ВВЕДЕНИЕ

Преимущества информационной системы: повышение эффективности и качества, снижение издержек, упрощение управления персоналом, централизованное хранение информации, улучшение планирования и взаимодействия между отделами. Информационная система хранит информацию о сотрудниках, отделах, этапах производства и технике.

Актуальность информационной системы заключается в том, что позволяет эффективно управлять всеми аспектами производства.

Целью работы является упрощение производства и улучшении эффективности производства.

Для достижения поставленной цели необходимо решить следующие задачи:

* изучить предметную область;
* спроектировать базу данных;
* разработать дизайн приложения;
* реализовать функцию авторизации сотрудников;
* реализовать функции для клиентов: просмотр данных производства, редактирование данных сотрудников;
* реализовать функции для начальника цеха: просмотр данных производства, добавление этапа производства, редактирование этапа производства, создание отчёта;
* разработать и протестировать веб-приложение.

1 Техническое задание

1 1 Назначение разработки

Система предназначена для упрощения работы сотрудников ГИБДД, посредством автоматизации процесса регистрации транспортных средств клиентов. Система должна быть обеспечена легким и интуитивно понятным интерфейсом, возможностями безопасной работы с заявлениями для сотрудников, управлениями пользователями системы для администраторов, простым отслеживанием текущих поданных заявлений и историй, легким процессом подачи регистрации транспортных средств для клиентов.

1 2 Требования к программе

Аппаратное обеспечение Сервера:

Сервер:

* Процессор: Intel Xeon E-2334 (или аналогичный AMD EPYC);
* Оперативная память: 32 GB DDR4 ECC REG;
* Накопитель: NVMe SSD 1 TB;
* Сетевая карта: 1 Gigabit Ethernet.

Рабочие станции:

* Процессор: Intel Core i5/i7/i9 (аналоги AMD Ryzen 5/7/9);
* Оперативная память: 8 GB DDR4;
* Накопитель: SSD 256 GB;
* Сетевая карта: Gigabit Ethernet.

Программное обеспечение:

Операционная система:

* Для сервера: Linux (например, Ubuntu Server);
* Рабочие станции: Windows 11 редакции Professional;

Прикладное программное обеспечение для рабочих станций:

* Офисный пакет: Microsoft Office;
* Браузер: Edge, Google Chrome;
* Антивирусное ПО: Kaspersky Anti-Virus.

1 3 Требования к функциональным характеристикам

Система «AutoReg» должна обеспечить выполнение функций:

Администратор:

* Просмотр данных о своем аккаунте;
* Поиск пользователей по определенным критериям;
* Добавление и удаление пользователей.

Сотрудник:

* Просмотр данных о своем аккаунте;
* Поиск заявлений по определенным критериям;
* Изменения статуса заявлений, в зависимости от степени успешности данной заявки.

Клиент:

* Просмотр данных о своем аккаунте;
* Просмотр текущих и архивных заявлений;
* Поиск заявлений текущих и архивных заявлений по определенным критериям;
* Управление своими текущими заявлениями;
* Возможность подачи новой заявки.

1 4 Стадии и этапы разработки

После утверждения технического задания организация-разработчик непосредственно приступает к разработке программного обеспечения.

Таблица 1.1 – Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| № этапа | Название этапа | Срок сдачи | Отчетность |
| 1 | 2 | 3 | 4 |
| 1 | Анализ требований |  | Анализ требований |
|  | Написание технического задания |  | Техническое задание |
|  | Написание описания предметной области |  | Описание предметной области |
|  | Написание входной/выходной информации |  | Входная/Выходная информации |
|  | Написание контрольного примера |  | Контрольный пример |
| 2 | Проектирование ПО |  |  |
|  | Разработка прототипа ПО |  | Прототип ПО |
|  | Разработка UML-диаграммы |  | UML-диаграмма |
|  | Создание алгоритма разрабатываемого ПО |  | Алгоритм разработки ПО |
|  | Создание структуры базы данных |  | Структура базы данных |
|  | Написание контроллеров+HTTP-запросов |  | Контроллеры+HTTP-запросы |
| 1 | 2 | 3 | 4 |
|  | Разработка UML-диаграммы |  | Прототип ПО |
| 3 | Разработка ПО |  |  |
|  | Создание базы данных |  | База данных |
|  | Создание API |  | API |
| 4 | Тестирование ПО, API |  |  |
|  | Написание модульного тестирования |  | Протокол модульного тестирования для веб-сервиса |
|  | Написание функционального тестирования |  | Чек-лист тестирования веб-сервиса |
|  | Юзабилити |  | Юзабилити |
| 5 | Сопровождение ПО |  |  |
|  | Сопровождение руководства пользователя |  | Руководство пользователя |

1 5 Описание входной информации

Входной информацией для системы будут являться данные о пользователях, о транспортных средствах, о заявках, об отделениях ГИБДД.

