**МИНОБРНАУКИ РОССИИ**

**Федеральное государственное бюджетное образовательное**

**учреждение высшего образования**

**«Чувашский государственный университет имени И.Н. Ульянова»**

**(ФГБОУ ВО «ЧГУ им. И.Н. Ульянова»)**

**Факультет информатики и вычислительной техники**

**Кафедра вычислительной техники**

ОТЧЕТ

О ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

(ТЕХНОЛОГИЧЕСКОЙ (ПРОЕКТНО-ТЕХНОЛОГИЧЕСКОЙ) ПРАКТИКЕ)

|  |  |  |
| --- | --- | --- |
| Обучающийся 1 курса, направление подготовки «Информатика и вычислительная техника», группа ИВТ-42-23 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ подпись, дата | Тихонов А.В. |
|  |  |
|  | | |
| Руководитель,  ст. преподаватель кафедры  вычислительной техники | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ подпись, дата | Юрьева Е.В. |
|  |  |
|  |  |  |
| Руководитель от профильной организации,  исполнительный директор | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ подпись, дата | Иванов Д.С. |
|  |  |
|  |  |  |
| Заведующий кафедрой  вычислительной техники,  к.п.н., доцент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ подпись, дата | Щипцова А.В. |
|  |  |
|  |  |  |

Чебоксары 2024

СОДЕРЖАНИЕ

[1 Теоретическая часть 5](#_Toc173422244)

[1.1 Назначение программы 5](#_Toc173422245)

[1.2 Функциональные возможности 5](#_Toc173422246)

[1.3 Информация об используемых технологиях 5](#_Toc173422247)

[1.3.1 Средство кроссплатформенной разработки Electron 5](#_Toc173422248)

[1.3.2 JavaScript 6](#_Toc173422249)

[1.3.3 Node.js 7](#_Toc173422250)

[1.3.4 NSIS 7](#_Toc173422251)

[1.3.5 Среда разработки Visual Studio Code 8](#_Toc173422252)

[2 Практическая часть 9](#_Toc173422253)

[2.1 Предметная область 9](#_Toc173422254)

[2.2 Используемые структуры данных 9](#_Toc173422255)

[2.2.1 Массивы 9](#_Toc173422256)

[2.2.2 Объекты 10](#_Toc173422257)

[2.3 Постановка задачи 12](#_Toc173422258)

[2.4 Процесс разработки 12](#_Toc173422259)

[2.4.1 Анализ требований: 12](#_Toc173422260)

[2.4.2 Проектирование: 12](#_Toc173422261)

[2.4.3 Разработка 17](#_Toc173422262)

[3 Описание программы 19](#_Toc173422263)

[3.1 Общие сведения 19](#_Toc173422264)

[3.2 Функциональное назначение 19](#_Toc173422265)

[3.3 Описание логической структуры 20](#_Toc173422266)

[3.4 Используемые технические средства 21](#_Toc173422267)

[3.5 Вызов и загрузка 21](#_Toc173422268)

[3.6 Входные данные 21](#_Toc173422269)

[3.7 Выходные данные 22](#_Toc173422270)

[ЗАКЛЮЧЕНИЕ 23](#_Toc173422271)

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 24

ВВЕДЕНИЕ

Цели практики:

– получение профессиональных знаний и опыта технологической (проектно-технологической) деятельности;

– закрепление, углубление теоретических и практических знаний и навыков, полученных ранее при изучении дисциплин учебного плана.

Задачи практики задачи:

– ознакомление с базой практики (профильной организацией), выпускаемой продукцией, структурой исследовательских, проектно-конструкторских и проектно-технологических подразделений, их ролью, задачами и взаимосвязями с другими подразделениями;

– ознакомление с научной организацией труда в исследовательских, проектно-конструкторских и проектно-технологических подразделениях профильной организации;

– изучение технологий создания программных средств вычислительной техники и автоматизированных систем: Electron - это фреймворк для разработки кроссплатформенных десктопных приложений с использованием HTML, CSS и JavaScript. В двоичный код Electron уже встроены [Chromium](https://www.chromium.org/" \t "_blank) и [Node.js](https://nodejs.org/).

– приобретение и закрепление навыков проектно-технологической работы (технологии проектирования программного обеспечения, программных интерфейсов, баз данных): верстка экранов декстопного приложения, построение и управление свойствами виджетов, работа со стилями, навигацией и архитектурой, установка и применение библиотек для реализации основного функционала, работа с API , внедрение механизмов операций с базой данных: чтение и запись, а также управления базой данных и обработки ошибок;

– ознакомление с технологиями и методами обеспечения и оценки качества разрабатываемого программного обеспечения;

– приобретение навыков разработки и оформления программной документации;

Планируемый результат: приложение для управления базой данных парикмахерской. Оно включает в себя информацию о клиентах, услугах и об истории заказов, предоставляет такие функции как, добавление редактирование, поиск, удаление, информации.

1. Теоретическая часть
   1. Назначение программы

Программа предназначена для автоматизации работы парикмахерской, включая управление данными о клиентах, предоставляемых услугах и истории заказов.

