

Дробышев Антон Вячеславович

Документация к CRM проекту

2023 год

Содержание

1. Введение.....	3
2. Инструкция по развертыванию проекта.....	4
2.1. Установка актуальных версий приложений.	4
2.2. Работа с API.	4
2.3. Работа с Web, WPF и Telegram-bot приложениями.	5
2.4. Дополнение по работе с Telegram-бот приложением.....	5
2.5. Логин и пароль	6
3. API – сервисное приложение.	7
3.1. Используемые библиотеки в API – сервисном приложении.	7
3.2. Описание API функций.....	7
Основная информация	8
3.2.1. Функции взаимодействия с Application (заявки).....	8
3.2.2. Функции взаимодействия с Projects (проекты).....	8
3.2.3. Функции взаимодействия с Services (услуги).....	9
3.2.4. Функции взаимодействия с Blog (блог)	10
3.2.5. Функции взаимодействия с Contacts (контакты)	11
3.2.6. Функции взаимодействия с Links (ссылки)	12
3.2.7. Функции взаимодействия с Tag (тэг).....	13
3.2.8. Функции взаимодействия с Title (титальный лист).....	13
3.2.9. Функции взаимодействия с Authorization (методы аутентификации и авторизации)	13
4. Web – приложение.	15
4.1. Используемые библиотеки в Web – приложении.	15
5. WPF – приложение.	16
5.1. Используемые библиотеки в WPF – приложении.	16
6. Telegram - бот.	17
6.1. Используемые библиотеки в Telegram - боте.....	17

1. Введение

В итоговой работе реализованы 4 приложения:

- 1) API – сервисное приложение.
- 2) Web – приложение.
- 3) WPF – приложение.
- 4) Telegram – бот.

Целевой платформой приложений является .NET с версией 7.0. Основным языком программирования, на котором написаны приложения является C#.

Для Web приложения, помимо C# использовались языки программирования HTML (вёрстка сайта), CSS (стили для вёрстки) и JavaScript (небольшой объём кода во View).

2. Инструкция по развертыванию проекта.

2.1. Установка актуальных версий приложений.

Для установки актуальных версий приложений необходимо перейти по ссылкам репозитория приложений на GitHub и скачать их.

Ссылка на API – сервисное приложение:

https://github.com/AntonDrobyshev94/FinalProject_API.git

Ссылка на Web – приложение :

https://github.com/AntonDrobyshev94/FinalProject_Web.git

Ссылка на WPF – приложение:

https://github.com/AntonDrobyshev94/FinalProject_WPF.git

Ссылка на Telegram-bot:

https://github.com/AntonDrobyshev94/FinalProject_Telegram-bot.git

2.2. Работа с API.

Для работы с API необходимо произвести следующие шаги:

- 1) Запустить приложение с помощью файла FinalProject_API.sln
- 2) Удостовериться, что среди файлов и папок присутствует папка Migrations с содержащимися в ней миграциями (2 файла).
- 2.1. При отсутствии файлов, в русской версии Visual Studio – открыть

Средства -> Диспетчер пакетов NuGet -> Консоль диспетчера пакетов.

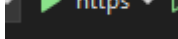
В консоли диспетчера ввести команду “add-migration migration_1”, где migration_1 – название миграции.

