Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ТЕХНИЧЕСКИЙ ПРОЕКТ**  
Информационная система для клининговой компании

Киров,

2024

**Содержание**

[1. Концепция предлагаемого решения 3](#_Toc182561317)

[1.1. Пользовательский интерфейс 3](#_Toc182561318)

[1.2. Организация хранения 5](#_Toc182561319)

[2. Бизнес-процессы системы 7](#_Toc182561320)

[3. Структура программы и алгоритмы 11](#_Toc182561321)

[4. Выбранная архитектура системы 12](#_Toc182561322)

# **Концепция предлагаемого решения**

## **Пользовательский интерфейс**

Пользовательский интерфейс главной страницы сайта представлен на рисунке 1.

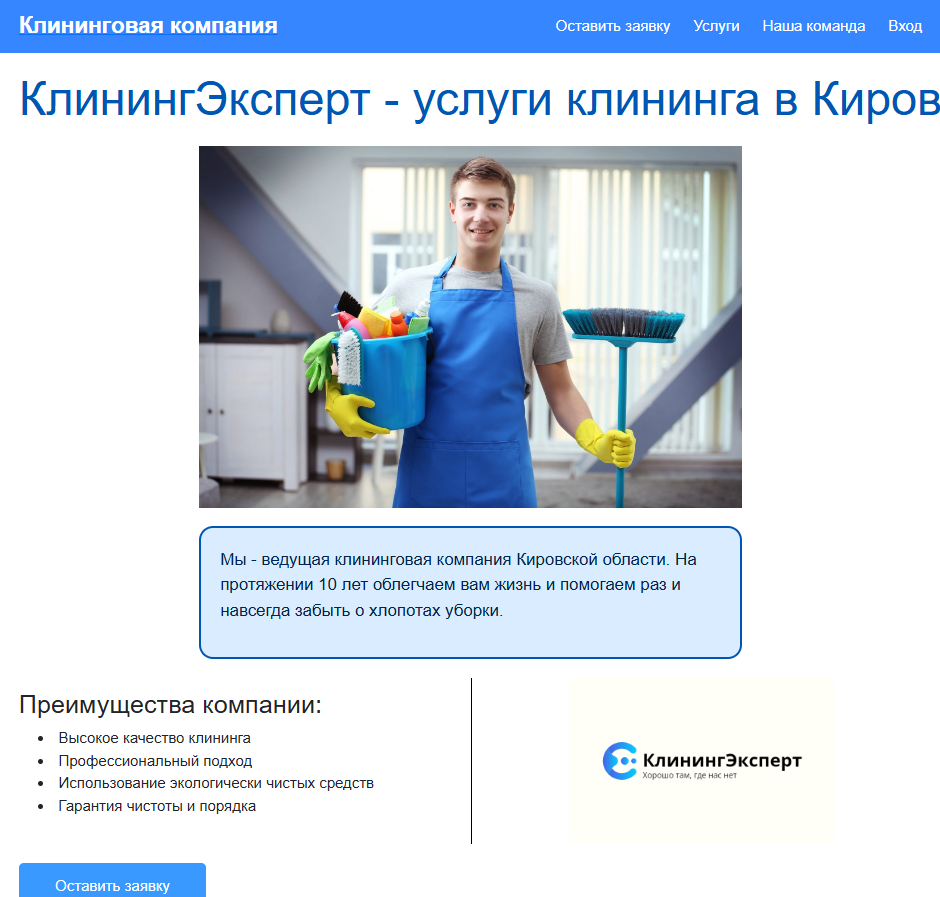


Рисунок 1 – Главная страница сайта

Сайт содержит 3 части: шапка, основная часть и подвал страницы. Главная страница должна содержать заголовок, описание и преимущества компании, логотип, динамическое обновление фотографий и кнопка «Оставить заявку».

Форма заявки представлена на рисунке 2.

Изображение выглядит как текст, электроника, снимок экрана, число

Автоматически созданное описание

Рисунок 2 – Заявка

Форма оформления заявки на уборку включает поля для указания имени, адреса, телефона, даты и времени уборки, а также площади объекта в квадратных метрах. Заказчик может выбрать предложенные услуги.

## **Организация хранения**

Для организации хранения данных используется СУБД MySQL.

На рисунке 2 представлена физическая ER-диаграмма базы данных.