Основные функции приложения включают:

– Ведение базы данных клиентов;

– Управление перечнем предоставляемых услуг;

– Создание заказов;

– Хранение и управление историей заказов;

* 1. Функциональные возможности

**Управление клиентами**: поиск по критериям, создание, редактирование и удаление записей о клиентах.

**Управление услугами**: поиск по критериям, добавление, редактирование и удаление услуг, предоставляемых парикмахерской.

**Создание заказов**: поиск по критериям, оформление новых заказов, выбор клиента и услуги, указание времени и даты заказа.

**История заказов**: просмотр и управление историей заказов, поиск заказов по различным критериям.

* 1. Информация об используемых технологиях
     1. Средство кроссплатформенной разработки Electron

Electron — это фреймворк для создания настольных приложений с использованием веб-технологий (HTML, CSS и JavaScript). Он позволяет разработчикам создавать приложения, которые работают на всех основных платформах (Windows, macOS и Linux), используя единую кодовую базу.

**Преимущества**:

– Возможность использования веб-технологий для разработки настольных приложений.

– Кроссплатформенность.

– Доступ к API Node.js, что позволяет взаимодействовать с файловой системой и операционной системой.

**Основные компоненты**:

– **Main process**: основной процесс, который создает окно приложения и управляет им.

– **Renderer process**: процесс, который рендерит пользовательский интерфейс, также позволяет использовать модули node js на клиентской стороне.

– **Applogic process**: процесс, который осуществляет логику приложения: создание интерфейса, взаимодействия его элементов с пользователем , функции редактирования, создания, поиска, удаления данных.

**Основные принципы работы с Electron:**

– **Создание файла main.js, который реализует создание окна и закрытие, масштабирование, завершение работы окна.**

– **Добавление файла renderer.js, который обеспечивает рендеринг страницы в окне.**

– **Создание файла конфигурации package.json c указанием названия приложения, версии, файлов с кодом программы, дополнительные зависимости, затем платформа на которой будет собран установочный файл.**

– **Вызов команды npm run dist, которая соберет приложение и сохранит дистрибутив в папке dist.**

* + 1. JavaScript

JavaScript — это язык программирования, который используется для создания интерактивных и динамических веб-страниц. В контексте Electron, JavaScript используется как для разработки пользовательского интерфейса (в Renderer и Applogic процессе), так и для создания окна, реализации функций, использующие node js модули (в Main процессе).

**Преимущества**:

– Широкая распространенность и поддержка.

– Большое количество библиотек и фреймворков.

– Возможность использования как на стороне клиента, так и на стороне сервера.

* + 1. Node.js

Node.js — это серверная платформа на основе JavaScript, которая позволяет создавать высокопроизводительные и масштабируемые сетевые приложения. В Electron Node.js используется для выполнения серверной логики и взаимодействия с файловой системой и операционной системой.

**Преимущества**:

– Асинхронная и событийно-ориентированная модель, что позволяет эффективно обрабатывать большое количество запросов.

– Широкий набор встроенных модулей и возможность использования множества внешних библиотек через npm.

* + 1. NSIS

Nullsoft Scriptable Install System (NSIS) — система создания установочных программ для [Microsoft Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows) с открытым исходным кодом, разработанная компанией [Nullsoft](https://ru.wikipedia.org/wiki/Nullsoft" \o "Nullsoft) — автором плеера [Winamp](https://ru.wikipedia.org/wiki/Winamp" \o "Winamp). NSIS был задуман как альтернатива [InstallShield](https://ru.wikipedia.org/wiki/InstallShield" \o "InstallShield), предназначенного для коммерческих продуктов. Позволяет создавать для приложения установочные программы , программы для деинсталляции.

* + 1. Среда разработки Visual Studio Code

Visual Studio Code (VS Code) — это среда разработки, разработанная компанией Microsoft, с помощью фреймворка Electron. Она на данный момент является одним из наиболее популярных инструментов для разработчиков.

VS Code — это бесплатная, с открытым исходным кодом, кроссплатформенная IDE, предназначенная для широкого спектра языков и технологий. Она предоставляет обширный набор функциональных возможностей, расширений и инструментов для удобной и эффективной разработки.

Преимуществами VS Code:

– расширения: VS Code предоставляет множество расширений, помогающие ускорит процессы разработки кода: отлов ошибок, тестирование, готовые шаблоны синтаксических конструкций;

– автодополнение и подсказки: они значительно упрощают процесс написания кода. Они подсказывают наименование объектов библиотек, подробное описание и назначение функций;

1. Практическая часть
   1. Предметная область

Предметная область программы охватывает автоматизацию управления данными парикмахерской, включая учет клиентов, услуг и заказов. В рамках предметной области необходимо обеспечить следующие функции:

1. Ведение детализированной базы данных клиентов, включая признак постоянного клиента и историю визитов;

2. Управление перечнем предоставляемых услуг, их стоимостью и описанием;

3. Создание и редактирование заказов, с возможностью выбора клиента и услуги, а также указания даты выполнения заказа;

4. Ведение истории заказов для каждого клиента и возможность поиска заказов по различным критериям;

* 1. Используемые структуры данных
     1. Массивы

**Определение**: Массив — это упорядоченная коллекция элементов, доступ к которым осуществляется по индексу. Массивы в JavaScript являются динамическими и могут содержать элементы различных типов данных.

