МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДОНЕЦКОЙ НАРОДНОЙ РЕСПУБЛИКИ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра ПИ

Факультет КНТ

Отчет по лабораторной работе №6

Тема: «Веб-сайт из репозитория».

Курс: «Профессиональная практика программной инженерии»

Выполнил

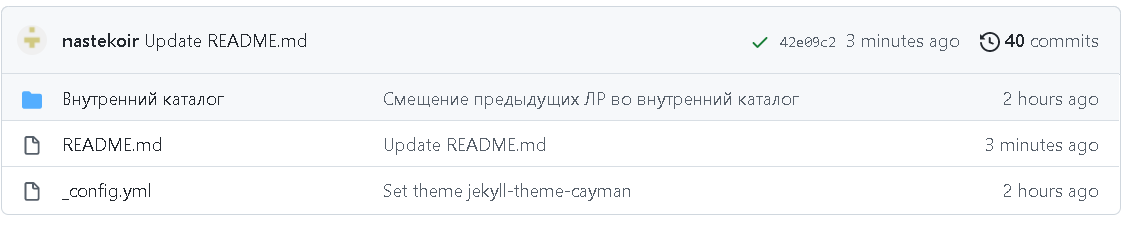
ст. гр. ПИ-18в

Настеко И.Р.

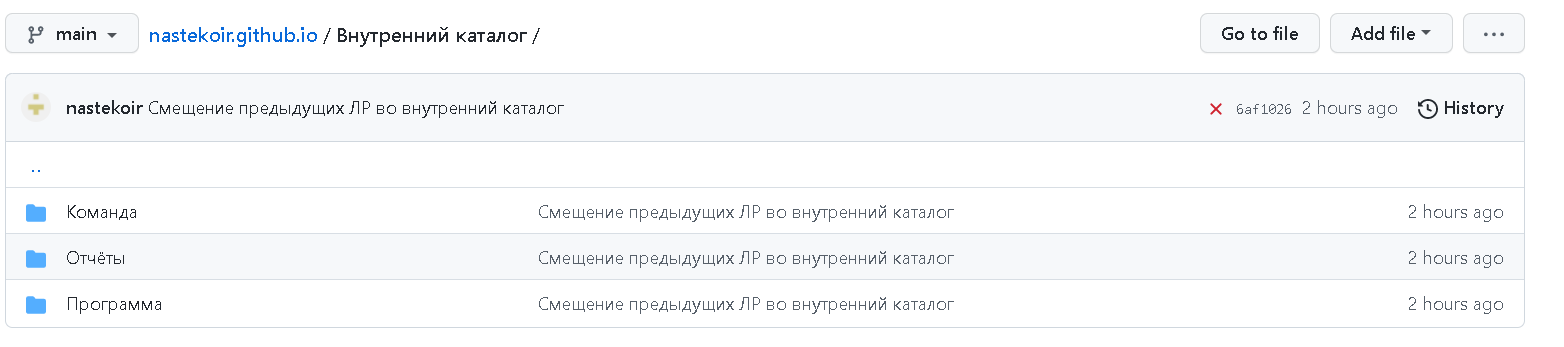
Донецк – 2022

Структура репозитория после фиксации коммита файлов веб-сайта.

Разметка одностраничного сайта находится в файле README.md



В этих каталогах находятся данные из предыдущих лабораторных работ



Разметка веб-сайта(языки разметки markdown и html)

# Форум с ветками-обсуждениями, с обязательной реализациейгрупповых ролей: гость, посетитель, модератор, администратор.

## Краткое описание разработанного продукта

1. Данная документация написана для проекта Форум с ветками-обсуждениями, с обязательной реализациейгрупповых ролей: гость, посетитель, модератор, администратор.

2. Разработанный продукт может быть использован в различных целях. Например как площадка для обсуждения программирования, решения проблем и помощи друг другу. Или как торговая площадка, где люди могут продавать, покупать, обмениватся различными вещами

3. Реализована минимальная часть для функционирования форума, а именно: регистрация и авторизация пользователя, панель администратора, создания веток обсуждения и отправка сообщений

4. Написаны Unit тесты для правильной работы программы, а также выявлены все возможные ошибки

5. Проект написан на языке C# с использованием фреймворка ASP.NET Core MVC

## Сценарии использования приложения

#### Сценарий 1. Работа с ветками

Пользователь создаёт ветку функцией CreateBranch

````

public void CreateBranch(string branchName)

{

//INSERT BranchTable branchName

}

````

Затем может переименовать её функцией RenameBranch

````

public void RenameBranch(Guid branchId, string newName)

{

//UPDATE BranchTable SET Name=newName WHERE Id=branchId

}