Информация о пользователях содержит следующие параметры:

* Фамилия и Имя пользователя;
* Паспортные данные;
* Контактные данные;

Информация о транспортных средствах содержит следующие параметры:

* Характеристики транспорта;

Информация о заявках содержит следующие параметры:

* Данные о клиенте;
* Данные о транспорте клиента;
* Информация о месте проведения (отделения);
* Информация о обрабатывающим заявку сотруднике;
* Информация о дате подачи, назначения и закрытия заявки;
* Информация о статусе и результате заявления;

Информация об отделениях ГИБДД содержит следующие параметры:

* Наименование отделения;
* Адрес отделения;
* Время работы;
* Данные о руководителе отделения;
* Номер телефона руководителя;

1 6 Описание выходного документа

Выходным документом будет являться отчет об успешности рассмотренного заявления. Описание документа представлен в таблице 1.2.

Таблица 1.2 – Описание выходного документа.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

Образец выходного документа



2 Разработка прототипа (скрины с фигмы)

Рисунок 2.1 – Окно входа.

Рисунок 2.2 – Окно профиля.

Рисунок 2.3 – Окно управления пользователями.

Рисунок 2.4 – Добавление пользователя

Рисунок 2.5 – Удаление пользователя

Рисунок 2.6 – Окно управления заявлениями

Рисунок 2.7 – Изменение данных в заявлении

Рисунок 2.8 – Страница подачи заявления

Рисунок 2.9 – Страница с текущими заявлениями

Рисунок 2.10 – Страница истории заявлений

3 Структура проекта

Структура проекта представлена на рисунке 3.1.

Рисунок 3.1 – Структура проекта

Таблица 1.3 – Описание модулей и процедур

|  |  |  |
| --- | --- | --- |
| Модули | Процедура | Назначение |
| 1 | 2 | 3 |
| MainWindow.xaml.cs | logButton\_Click | Вход в систему по нажатии на кнопку |
|  | buttonExit\_Click | Выход из системы по нажатии на кнопку |
|  | Window\_MouseLeftButtonDown | Перемещение окна при зажатии ЛКМ |
| AccountWindow.xaml.cs | GoUserControl\_Click | Переход в окно управления пользователями по нажатии на кнопку |
|  | GoApplicationControl\_Click | Переход в окно управления заявлениями по нажатии на кнопку |
|  | navbar\_MouseLeftButtonDown | Перемещение окна при зажатии ЛКМ |
|  | buttonExit\_Click | Выход из системы по нажатии на кнопку |
| UserControlWindow.xaml.cs | GetRoles | Получение списка ролей |
|  | UpdateDB | Обновление таблицы |
|  | buttonAddNewUser\_Click | Добавление нового пользователя по нажатии на кнопку |
|  | buttonChangeUserData\_Click | Изменение данных пользователя по нажатии на кнопку |
|  | buttonDeleteUser\_Click | Удаление пользователя по нажатии на кнопку |
|  | navbar\_MouseLeftButtonDown | Перемещение окно при зажатии ЛКМ |
|  | buttonExit\_Click | Выход из системы |
|  | tableUsers\_SelectionChanged | Изменение выбранной строки в таблице |
|  | buttonUpdateDB\_Click | Обновить таблицу по нажатию кнопки |
|  | buttonSearch\_Click | Поиск строк в таблице по нажатию кнопки |
| ApplicationWindow.xaml.cs | GetStatus | Получение списка статусов |
|  | ClearFields | Очищение полей ввода |
|  | FillFields | Заполнение полей ввода |
|  | UpdateDB | Обновление таблицы |
|  | UpdateSelectedItemBars | Обновление полей ввода для выбранной строки в таблице |
|  | tableApplications\_SelectionChanged | Изменение выбранной строки в таблице |
|  | navbar\_MouseLeftButtonDown | Перемещение окна по зажатии ЛКМ |
|  | buttonExit\_Click | Выход из системы по нажатию на кнопку |
|  | buttonSearchData\_Click | Поиск строк по нажатию на кнопку |
|  | buttonCloseApplication\_Click | Успешное закрытие заявления по нажатие на кнопку |
|  | buttonDeclineApplication\_Click | Отклонение заявление по нажатию на кнопку |
|  | buttonAcceptApplication\_Click | Рассмотрение заявления по нажатию на кнопку |
|  | buttonClearFilteredData\_Click | Стирание строк в таблице на все строки |
|  | buttonUpdateDB\_Click | Обновление таблицы по нажатию на кнопку |