**Структура в памяти**: Массивы хранят данные в виде последовательности элементов. В памяти они представляют собой непрерывный блок, что обеспечивает быстрый доступ к элементам по индексу.

**Функции и методы**:

– **Добавление элемента**: push(), unshift();

– **Удаление элемента**: pop(), shift(), splice();

– **Поиск элемента**: indexOf(), find(), includes();

– **Итерация**: forEach(), map(), filter(), reduce();

**Временная сложность**:

– **Доступ по индексу**: O(1) — доступ к элементам массива по индексу выполняется за постоянное время;

– **Добавление/удаление в конец**: O(1) — операция push() или pop() работает за постоянное время;

– **Добавление/удаление в начало или середину**: O(n) — операции unshift(), shift(), и splice() требуют перестановки элементов, что занимает время, пропорциональное количеству элементов;

**Плюсы**:

– Быстрый доступ к элементам по индексу;

– Поддержка встроенных методов для манипуляции данными;

– Динамическое изменение размера;

**Минусы**:

– Неэффективен для частого добавления и удаления элементов в начале или середине массива;

– Отсутствие ключей для идентификации элементов, что затрудняет работу с ассоциированными данными;

**Область использования**:

– Хранение упорядоченных данных, таких как списки клиентов, услуг или заказов;

– Идеален для ситуаций, когда важен порядок элементов и требуется частая итерация;

* + 1. Объекты

**Определение**: Объект — это коллекция пар ключ-значение, где ключи представляют собой строки или символы, а значения могут быть любого типа данных, включая другие объекты и массивы.

**Структура в памяти**: Объекты хранят пары ключ-значение в виде хэш-таблицы, что позволяет быстродействующий доступ по ключу.

**Функции и методы**:

– **Создание**: Литерал объекта {}, конструктор new Object();

– **Добавление/Изменение**: obj[key] = value;

– **Удаление**: delete obj[key];

– **Итерация**: for...in, Object.keys(), Object.values(), Object.entries();

**Временная сложность**:

– **Доступ по ключу**: O(1) — доступ к значениям объекта по ключу выполняется за постоянное время;

– **Добавление/удаление ключа**: O(1) — операции добавления и удаления пар ключ-значение выполняются за постоянное время;

**Плюсы**:

– Эффективен для хранения и управления данными с ассоциированными ключами;

– Удобен для организации данных и создания сложных структур;

**Минусы**:

– Не гарантирован порядок свойств (хотя современные реализации JavaScript сохраняют порядок добавления);

– Требует дополнительных операций для работы с данными (например, преобразование ключей в массивы);

**Область использования**:

– Хранение данных с ассоциированными ключами, таких как записи клиентов и услуг;

– Идеален для организации данных и создания сложных структур данных, таких как объекты клиентов или заказов;

* 1. Постановка задачи

Разработка приложения для управления базой данных парикмахерской требовала создания системы, способной эффективно обрабатывать и хранить информацию о клиентах, услугах, заказах и истории заказов. Основными требованиями к системе были:

– Удобный пользовательский интерфейс.

– Возможность создания, редактирования и удаления записей;

– Быстрый доступ к данным и их обработка;

* 1. Процесс разработки
     1. ****Анализ требований****:

Мною изучено требование технического задания. Был составлен план по созданию основных элементов приложения: главное меню, секции отображающие информацию о клиентах, об услугах, о истории заказов, и элементы для осуществления манипуляций над данными из json файлов. А также логика, которая добавляет интерактивность и осуществляет обращение к базе данных.

* + 1. ****Проектирование****:

Разработана архитектура приложения, включающая Main и Renderer процессы в Electron. В них были описаны функции создания окна, масштабирования, скрытия окна и функции для манипуляции с базой данных .Спроектированы структуры данных для хранения информации о клиентах, услугах и заказах.

Структура json файла клиента:

 {

      "code": "2",

      "F": "Нагиев",

      "I": "Дмитрий",

      "O": "Владимирович",

      "birthDate": "1967-04-04",

      "gender": "1",

      "orders": 29

    }

Структура json файла услуги:

 {

      "code": "1",

      "name": "Андеркат",

      "gender": "1",

      "cost": "300"

    }

Структура json файла заказа:

 {

      "code": "2",

      "clientCode": "2",

      "haircutCode": "1",

      "date": "2024-04-24",

      "cost": "500",

      "dis": "0"

    }

Создан макет пользовательского интерфейса с учетом удобства и функциональности. Далее показаны некоторые элементы интерфейса.

Главное меню:

Здесь расположились следующие разделы. Раздел клиентов открывает секцию с клиентами и с инструментами манипулирования над списком клиентов.

