{

return frameworkElement;

}

}

}

return null;

}

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null)

{

PropertyChangedEventHandler handler = PropertyChanged;

if (handler != null) handler(this, new PropertyChangedEventArgs(propertyName));

}

#endregion

}

}

ПРИЛОЖЕНИЕ В

Листинг 2. Код класса RelayCommandParametr

public class RelayCommandParametr : ICommand

{

private Action<object> execute;

private Func<object, bool> canExecute;

public event EventHandler CanExecuteChanged

{

add { CommandManager.RequerySuggested += value; }

remove { CommandManager.RequerySuggested -= value; }

}

public RelayCommandParametr(Action<object> execute, Func<object, bool> canExecute = null)

{

this.execute = execute;

this.canExecute = canExecute;

}

public bool CanExecute(object parameter)

{

return this.canExecute == null || this.canExecute(parameter);

}

public void Execute(object parameter)

{

this.execute(parameter);

}

}

ПРИЛОЖЕНИЕ Г

Листинг 3. Код класса ViewModelLocator

using CommonServiceLocator;

using GalaSoft.MvvmLight.Ioc;

using System;

namespace KlingenRestaurant

{

public class ViewModelLocator

{

static ViewModelLocator()

{

ServiceLocator.SetLocatorProvider(() => SimpleIoc.Default);

SimpleIoc.Default.Register<LoginWindowViewModel>();

SimpleIoc.Default.Register<MainViewModel>();

SimpleIoc.Default.Register<AdminViewModel>();

SimpleIoc.Default.Register<CookViewModel>();

SetupNavigation();

}

private static void SetupNavigation()

{

var navigationService = new FrameNavigationService();

navigationService.Configure("Login", new Uri("../Pages/Login/LoginPage.xaml", UriKind.Relative));

navigationService.Configure("Registration", new Uri("../Pages/Login/RegistrationPage.xaml", UriKind.Relative));

navigationService.Configure("Home", new Uri("../Pages/Menu/HomePage.xaml", UriKind.Relative));

navigationService.Configure("Menu", new Uri("../Pages/Menu/MenuPage.xaml", UriKind.Relative));

navigationService.Configure("AboutDish", new Uri("../Pages/Menu/AboutDishPage.xaml", UriKind.Relative));

navigationService.Configure("Reservation", new Uri("../Pages/Menu/ReservationPage.xaml", UriKind.Relative));

navigationService.Configure("Feedback", new Uri("../Pages/Menu/FeedbackPage.xaml", UriKind.Relative));

navigationService.Configure("Favourites", new Uri("../Pages/Menu/FavouritesPage.xaml", UriKind.Relative));

navigationService.Configure("Admin", new Uri("../Pages/Menu/AdminPage.xaml", UriKind.Relative));

navigationService.Configure("Cook", new Uri("../Pages/Menu/CookPage.xaml", UriKind.Relative));

navigationService.Configure("AddMenuDish", new Uri("../Pages/Admin/AddMenuDishPage.xaml", UriKind.Relative));

navigationService.Configure("AddNewsBlock", new Uri("../Pages/Admin/AddNewsBlockPage.xaml", UriKind.Relative));

navigationService.Configure("ReservedTables", new Uri("../Pages/Admin/ReservedTablesPage.xaml", UriKind.Relative));

navigationService.Configure("OrderingProduct", new Uri("../Pages/Cook/OrderingProductPage.xaml", UriKind.Relative));

navigationService.Configure("Warehouse", new Uri("../Pages/Cook/WarehousePage.xaml", UriKind.Relative));

SimpleIoc.Default.Register<IFrameNavigationService>(() => navigationService);

}

public MainViewModel Main

{

get

{

return ServiceLocator.Current.GetInstance<MainViewModel>();

}

}

public LoginWindowViewModel LoginWindowViewModel

{

get

{

return ServiceLocator.Current.GetInstance<LoginWindowViewModel>();

}

}

public AdminViewModel AdminViewModel

{

get

{

return ServiceLocator.Current.GetInstance<AdminViewModel>();

}

}

public CookViewModel CookViewModel

{

get

{

return ServiceLocator.Current.GetInstance<CookViewModel>();

}

}

/// <summary>

/// Cleans up all the resources.