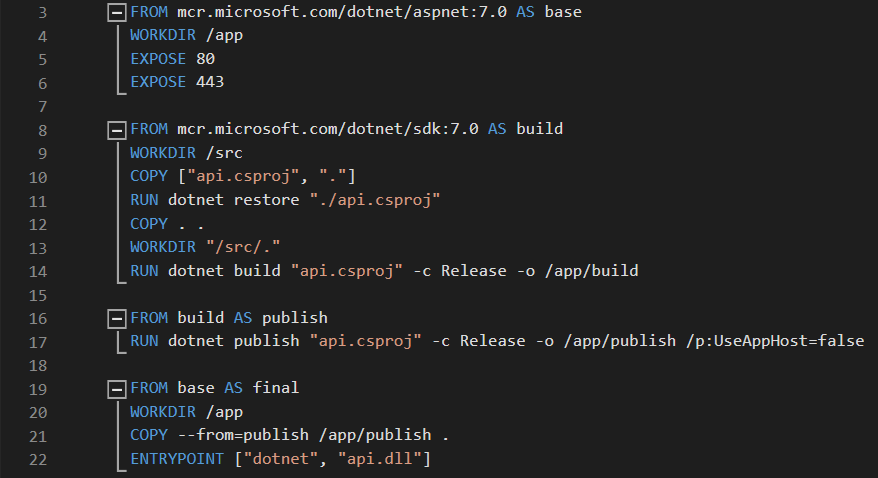
4 Описание запросов API

|  |  |  |
| --- | --- | --- |
| Метод | Response body (возвращаемый результат) | Код результата |
|  |  |  |

|  |  |
| --- | --- |
| Curl (входные данные) | URL (Строка запроса) |
|  |  |

5 Создание Docker контейнера

Для создания Docker контейнера создается специальный файл Dockerfile с кодом, представленным на рисунке 5.1.

 Рисунок 5.1 – Код внутри Dockerfile.

Затем описывается docker-compose.yml как основной файл для создания контейнеров. Код docker-compose.yml представлен в рисунке 5.2.

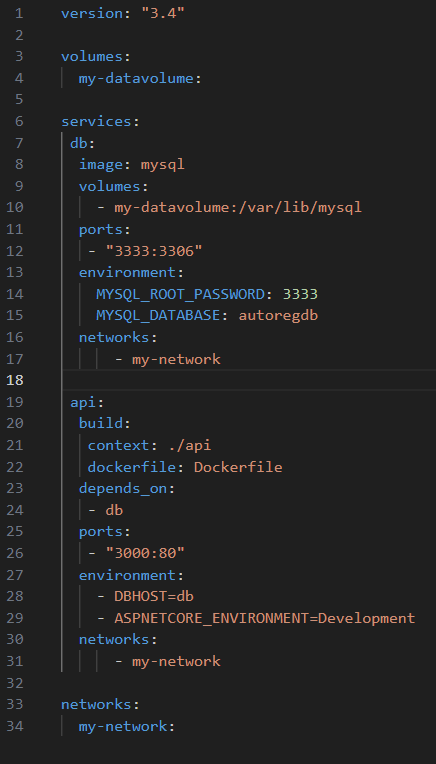
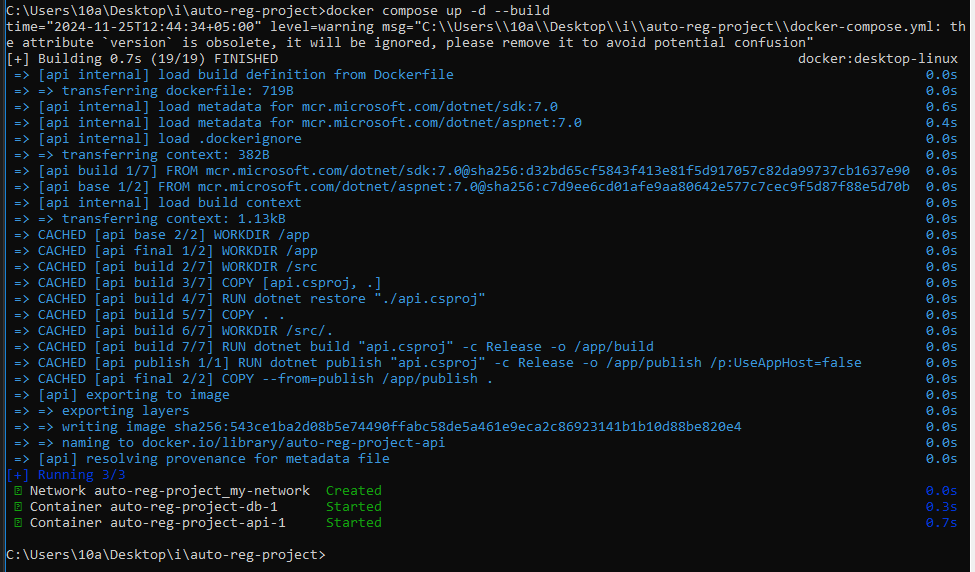


Рисунок 5.2 – Код внутри docker-compose.yml

Далее с помощью команды “docker compose up -d --build” собираем проект используя данные в docker-compose.yml и Dockerfile и запускаем в фоновом режиме. Процесс работы команды представлен на рисунке 5.3.

 Рисунок 5.3 – Процесс работы команды docker-compose up.

6 Результат работы

Заключение