Изображение выглядит как текст, снимок экрана, диаграмма, Параллельный

Автоматически созданное описание

Рисунок 2 – Физическая ER-диаграмма

Исходя из диаграммы можно выделить 6 сущностей:

* Клиент.
* Уборщик.
* Заказ.
* Услуга.
* Список услуг.
* Бригада.

В таблице 5 выделен набор сущностей с их атрибутами.

Таблица 5 – Набор сущностей и их атрибутов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Сущность** | **Свойство** | **Тип** | **Уникальность** | **Обязательность заполнения** | **Ограничения** |
| Client | ClientId | Int | Да | Да | PK |
| Surname | String |  | Да |  |
| Name | String |  | Да |  |
| MiddleName | String |  |  |  |
| PhoneNumber | String |  | Да |  |
| BirthDate | Date |  | Да |  |
| Cleaner | CleanerId | Int | Да | Да | PK |
| Surname | String |  | Да |  |
| Name | String |  | Да |  |
| MiddleName | String |  |  |  |
| PhoneNumber | String |  | Да |  |
| BirthDate | Date |  | Да |  |
| Order | OrderId | Int | Да | Да | PK |
| DateRegistration | Date |  | Да |  |
| Status | String |  | Да |  |
| Address | String |  | Да |  |
| Date | Date |  | Да |  |
| Time | String |  | Да |  |
| Workload | Int |  | Да |  |
| ClientId | Int |  | Да | FK |
| Service | ServiceId | Int |  | Да | PK |
| Title | String |  | Да |  |
| Description | String |  | Да |  |
| Price | Ushort |  | Да | >0 |
| ServiceList | ServiceId | Int | Да | Да | FK |
| OrderId | Int | Да | Да | FK |
| CleaningCrew | CleanerId | Int | Да | Да | FK |
| OrderId | Int | Да | Да | FK |

Связи между сущностями:

«Клиент» – «Заказ»: один ко многим, у каждого заказа определен конкретный клиент, но один клиент может оформлять несколько заказов.

«Заказ» – «Уборщик»: многие ко многим, конкретный заказ в данный момент времени может быть задействован в нескольких бригадах, но в этих бригадах может быть несколько уборщиков.

«Заказ» – «Услуга»: многие ко многим, конкретный заказ в данный момент времени может содержать несколько услуг.

# **Бизнес-процессы системы**

На рисунке 3 представлена контекстная диаграмма IDF0 для ИС.

Изображение выглядит как текст, диаграмма, снимок экрана, линия

Автоматически созданное описание

Рисунок 3 – Контекстная диаграмма IDF0

Исходя из контекстной диаграммы можно сделать вывод, что входными данными являются информации о заявке, клинерах и услугах, ресурсами является администратор и клиент, методами контроля являются устав компании и законодательство РФ, а выходными данными является обработанная заявка.

На рисунке 4 представлена декомпозиция контекстной диаграммы.