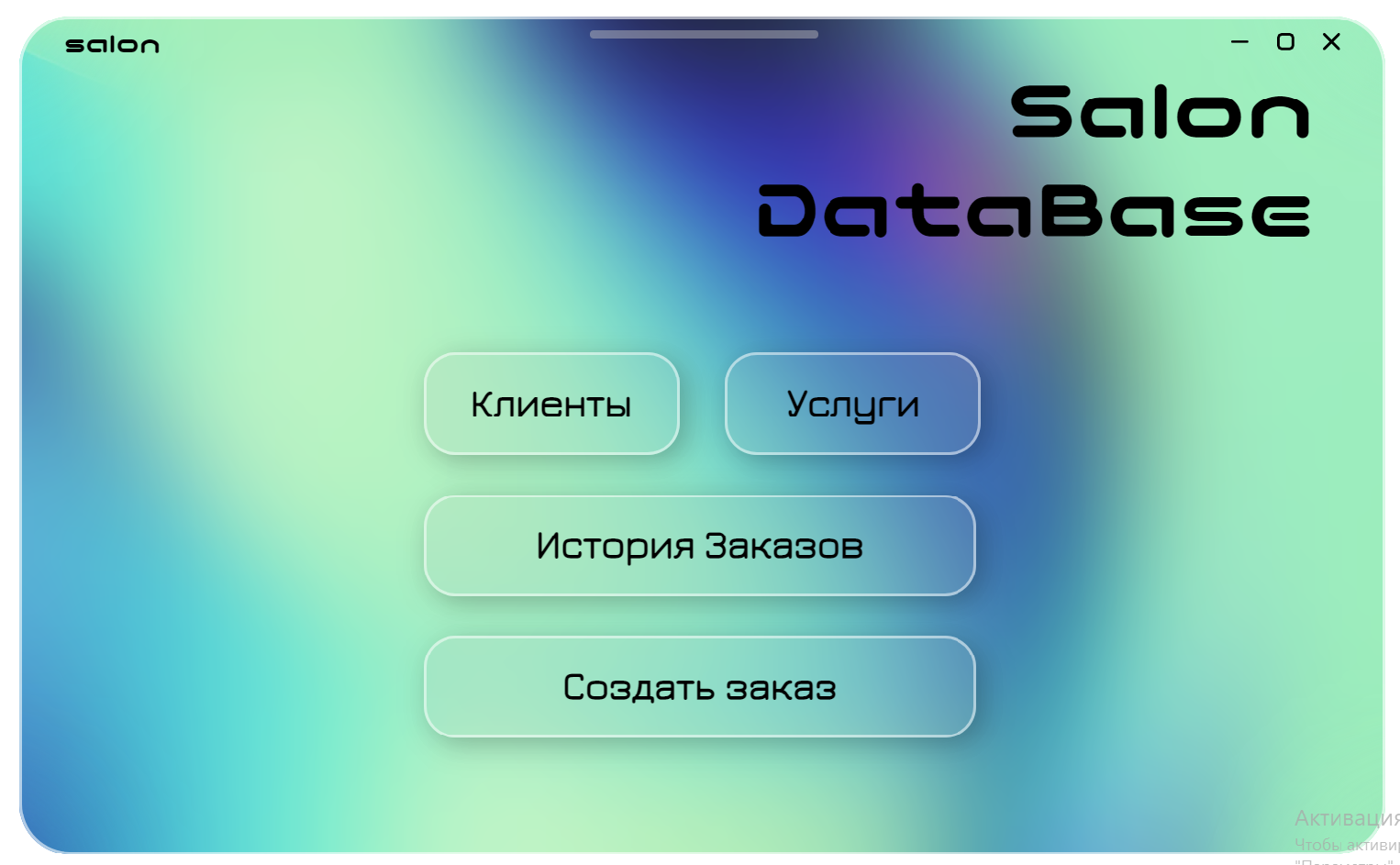


Рисунок 1 – Экран главного меню

Секция клиентов:

Здесь расположились карточки с информацией о каждом клиенте. Добавлены функции поиска, редактирования, добавления и удаления.