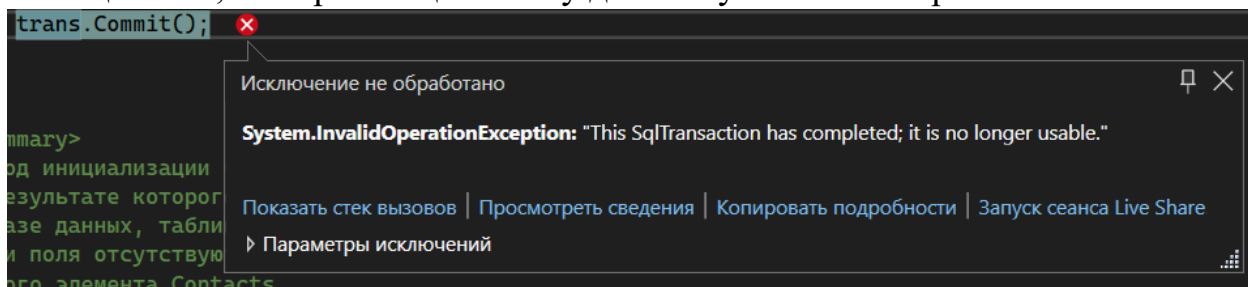
- 2.2) При наличии папки и файлов – перейти к следующему пункту.
- 3) Обновить базу данных/создать новую (по умолчанию, база данных имеет название **FinalProjectAPIDB** - можно найти и изменить в файле appsettings.json)

Для обновления базы данных, , в русской версии Visual Studio – открыть

Средства -> Диспетчер пакетов NuGet -> Консоль диспетчера пакетов.

В консоли диспетчера ввести команду “update-database”.

- 4) Запустить программу  и ожидать следующего окна, которое сообщит нам, что транзакция в базу данных успешно завершена:



- 5) Остановить работу программы  и повторно запустить программу



. В результате чего должна открыться страница с адресом <https://localhost:7037/api/values>

- 6) Для дальнейшей работы с приложениями Web, WPF и Telegram-bot'ом необходима работа API в фоновом режиме (постоянно запущенном). Для этих целей можно запускать в том числе файл FinalProject_API.exe, расположенный в папке по пути:
FinalProject_API\FinalProject_API\bin\Debug\net7.0

2.3. Работа с Web, WPF и Telegram-bot приложениями.

1. Для работы с Web, WPF и Telegram-bot приложениями, после их установки, необходимо найти файлы FinalProject_Web.sln, FinalProject_WPF.sln, FinalProject_API.sln соответственно и открыть их или использовать их exe файлы, которые расположены в папке bin проектов по пути ...\\bin\\Debug\\net7.0

2. Запустить приложение с помощью кнопки запуска .

2.4. Дополнение по работе с Telegram-бот приложением

Для проверки работоспособности Telegram-бот приложения необходимо создать бота на сайте Telegram и ввести полученный токен в переменную token:

```
public class Program
{
    static string service = string.Empty;
    static string token = "";

    Ссылка: 0
    static void Main(string[] args)
    {
```

2.5. Логин и пароль

Для работы с приложением по умолчанию при создании миграций создаётся 2 пользователя: пользователь **admin** с паролем **admin** (роль администратора) и пользователь **user** с паролем **user** (роль обычного пользователя).

3. API – сервисное приложение.

3.1. Используемые библиотеки в API – сервисном приложении.

В API – сервисном приложении использовано 5 пакетов “верхнего уровня”, версии 7.0.10. и 72 транзитивных пакета. Помимо основных библиотек .NET, таких как Microsoft.Extensions, Microsoft.AspNetCore и различных системных библиотек, использованы следующие библиотеки:

- 1) Библиотека авторизации и аутентификации ASP.NET Core:
 - 1.1) Microsoft.AspNetCore.Authentication.JwtBearer – библиотека, которая содержит типы для проверки подлинности на основе JWT токенов;
 - 1.2) Microsoft.AspNetCore.Identity.EntityFrameworkCore – библиотека, которая служит для создания базы данных пользователей (управление паролями, ролями, именами пользователей и т.д.)
 - 1.3) System.IdentityModel – библиотеки, содержащие классы и интерфейсы для работы с токенами безопасности, включая классы для создания, проверки и обработки JSON Web Tokens (JWT).
- 2) Библиотеки базы данных EntityFrameworkCore:
 - 2.1) Microsoft.AspNetCore.Identity.EntityFrameworkCore - библиотека, которая служит для создания базы данных пользователей (управление паролями, ролями, именами пользователей и т.д.)
 - 2.2) Microsoft.EntityFrameworkCore.SqlServer – библиотека, которая позволяет использовать базу данных Microsoft SQL Server.
 - 2.3) Microsoft.EntityFrameworkCore.Tools – библиотека, которая служит помощником для взаимодействия разработчика с базой данных EntityFramework (в частности реализует возможность осуществлять миграцию базы данных: автоматическое создание БД на основе контекста данных).
- 3) Библиотека сериализации/десериализации объектов:
Microsoft.AspNetCore.JsonSerializer версии **13.0.3** – библиотека, которая служит для сериализации объектов в формат JSON (JavaScript Object Notation) и десериализации объектов из этого формата. Является удобным средством для передачи данных между API сервисом и взаимодействующими с ней приложениями.