````

И удалить DeleteBranch

````

public void DeleteBranch(Guid branchId)

{

//DELETE BranchTable WHERE Id=branchId

}

````

#### Сценарий 2. Работа с сообщениями

Пользователь создаёт ветку функцией CreateBranch

````

public void CreateBranch(string branchName)

{

//INSERT BranchTable branchName

}

````

Затем может отправить сообщение в ветку функцией SendMessage

````

public void SendMessage(Guid branchId, Message message)

{

//INSERT MessageTable message.MessageText,message.UserId,message.CreatedDate,branchId

}

````

После отправки можно редактировать сообщение, если было отправлено с ошибкой

````

public void EditMessage(Guid messageId, string messageText)

{

//UPDATE MessageTeble SET MessageText=messageText WHERE Id=messageId

}

````

Или удалить при необходимости

````

public void DeleteMessage(Guid messageId)

{

//DELETE MessageTable WHERE Id=messageId

}

````

#### Сценарий 3. Работа с пользователями

Пользователь зайдя под гостем, может зарегистрироваться на форуме, что бы стать посетителем

````

public bool ResgisterNewUser(IUser user)

{

if(true)

return true;

}

````

Затем ему нужно авторизоваться, для чего сначала вызывается функция проверки на существование пользователя в базе данных

````

public bool CheckIsUserExist(Guid userID)

{

return true;

}

````

После чего происходит авторизация

````

public bool AuthorizeUser(string userName, string password)

{

return true;

}

````

## Краткое описание компонентов модулируемого проекта

#### Модели

#### Роли

У всех ролей кроме гостя есть обязательные поля: Айди ,Имя, Логин, Пароль, Почта, Права

\*\*Администратор\*\*

````

public class Admin : IUser

{

public Guid Id { get; }

public string Name { get; set; }

public string Login { get; set; }

public string Password { get; set; }

public string Email { get; set; }

public List<UserRightsEnum> Rights { get; set; }

}

````

\*\*Модератор\*\*

````

public class Moderator : IUser

{

public Guid Id { get; }

public string Name { get; set; }

public string Login { get; set; }

public string Password { get; set; }

public string Email { get; set; }

public List<UserRightsEnum> Rights { get; set; }

}

````

\*\*Посетитель\*\*

````

public class Visitor : IUser

{

public Guid Id { get; }

public string Name { get; set; }

public string Login { get;set; }

public string Password { get; set; }

public string Email { get; set; }

public List<UserRightsEnum> Rights { get; set; }

}

````

\*\*Гость\*\*

````

public class Guest

{

public string Name { get; set; }

public List<UserRightsEnum> Rights { get; set; }

}

````

\*\*Перечисление для прав пользователей\*\*

````

public enum UserRightsEnum

{

CreateBranch,

DeleteBranch,

BanUser,

UnBanUser,

AddMessage,

DeleteMessage,

DeleteUserMessage,

ReadMessages

}

````

CreateBranch, – создание ветки

DeleteBranch, – удаления ветки

BanUser, – забанить пользователя

UnBanUser, – разбанить пользователя

AddMessage, – отправить сообщение

DeleteMessage, – удалить сообщение

DeleteUserMessage, – удалить сообщение определённого юзера

ReadMessages – чтение сообщений

Также созданы модели \*\*Сообщение, Ветка обсуждений\*\*

````

public class Message

{

public Guid Id { get; set; }

public string MessageText { get; set; }

public Guid UserId { get; set; }

public DateTime CreatedDate { get; set; }

public Guid BranchId { get; set; }

}

````

````

public class Branch

{

public Guid Id { get; set; }

public string Name { get; set; }

public int MessageCount { get; set; }

}

````

####Сервисы

Для взаимодействия с моделями созданы интерфейсы сервисов

\*\*Функции для администратора\*\*

````

internal interface IAdminService

{

bool BanUser(Guid userId);

bool AppointModerator(Guid userId);

void SetRights(Guid userId, UserRightsEnum[] rights);

}

````

BanUser – забанить пользователя

AppointModerator – назначить пользователю права модератора

\*\*Функции веток обсуждения\*\*

````

public interface IBranchService

{

void CreateBranch(string branchName);

void DeleteBranch(Guid branchId);

}

````

CreateBranch/DeleteBranch – создать/удалить ветку

\*\*Функции иденификации пользователей\*\*

````

public interface IIdentityService

{

bool ResgisterNewUser(IUser user);

bool CheckIsUserExist(Guid userID);

bool AuthorizeUser(string userName, string password);

}