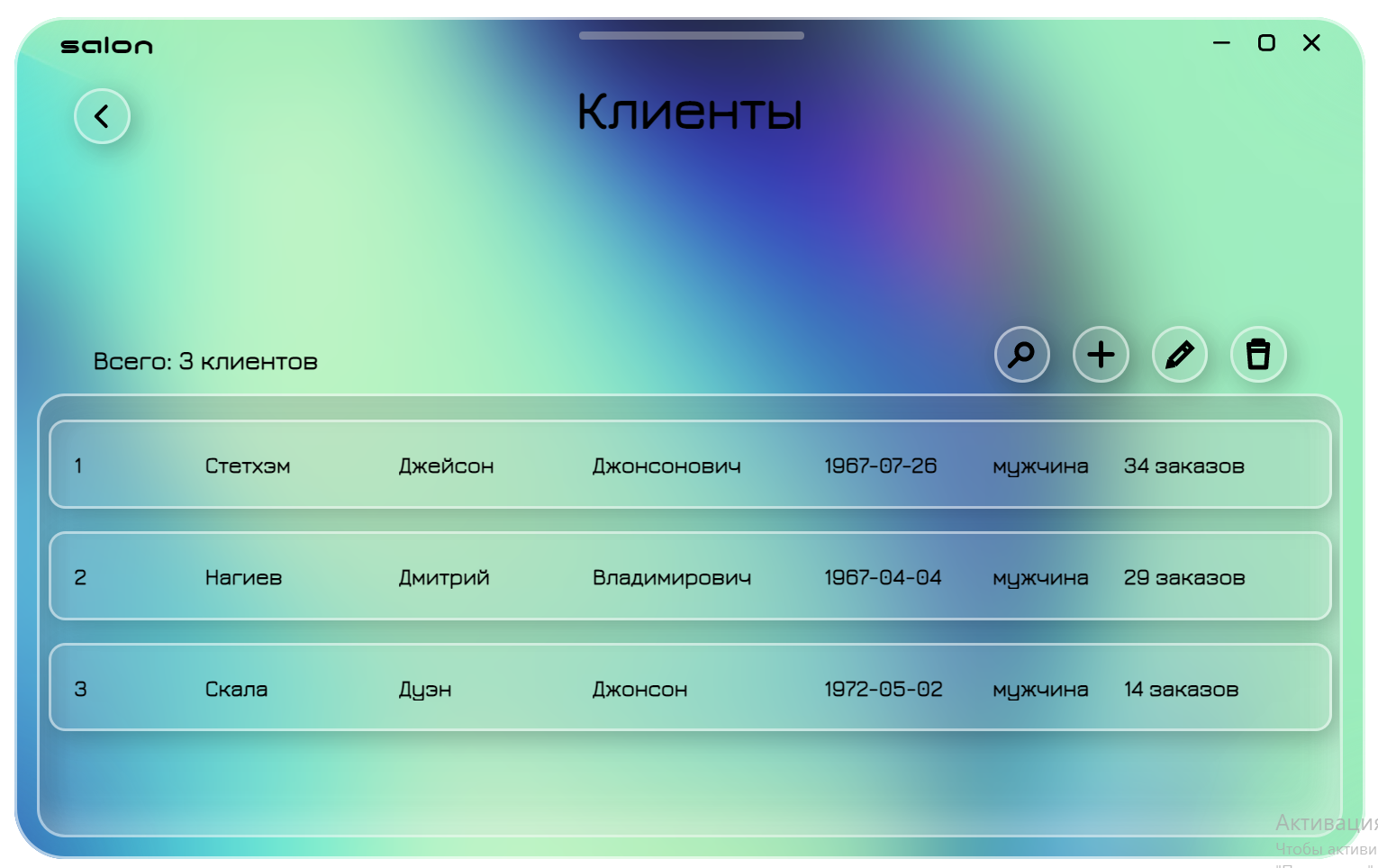


Рисунок 2– Экран секции клиентов

Меню редактирования:

Здесь путем выбора из списка происходит копирование данных в поля для редактирования. После редактирования нажимается кнопка в виде карандаша, и происходит замена данных на обновленные сведения.