3.2. Описание API функций

Основная информация

Url адрес API задан по адресу <https://localhost:7037>.

Для обращения к контроллеру API необходимо добавить строку api и название контроллера (Values): <https://localhost:7037/api/values>

3.2.1. Функции взаимодействия с Application (заявки)

1) Get запрос на возвращение списка заявок

url запроса - <https://localhost:7037/api/values>

Является Get запросом.

Возвращает последовательность коллекции объектов, реализующих интерфейс IApplication с помощью интерфейса IEnumerable.

2) Post запрос добавления новой заявки

url запроса - <https://localhost:7037/api/values>

Является Post запросом.

Принимает экземпляр заявки (модель типа Application).

Экземпляр сохраняется в таблице Requests базы данных.

3) Delete запрос удаления заявки

url запроса - <https://localhost:7037/api/values/{id}>, где id – идентификационный номер заявки.

Является Delete запросом.

Запрос имеет атрибут Authorize с Policy AdminOnly (доступ только для пользователей с токеном, содержащим сведения о роли администратора).

Принимает id заявки в url запросе и удаляет экземпляр заявки из таблицы Requests базы данных. Результат сохраняется.

4) Post запрос изменения статуса заявки

url запроса - <https://localhost:7037/api/values/ChangeStatus>

Является Post запросом.

Запрос имеет атрибут Authorize с Policy AdminOnly (доступ только для пользователей с токеном, содержащим сведения о роли администратора).

Принимает экземпляр заявки (модель типа Application).

Экземпляр сохраняется в таблице Requests базы данных вместо экземпляра со сходным Id.

3.2.2. Функции взаимодействия с Projects (проекты)

1) Get запрос на возвращение списка проектов

url запроса - <https://localhost:7037/api/values/GetProjects>

Является Get запросом.

Возвращает последовательность коллекции объектов, реализующих интерфейс IProjectModel с помощью интерфейса IEnumerable.

2) Post запрос добавления нового проекта

url запроса - <https://localhost:7037/api/values/AddProject>

Является Post запросом.

Запрос имеет атрибут Authorize с Policy AdminOnly (доступ только для пользователей с токеном, содержащим сведения о роли администратора).

Принимает экземпляр проекта (модель типа ProjectModel).

Экземпляр сохраняется в таблице Projects базы данных.

3) Delete запрос удаления проекта

url запроса - <https://localhost:7037/api/values/DeleteProject/{id}>, где id – идентификационный номер проекта.

Является Delete запросом.

Запрос имеет атрибут Authorize с Policy AdminOnly (доступ только для пользователей с токеном, содержащим сведения о роли администратора).

Принимает id проекта в url запросе и удаляет экземпляр проекта из таблицы Projects базы данных. Результат сохраняется.

4) Post запрос изменения проекта

url запроса - <https://localhost:7037/api/values/ChangeProjects>

Является Post запросом.

Запрос имеет атрибут Authorize с Policy AdminOnly (доступ только для пользователей с токеном, содержащим сведения о роли администратора).

Принимает экземпляр проекта (модель типа ProjectModel).