/// </summary>

public static void Cleanup()

{

}

}

}

ПРИЛОЖЕНИЕ Д

Листинг 4. Код класса MainViewModel

using GalaSoft.MvvmLight;

using GalaSoft.MvvmLight.Command;

namespace KlingenRestaurant

{

public class MainViewModel : ViewModelBase

{

#region Private Fields

private User user;

private bool isAdmin;

private bool isCook;

private bool isOpenDialog;

private IFrameNavigationService \_navigationService;

#endregion

#region Public Fields

public User User

{

get

{

return user;

}

set

{

if (user == value)

{

return;

}

user = value;

RaisePropertyChanged();

}

}

public bool IsOpenDialog

{

get

{

return isOpenDialog;

}

set

{

if (isOpenDialog == value)

{

return;

}

isOpenDialog = value;

RaisePropertyChanged();

}

}

public bool IsAdmin

{

get

{

return isAdmin;

}

set

{

if (isAdmin == value)

{

return;

}

isAdmin = value;

RaisePropertyChanged();

}

}

public bool IsCook

{

get

{

return isCook;

}

set

{

if (isCook == value)

{

return;

}

isCook = value;

RaisePropertyChanged();

}

}

public string Message { get; internal set; }

#endregion

#region Commands

private RelayCommand \_loginpageCommand;

public RelayCommand LoginPageCommand

{

get

{

return \_loginpageCommand

?? (\_loginpageCommand = new RelayCommand(

() =>

{

\_navigationService.NavigateTo("Login");

}));

}

}

private RelayCommand \_homeCommand;

public RelayCommand HomeCommand

{

get

{

return \_homeCommand

?? (\_homeCommand = new RelayCommand(

() =>

{

\_navigationService.NavigateTo("Home");

}));

}

}

private RelayCommand \_menuCommand;

public RelayCommand MenuCommand

{

get

{

return \_menuCommand

?? (\_menuCommand = new RelayCommand(

() =>

{

\_navigationService.NavigateTo("Menu");

}));

}

}

private RelayCommand \_reservationCommand;

public RelayCommand ReservationCommand

{

get

{

return \_reservationCommand

?? (\_reservationCommand = new RelayCommand(

() =>

{

\_navigationService.NavigateTo("Reservation");

}));

}

}

private RelayCommand \_feedbackCommand;

public RelayCommand FeedbackCommand

{

get

{

return \_feedbackCommand

?? (\_feedbackCommand = new RelayCommand(

() =>

{

\_navigationService.NavigateTo("Feedback");

}));

}

}

private RelayCommand \_favouritesCommand;

public RelayCommand FavouritesCommand

{

get

{

return \_favouritesCommand

?? (\_favouritesCommand = new RelayCommand(

() =>

{

\_navigationService.NavigateTo("Favourites");

}));

}

}

private RelayCommand \_adminCommand;

public RelayCommand AdminCommand

{

get

{

return \_adminCommand

?? (\_adminCommand = new RelayCommand(

() =>

{

\_navigationService.NavigateTo("Admin");

}));

}

}

private RelayCommand \_cookCommand;

public RelayCommand CookCommand

{

get

{

return \_cookCommand

?? (\_cookCommand = new RelayCommand(

() =>

{

\_navigationService.NavigateTo("Cook");

}));

}

}

private RelayCommand closeDialodCommand;

public RelayCommand CloseDialodCommand

{

get

{

return closeDialodCommand

?? (closeDialodCommand = new RelayCommand(

() =>

{

IsOpenDialog = false;

}));

}

}

#endregion

#region ctor

public MainViewModel(IFrameNavigationService navigationService)

{

\_navigationService = navigationService;

}

#endregion

}

}

**ПРИЛОЖЕНИЕ Е**

Листинг 5. Код класса RestaurantContext

using System.Data.Entity;

namespace KlingenRestaurant

{

public class RestaurantContext : DbContext

{

// Имя будущей базы данных можно указать через

// вызов конструктора базового класса

public RestaurantContext() : base("KlingenRestaurant")

{

}

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

}

// Отражение таблиц базы данных на свойства с типом DbSet

public DbSet<User> Users { get; set; }

public DbSet<MenuDish> Dishes { get; set; }

public DbSet<NewsBlock> News { get; set; }

public DbSet<Reservation> Reservations { get; set; }

public DbSet<Table> Tables { get; set; }

public DbSet<Product> Products { get; set; }

public DbSet<Feedback> Feedbacks { get; set; }

public DbSet<FavoriteDish> FavoriteDishes { get; set; }

}

}