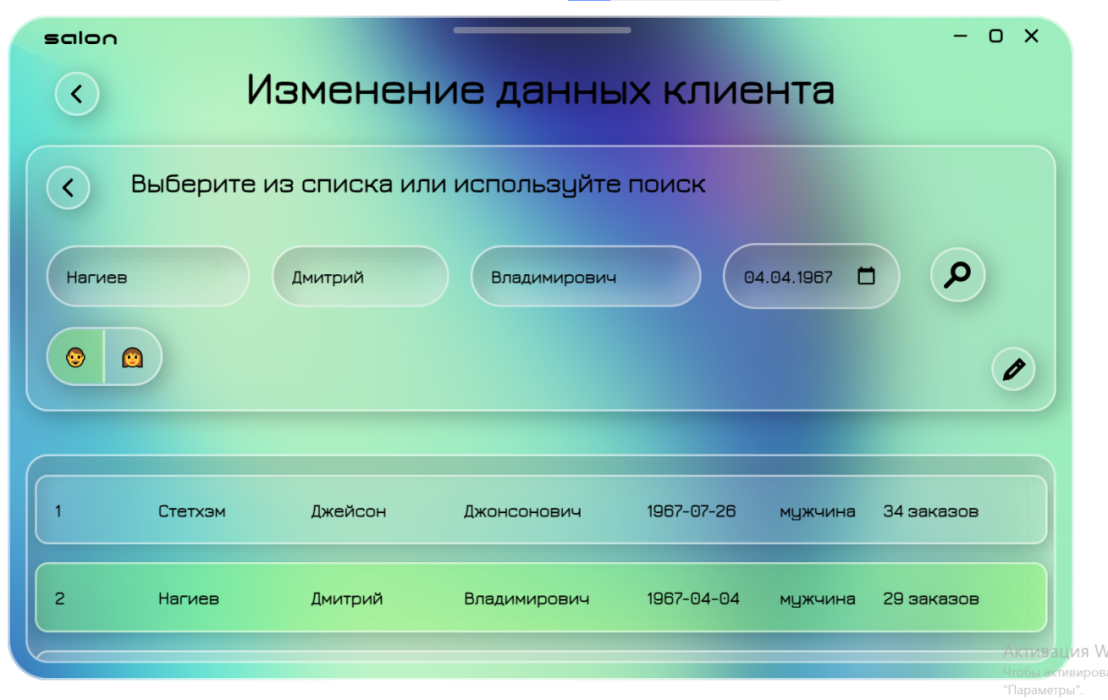


Рисунок 3 – Меню редактирования

Секция история заказов:

Здесь расположены карточки заказов, где написаны: дата совершения заказа, цена заказа, оказанная услуга и данные клиента. В случае того, если информация не помещается в поле, есть возможность наведения на поле для просмотра полной информации.

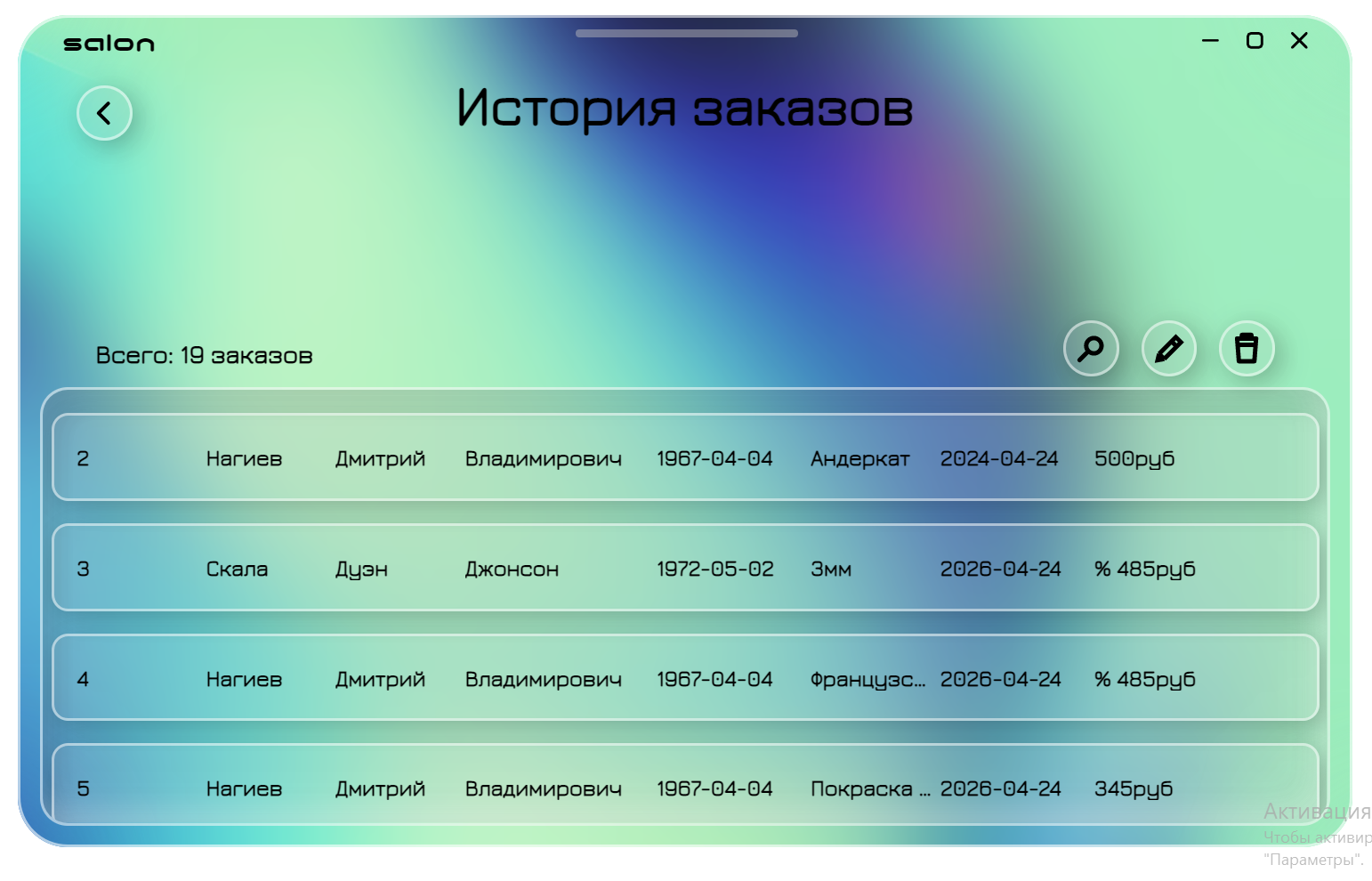


Рисунок 4 – История заказов

Секция создания заказа.