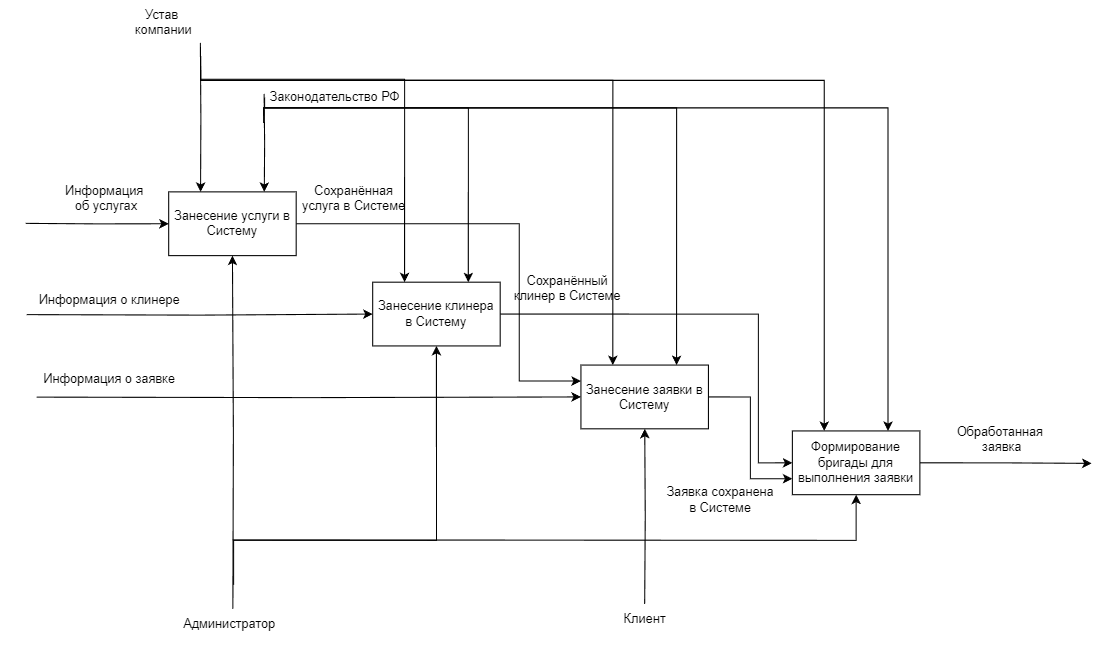


Рисунок 4 – Декомпозиция контекстной диаграммы

Исходя из декомпозиции можно сделать вывод, что для получения обработанной заявке нужно пройти через 4 процесса: Занесение услуги, клинера, заявки в систему и формирование бригады для выполнения заявки.

На рисунке 5 представлена декомпозиция процесса занесения заявки в Систему.

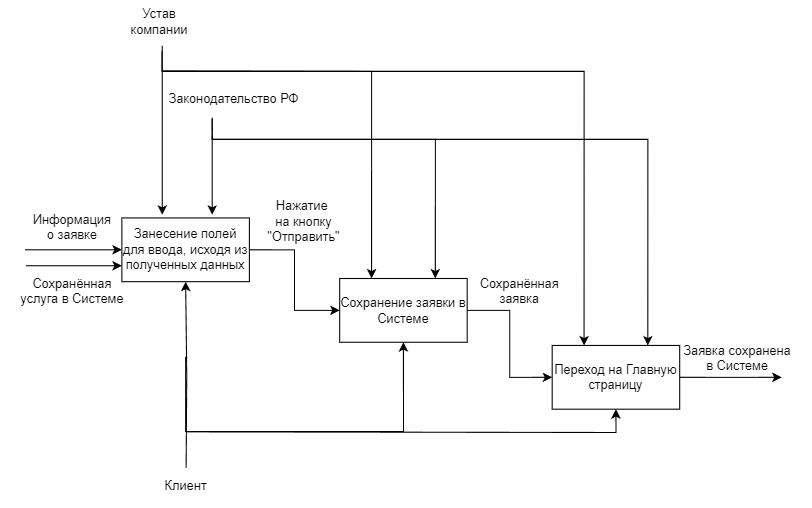


Рисунок 5 – Декомпозиция процесса занесения заявки

На рисунке 6 представлена декомпозиция процесса занесения клинера в систему.

Изображение выглядит как текст, диаграмма, Технический чертеж, План

Автоматически созданное описание

Рисунок 6 – Декомпозиция процесса занесения клинера в систему

На рисунке 7 представлен процесс декомпозиции процесса сохранения услуги в системе.

Изображение выглядит как текст, диаграмма, План, Технический чертеж

Автоматически созданное описание

Рисунок 7 – Декомпозиция процесса сохранения услуги в Системе

На рисунке 8 представлен процесс формирования бригад для выполнения заявки.

Изображение выглядит как диаграмма, текст, План, Технический чертеж

Автоматически созданное описание

Рисунок 8 – Процесс формирования бригад

# **Структура программы и алгоритмы**

* 1. **Структура программы**

Выбранный язык программирования – C# с использованием фреймворка ASP.NET Core MVC. ASP.NET (Active Server Pages для .NET) — платформа разработки веб-приложений, в состав которой входят веб-сервисы, программная инфраструктура и модель программирования. Технология построения пользовательского интерфейса и обработки ввода пользователя – MVC. ASP.NET Core MVC представляет в общем виде построения приложения вокруг трех основных компонентов - Model (модели), View (представления) и Controller (контроллеры), где модели отвечают за работу с данными, контроллеры представляют логику обработки запросов, а представления определяют визуальную составляющую.