````

RegisterNewUser – зарегистрировать нового пользователя

CheckIsUserExist – проверить существование пользователя

AuthorizeUser – авторизовать пользователя

\*\*Функции сообщений\*\*

````

public interface IMessageService

{

void SendMessage(Guid branchId, Message message);

void DeleteMessage(Guid messageId);

}

````

SendMessage – отправить сообщение в ветку

DeleteMessage – удалить сообщение

## Руководство программиста к проекту

\*\*"Форум с ветками-обсуждениями, с обязательной реализацией групповых ролей: гость, посетитель, модератор, администратор"\*\*

1. На данный момент реализованы следующие сервисы: IdentityService, BranchService, MessageService, AdminService

2. Для добавления новых сервисов требуется добавить интерфейс и затем его реализовать

3. Называть интерфейсы начиная буквой "I"

4. Для наименование свойств использовать CamelCasing и существительные

5. Для наименования методов использовать CamelCasing и глаголы

6. У каждого свойства, метода, класса должно быть осмысленное название. Например, если хотим "получить что-нибудь" с базы данных, то GetSomething()

7. Использовать одинаковый уровень отступа как для комментариев, так и для кода.

8. Использовать одну пустую строку для разделения логических групп кода.

9. Открывающая фигурная скобка не должна стоять на той же строке, что и if, for и т. д. Вместо этого она должна стоять на следующей строке. Пример

````

If(...)

{

//...

}

````

10. Все классы и их общедоступные члены должны быть тщательно задокументированы с использованием XML-комментариев.

<h2>Руководство пользователя к веб-приложению "Форум с ветками-обсуждениями, с обязательной реализацией

групповых ролей: гость, посетитель, модератор, администратор." </h2>

<ol>

<li>Разработанный продукт может быть использован в различных целях. Например как площадка для обсуждения программирования, решения проблем и помощи друг другу. Или как торговая площадка, где люди могут продавать, покупать, обмениватся различными вещами</li>

<li>Для запуска приложения требуется установить .Net 5 SDK</li>

<li>После запуска приложения у пользователя есть возможность войти в аккаунт или зарегистрироватся, или войти как гость</li>

<li>У зарегистрированных пользователей имеется возможность создавать и удалять свои ветки</li>

<li>У зарегистрированных пользователей имеется возможность отправять и удалить свои сообщения в созданных другими пользователями ветках обсуждений</li>

<li>Войдя под администратором можно назначить любого пользователя модератором системы или выдать пользователю определённые права. Также имеется возможность удалять любые ветки и сообщения</li>

<li>Для полноценного функционирования веб-приложения, его нужно развернуть на сервере</li>