Здесь расположены поля для ввода информации. Вводятся существующий клиент , услуга из базы данных, дата совершения заказа и цена заказа. Если не указать цену заказа, то его цена будет соответствовать цене услуги и будет применена скидка 3% , если клиент постоянный, в противном случае цена будет соответствовать указанной в поле цене.

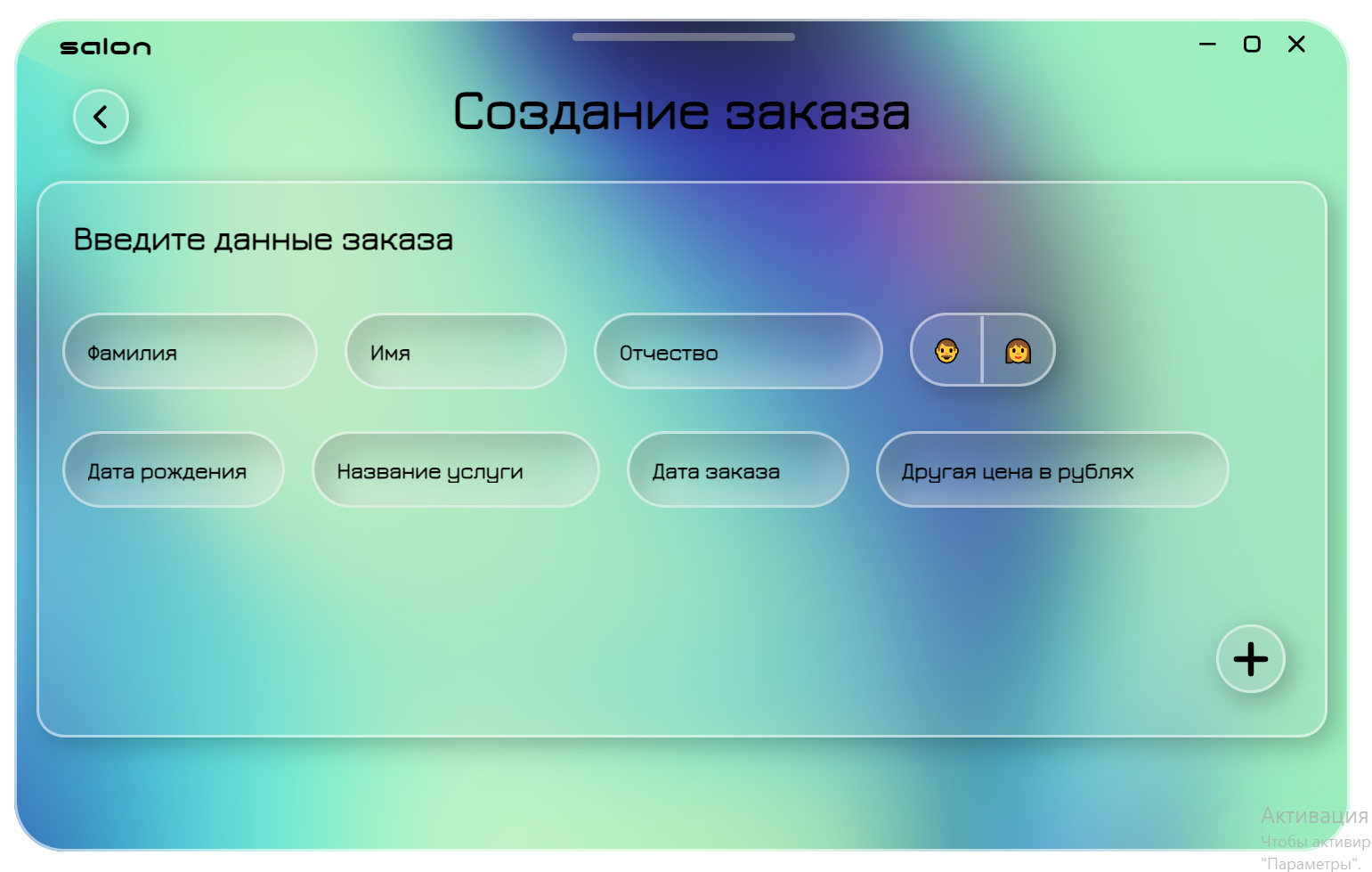


Рисунок 5 – Создание заказа

* + 1. ****Разработка****

**Applogic Process**:

Создан HTML/CSS интерфейс для отображения данных и взаимодействия с пользователем. Реализованы функции для отображения и редактирования данных, создания и удаления записей.

**Main и Renderer Process**:

Реализована логика взаимодействия с файловой системой. Обеспечен обмен данными между пользовательским интерфейсом и базой данных через IPC (межпроцессное взаимодействие).

**База данных**:

Выбрана структура хранения данных (JSON-файлы) и реализованы функции для их чтения и записи.

**Тестирование**:

Проведено тестирование пользовательского интерфейса и функций манипулирования с базой данных. Исправлены ошибки и улучшены функциональные возможности.