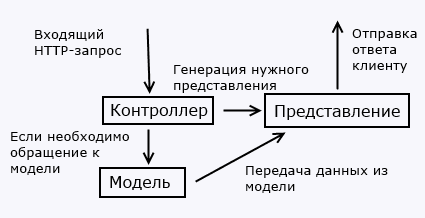


Рисунок 9 – MVC

Преимущества использования ASP.NET Core MVC включают:

Кроссплатформенность: ASP.NET Core может работать на различных операционных системах, таких как Windows, macOS и Linux, что дает возможность разработать и развертывать приложение на любом сервере.

Производительность: ASP.NET Core обеспечивает высокую производительность и низкое время отклика благодаря оптимизированному коду и возможности работы с асинхронными запросами.

Гибкость: Фреймворк позволяет использовать различные практики проектирования и архитектурные паттерны, предоставляя разработчикам свободу в выборе подходов к созданию приложений.

Безопасность: ASP.NET Core предлагает встроенные средства для защиты приложений, такие как атрибуты авторизации, аутентификация и защита от CSRF-атак, что обеспечивает высокий уровень безопасности разрабатываемых веб-приложений.

Поддержка сообщества и экосистемы: Обширное сообщество разработчиков и поддержка множества библиотек и инструментов делают разработку более эффективной и простой.

Совместимость с современными стандартами: ASP.NET Core поддерживает современные стандарты веб-разработки, включая RESTful API, что позволяет легко интегрировать приложение с другими сервисами и приложениями.

Интеграция с современными инструментами разработки: Легкая интеграция с популярными инструментами, такими как Visual Studio, Docker и Azure, упрощает процесс разработки, тестирования и развертывания.

Эти преимущества делают ASP.NET Core MVC мощным инструментом для разработки современных и масштабируемых веб-приложений, поэтому для разработки был выбран именно этот фреймворк.

* 1. **Алгоритмы решения задач**

Форма для отправки заявки от клиента проверяет наличие заполненных полей для ввода, в случае незаполненного поля показывает сообщение о том, что необходимо заполнить все поля. Также такие поля как «дата» и «площадь объекта» должны проходить проверку по следующим условиям: дата> = текущей дате; площадь объекта> 0, если возвращается значение False, значит сайт выводит сообщение о том, что данные поля заполнены некорректными данными, при этом отправка заявки с некорректными данными недопустима.

Форма для входа в личный кабинет клиента/панель администратора проверяет наличие введённого пользователя в базе данных по логину и паролю, в случае несовпадения показывает сообщение, что такой пользователь не найден. В обратном же случае происходит переход в личный кабинет, при помощи API запроса Clients/Login/{id}, где id – это идентификатор найденного пользователя.

Для вывода информации из базы данных используется SQL-запрос SELECT, после чего полученный результат помещается в переменную item и выводится циклом for в таблицу. Редактирование и удаление осуществляется по id записи.

Распределение заявок по клинерам осуществляется выбором id клинера и id заявки, так как связь между таблицами «Клинер» и «Заявки» многие-ко-многим, то на одну заявку могут выезжать бригады клинеров. Добавление одного клинера к одной и той же заявке несколько раз недопустимо.

# **Выбранная архитектура системы**

Для созданного решения была выбрала двухзвенная архитектура «Клиент-сервер» с базой данных. Архитектура MySQL включает в себя две основные компоненты: клиентскую и серверную. Серверная часть представляет собой основу, которая управляет базой данных и обработкой запросов. Клиентская часть выполняет запросы и получает ответы от серверной части.

MySQL работает в режиме клиент-сервер. Клиент отправляет запросы на сервер, который в свою очередь получает эти запросы, обрабатывает их и возвращает результаты. Клиент в данном случае реализован на языке C# с использованием фреймворка ASP.NET Core MVC. ASP.NET.

Архитектура приложения представлена на рисунке 10.

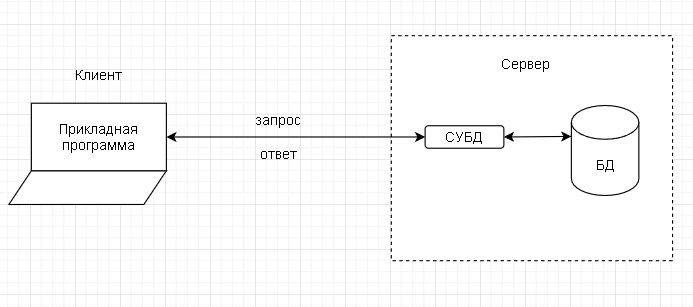


Рисунок 10 – Архитектура приложения