<li>При возниковении ошибок обратиться к разработчикам приложения</li>

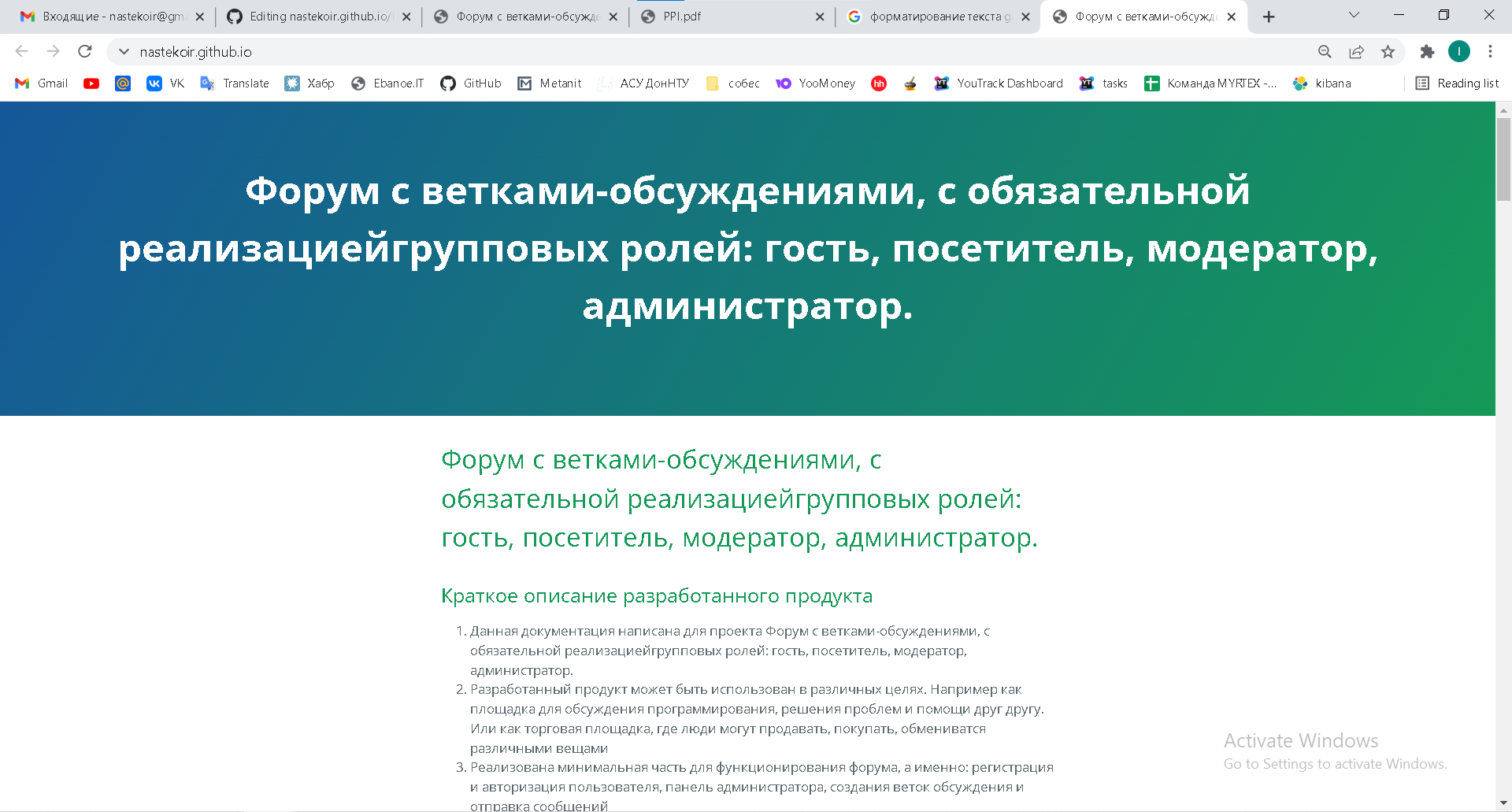
<li>При локальном запуске приложения необходимо дополнительно установить SQL Server</li>

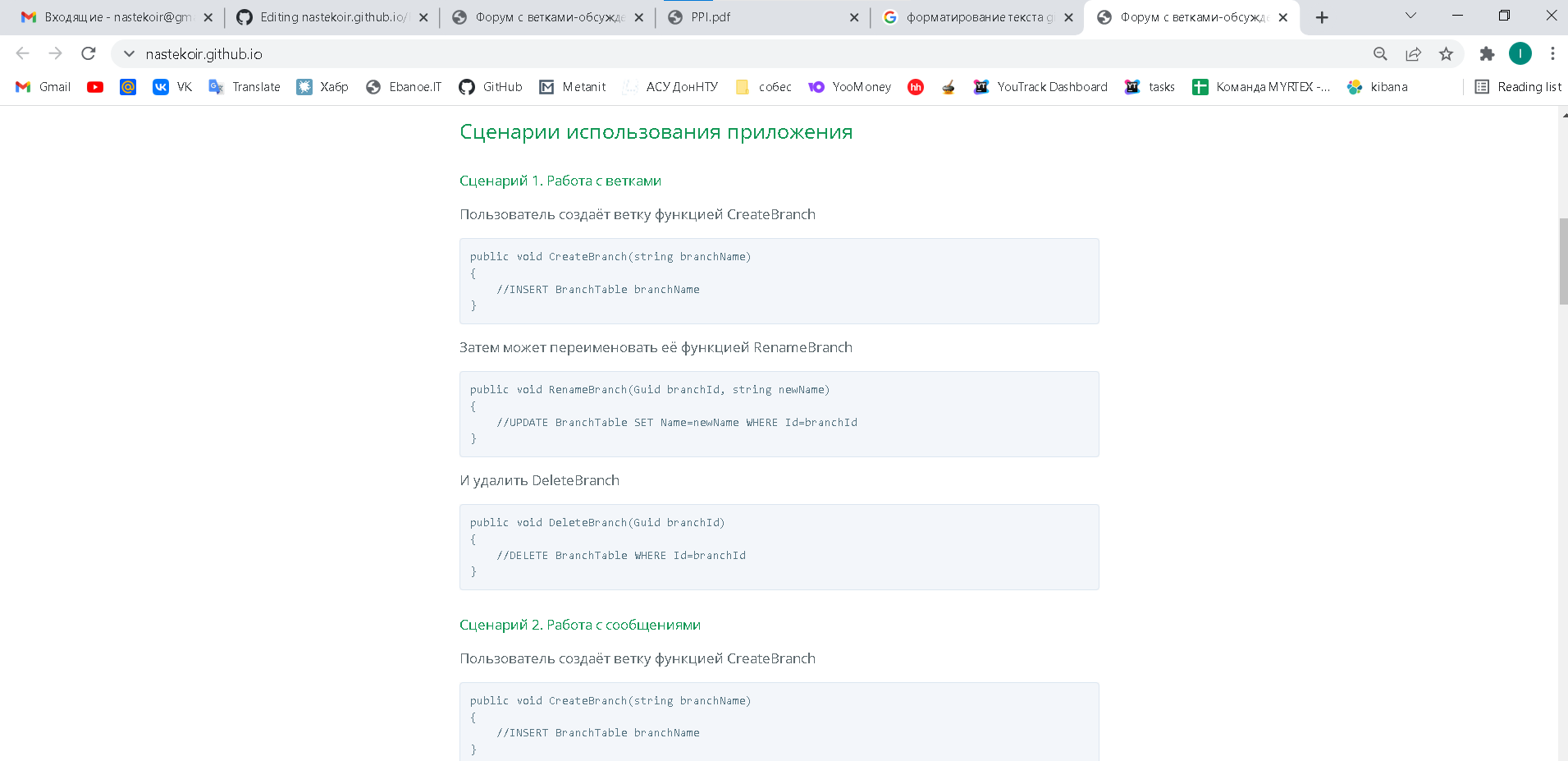
<li>При не правильном вводе данных пользователь будет предупреждён всплывающим сообщением</li>