**Распространение:**

Создан установочный файл .exe для удобной установки и деинсталляции.

1. Описание программы
   1. Общие сведения

**Обозначение и наименование программы:**

**Название программы:** "Salon".

**Программное обеспечение, необходимое для функционирования программы:**

**Операционные системы:** Windows 7 и выше.

**Языки программирования, на которых написана программа:**

**Язык программирования** JavaScript.

Языки разметки и стилей HTML/CSS.

* 1. Функциональное назначение

**Классы решаемых задач и/или назначение программы:**

**Назначение:** Программа предназначена для управления базой данных парикмахерской, включая хранение и обработку информации о клиентах, услугах, истории заказов, а также для создания и редактирования заказов.

**Функциональные возможности:**

– Ввод и хранение данных о клиентах;

– Управление перечнем услуг;

– Создание и редактирование заказов;

– Просмотр и обновление истории заказов;

**Сведения об функциональных ограничениях на применение:**

**Ограничения:** Программа предназначена для использования в парикмахерских и салонах красоты малого и среднего бизнеса. Не имеет возможности удаленного доступа к данным с других устройств. Не поддерживает интеграцию с POS-системами и другими специализированными бизнес-приложениями.

* 1. Описание логической структуры

**Алгоритм программы:**

**Общий алгоритм:** При запуске программа загружает пользовательский интерфейс. Пользователи могут взаимодействовать с интерфейсом для добавления и редактирования информации о клиентах, услугах и заказах. Программа записывает и читает данные из JSON файлов, обновляет информацию и отображает результаты.

**Используемые методы:**

**Методы:**

– Использование JSON для хранения данных;

– Использование Electron.js для создания настольного приложения;

– Обработка событий в реальном времени и обновление UI с помощью JavaScript;

**Структура программы с описанием функций составных частей и связей между ними:**

– **Главный процесс:** создает диалоговое окно и функции для работы с json;

– **Рендер-процессы:** Отвечают за отображение пользовательского интерфейса и взаимодействие с пользователем;

**Функции:**

– Модуль клиентов: добавление, редактирование, удаление и поиск клиентов. Хранение данных в clients.json;

– Модуль услуг: управление списком услуг и их ценами. Хранение данных в services.json;

– Модуль заказов: создание, редактирование и просмотр заказов. Хранение данных в orders.json;

**Связи между модулями:**

Модули взаимодействуют через общий формат данных (JSON) и обрабатываются через API интерфейс.

**Связи программы с другими программами:**

**Связи:**

Программа не взаимодействует напрямую с другими программами. Все данные хранятся и обрабатываются локально.

* 1. Используемые технические средства

**Типы ЭВМ и устройств, которые используются при работе программы:**

**Требования к аппаратным средствам:**

Место на диске: 178 МБ свободного места для установки программы.

* 1. Вызов и загрузка

**Способ вызова программы с соответствующего носителя данных:**

**Запуск:** Программа запускается через исполняемый файл salon.exe , который создается при сборке приложения.

**Входные точки в программу:**

**Основная точка входа:** main.js файл, который инициализирует приложение и загружает основной пользовательский интерфейс.

**Сведения об использовании ОЗУ:**

**Потребление памяти:** Программа использует примерно 120-150 МБ оперативной памяти.

**Объем программы:**

**Объем:** Программа занимает около 178 МБ на диске после установки.

* 1. Входные данные

**Характер, организация и предварительная подготовка входных данных:**

**Типы данных:** Информация о клиентах, услугах, заказах.

**Формат данных:** Данные вводятся в форме текстовых полей, списков в пользовательском интерфейсе.

**Формат, описание и способ кодирования входных данных:**

**Формат данных:** Данные хранятся в JSON файлах (clients.json, services.json, orders.json).

**Кодирование:** Данные кодируются в UTF-8 для обеспечения совместимости с международными символами.

* 1. Выходные данные

**Характер, организация и предварительная подготовка выходных данных:**

**Выходные данные:** Отчеты, списки заказов, информация о клиентах и услугах.

**Формат данных:** Выходные данные отображаются в списков в пользовательском интерфейсе.

**Формат, описание и способ кодирования выходных данных:**

**Формат данных:** Выходные данные сохраняются в JSON файлы или отображаются в пользовательском интерфейсе.

**Кодирование:** Вся информация представлена в UTF-8 для обеспечения совместимости.

ЗАКЛЮЧЕНИЕ

В ходе практики был проведен анализ, проектирование, тестирование и отладка.

Был получен опыт разработки приложения, которое взаимодействует с базой данных. Были укреплены знания по Javascript и Node js. Появилось умение собирать установочный файл для electron js приложений.

Программа получилась с современным ux/ui, легковесная и с простой установкой на компьютеры.

Техническое задание было реализовано в полном объеме. Требования к заданию были учтены.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация Node js [Электронный ресурс]. URL: <https://nodejs.org/docs/latest/api/>
2. Документация Electron [Электронный ресурс]. URL: https://www.electronjs.org/ru/docs/latest