Экземпляр сохраняется в таблице Projects базы данных вместо экземпляра со сходным Id.

5) Get запрос поиска проекта

url запроса - <https://localhost:7037/api/values/ProjectDetails/{id}>

Является Get запросом.

Принимает id проекта в url запросе и возвращает экземпляр проекта, полученный в результате перебора таблицы Projects базы данных на предмет совпадения id, в виде модели ProjectModel.

3.2.3. Функции взаимодействия с Services (услуги)

1) Get запрос на возвращение списка услуг

url запроса - <https://localhost:7037/api/values/GetServices>

Является Get запросом.

Возвращает последовательность коллекции объектов, реализующих интерфейс IService с помощью интерфейса IEnumerable.

2) Post запрос добавления новой услуги

url запроса - <https://localhost:7037/api/values/AddService>

Является Post запросом.

Запрос имеет атрибут Authorize с Policy AdminOnly (доступ только для пользователей с токеном, содержащим сведения о роли администратора).

Принимает экземпляр услуги (модель типа Service).

Экземпляр сохраняется в таблице Services базы данных.

3) Delete запрос удаления услуги

url запроса - <https://localhost:7037/api/values/DeleteService/{id}>, где id – идентификационный номер услуги.

Является Delete запросом.

Запрос имеет атрибут Authorize с Policy AdminOnly (доступ только для пользователей с токеном, содержащим сведения о роли администратора).

Принимает id услуги в url запросе и удаляет экземпляр услуги из таблицы Services базы данных. Результат сохраняется.

4) Post запрос изменения услуги

url запроса - <https://localhost:7037/api/values/ChangeService>

Является Post запросом.

Запрос имеет атрибут Authorize с Policy AdminOnly (доступ только для пользователей с токеном, содержащим сведения о роли администратора).

Принимает экземпляр услуги (модель типа Service).

Экземпляр сохраняется в таблице Services базы данных вместо экземпляра со сходным Id .

5) Get запрос поиска услуги

url запроса - <https://localhost:7037/api/values/FindService/{id}>

Является Get запросом.

Принимает id услуги в url запросе и возвращает экземпляр услуги, полученный в результате перебора таблицы Services базы данных на предмет совпадения id, в виде модели Service.

3.2.4. Функции взаимодействия с Blog (блог)

1) Get запрос на возвращение списка записей в блоге

url запроса - <https://localhost:7037/api/values/GetBlog>

Является Get запросом.

Возвращает последовательность коллекции объектов, реализующих интерфейс IBlogModel с помощью интерфейса IEnumerable.

2) Post запрос добавления новой записи в блоге

url запроса - <https://localhost:7037/api/values/ChangeBlog>

Является Post запросом.

Запрос имеет атрибут Authorize с Policy AdminOnly (доступ только для пользователей с токеном, содержащим сведения о роли администратора).
Принимает экземпляр записи в блоге (модель типа BlogModel).
Экземпляр сохраняется в таблице Blogs базы данных.

3) Delete запрос удаления записи в блоге

url запроса - <https://localhost:7037/api/values/DeleteBlog/{id}>, где id – идентификационный номер записи в блоге.

Является Delete запросом.

Запрос имеет атрибут Authorize с Policy AdminOnly (доступ только для пользователей с токеном, содержащим сведения о роли администратора).
Принимает id записи в блоге в url запросе и удаляет экземпляр записи из таблицы Blogs базы данных. Результат сохраняется.

4) Post запрос изменения записи в блоге

url запроса - <https://localhost:7037/api/values/ChangeBlog>

Является Post запросом.

Запрос имеет атрибут Authorize с Policy AdminOnly (доступ только для пользователей с токеном, содержащим сведения о роли администратора).
Принимает экземпляр записи в блоге (модель типа BlogModel).
Экземпляр сохраняется в таблице Blogs базы данных вместо экземпляра со сходным Id .

5) Get запрос поиска записи в блоге

url запроса - <https://localhost:7037/api/values/BlogDetails/{id}>

Является Get запросом.

Принимает id записи в блоге в url запросе и возвращает экземпляр записи, полученный в результате перебора таблицы Blogs базы данных на предмет совпадения id, в виде модели BlogModel.

3.2.5. Функции взаимодействия с Contacts (контакты)

1) Get запрос на возвращение экземпляра контактов

url запроса - <https://localhost:7037/api/values/GetContacts>

Является Get запросом.

Возвращает экземпляр контактов Contacts.

2) Post запрос изменения контактов

url запроса - <https://localhost:7037/api/values/ChangeContacts>

Является Post запросом.

Запрос имеет атрибут Authorize с Policy AdminOnly (доступ только для пользователей с токеном, содержащим сведения о роли администратора).
Принимает экземпляр контактов (модель типа Contacts).

Экземпляр сохраняется в таблице Contacts базы данных.

3.2.6. Функции взаимодействия с Links (ссылки)

1) Get запрос на возвращение списка ссылок

url запроса - <https://localhost:7037/api/values/GetLinks>

Является Get запросом.

Возвращает последовательность коллекции объектов, реализующих интерфейс ILinkModel с помощью интерфейса IEnumerable.

2) Post запрос добавления новой ссылки

url запроса - <https://localhost:7037/api/values/AddLink>

Является Post запросом.

Запрос имеет атрибут Authorize с Policy AdminOnly (доступ только для пользователей с токеном, содержащим сведения о роли администратора).

Принимает экземпляр ссылки (модель типа LinkModel). Экземпляр сохраняется в таблице Links базы данных.

3) Delete запрос удаления ссылки

url запроса - <https://localhost:7037/api/values/DeleteLink/{id}>, где id – идентификационный номер ссылки.

Является Delete запросом.

Запрос имеет атрибут Authorize с Policy AdminOnly (доступ только для пользователей с токеном, содержащим сведения о роли администратора).

Принимает id ссылки в url запросе и удаляет экземпляр ссылки из таблицы Links базы данных. Результат сохраняется.

4) Post запрос изменения ссылки

url запроса - <https://localhost:7037/api/values/LinkDetails>

Является Post запросом.

Запрос имеет атрибут Authorize с Policy AdminOnly (доступ только для пользователей с токеном, содержащим сведения о роли администратора).

Принимает экземпляр ссылки (модель типа LinkModel).

Экземпляр сохраняется в таблице Links базы данных вместо экземпляра со сходным Id .

5) Get запрос поиска ссылки

url запроса - <https://localhost:7037/api/values/LinkDetails/{id}>

Является Get запросом.

Принимает id ссылки в url запросе и возвращает экземпляр ссылки, полученный в результате перебора таблицы Links базы данных на предмет совпадения id, в виде модели LinkModel.

3.2.7. Функции взаимодействия с Tag (тэг)

1) Get запрос на возвращение списка тэгов

url запроса - <https://localhost:7037/api/values/GetTags>

Является Get запросом.

Возвращает последовательность коллекции объектов, реализующих интерфейс ITagModel с помощью интерфейса IEnumerable.

2) Post запрос добавления тэга

url запроса - <https://localhost:7037/api/values/AddTag>

Является Post запросом.

Запрос имеет атрибут Authorize с Policy AdminOnly (доступ только для пользователей с токеном, содержащим сведения о роли администратора).

Принимает экземпляр тэга (модель типа TagModel). Экземпляр сохраняется в таблице Tags базы данных.

3) Delete запрос удаления тэгов

url запроса - <https://localhost:7037/api/values/DeleteTag/{id}>, где id – идентификационный номер тэга.

Является Delete запросом.

Запрос имеет атрибут Authorize с Policy AdminOnly (доступ только для пользователей с токеном, содержащим сведения о роли администратора).

Принимает id тэга в url запросе и удаляет экземпляр тэга из таблицы Tags базы данных. Результат сохраняется.

3.2.8. Функции взаимодействия с Title (титульный лист)

1) Get запрос на возвращение экземпляра титульного меню

url запроса - <https://localhost:7037/api/values/GetTitle>

Является Get запросом.

Возвращает экземпляр титульного меню TitleModel.

2) Post запрос изменения титульного меню

url запроса - <https://localhost:7037/api/values/ChangeTitle>

Является Post запросом.

Запрос имеет атрибут Authorize с Policy AdminOnly (доступ только для пользователей с токеном, содержащим сведения о роли администратора).

Принимает экземпляр титульного меню (модель типа TitleModel).

Экземпляр сохраняется в таблице Title базы данных.

3.2.9. Функции взаимодействия с Authorization (методы аутентификации и авторизации)

1) Post запрос регистрации нового пользователя

url запроса - <https://localhost:7037/api/values/Registration>

Является Post запросом.

Возвращает экземпляр модели TokenResponseModel, содержащей токен и имя пользователя.

2) Post запрос аутентификации пользователя (проверки подлинности) и авторизации

url запроса - <https://localhost:7037/api/values/Authenticate>

Является Post запросом.

Принимает экземпляр UserLoginProp (модель авторизации, содержащая имя пользователя и пароль). В результате успешной аутентификации посылает ответ в виде модели TokenResponseModel, содержащей токен и имя пользователя. При неудачной аутентификации посылается ответ Unauthorized (вызывает ошибку авторизации 401).

3) Get запрос проверки токена

url запроса - <https://localhost:7037/api/values/CheckToken>

Является Get запросом.

Данный запрос производит проверку HTTP-запроса на предмет наличия токена в его заголовке (header). Если токен валидный (процедура аутентификации пройдена и срок токена не истёк), происходит возврат логической переменной с результатом true. При невалидности – false.

4. Web – приложение.

4.1. Используемые библиотеки в Web – приложении.

В Web –приложении использовано 2 пакета “верхнего уровня”, версии 7.0.10. и 12 транзитивных пакетов. Помимо основных библиотек .NET, таких как системные, использованы следующие библиотеки:

- 1) Библиотека аутентификации:
 - 1.1) System.IdentityModel.Tokens.Jwt – библиотека, предоставляющая функционал для работы с токенами безопасности JSON Web Token (JWT).
- 2) Библиотека сериализации/десериализации объектов:
Microsoft.AspNetCore.JsonSerializer версии **13.0.3** – библиотека, которая служит для сериализации объектов в формат JSON (JavaScript Object Notation) и десериализации объектов из этого формата. Является удобным средством для передачи данных между API сервисом и взаимодействующими с ней приложениями.

5. WPF – приложение.

5.1. Используемые библиотеки в WPF – приложении.

В WPF – приложении использован 1 пакет “верхнего уровня”, **версии 13.0.3** – библиотека Newtonsoft.Json, которая необходима для сериализации/десериализации объектов в формат JSON (JavaScript Object Notation) и десериализации объектов из этого формата. Является удобным средством для передачи данных между API сервисом и взаимодействующими с ней приложениями.

Также в проекте использована dll библиотека FinalProject_WPF_ClassLibrary, в которой содержатся основные переменные и логика приложения.

6. Telegram - бот.

6.1. Используемые библиотеки в Telegram - боте.

В Telegram - бот – приложении использовано 2 пакета “верхнего уровня”.

1. Библиотека `NewtonSoft.Json` версии 13.0.3 – библиотека, которая необходима для сериализации/десериализации объектов в формат JSON (JavaScript Object Notation) и десериализации объектов из этого формата. Является удобным средством для передачи данных между API сервисом и взаимодействующими с ней приложениями.
2. Библиотека `Telegram.Bot` версии 19.0.0 – библиотека с набором инструментов для разработки приложений, использующих API Telegram.