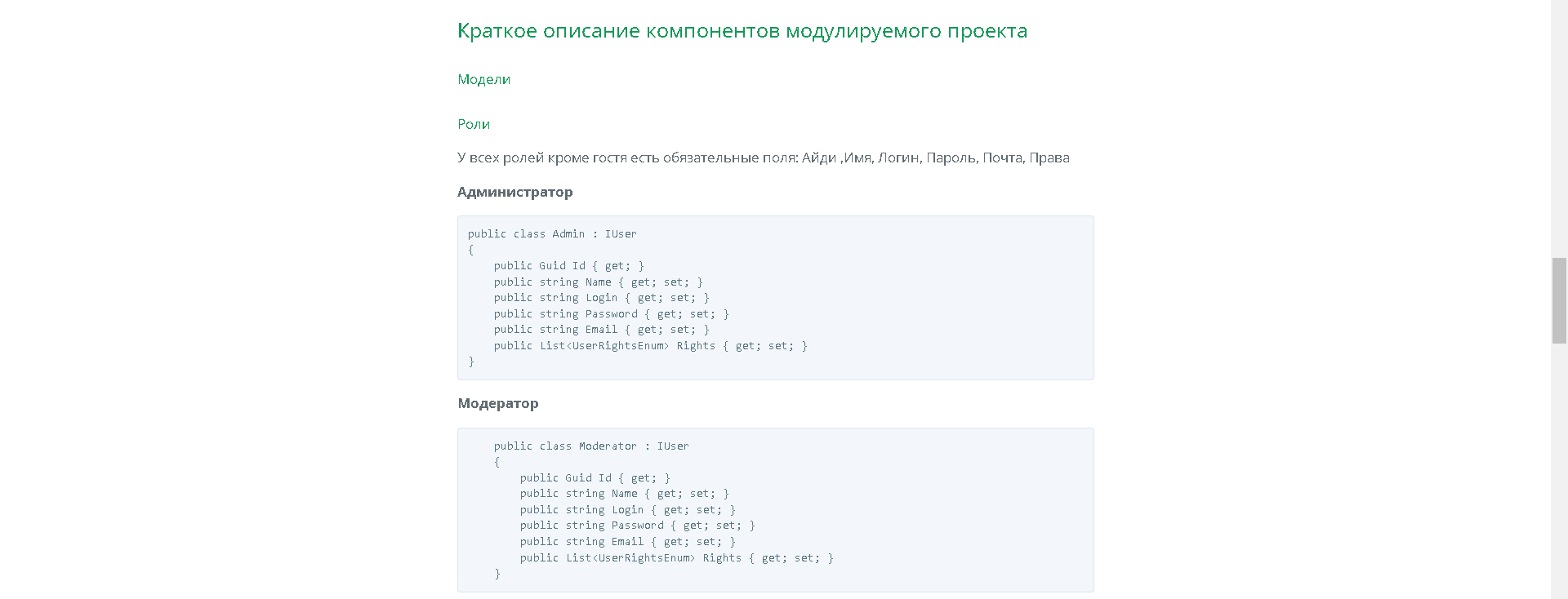
</ol>

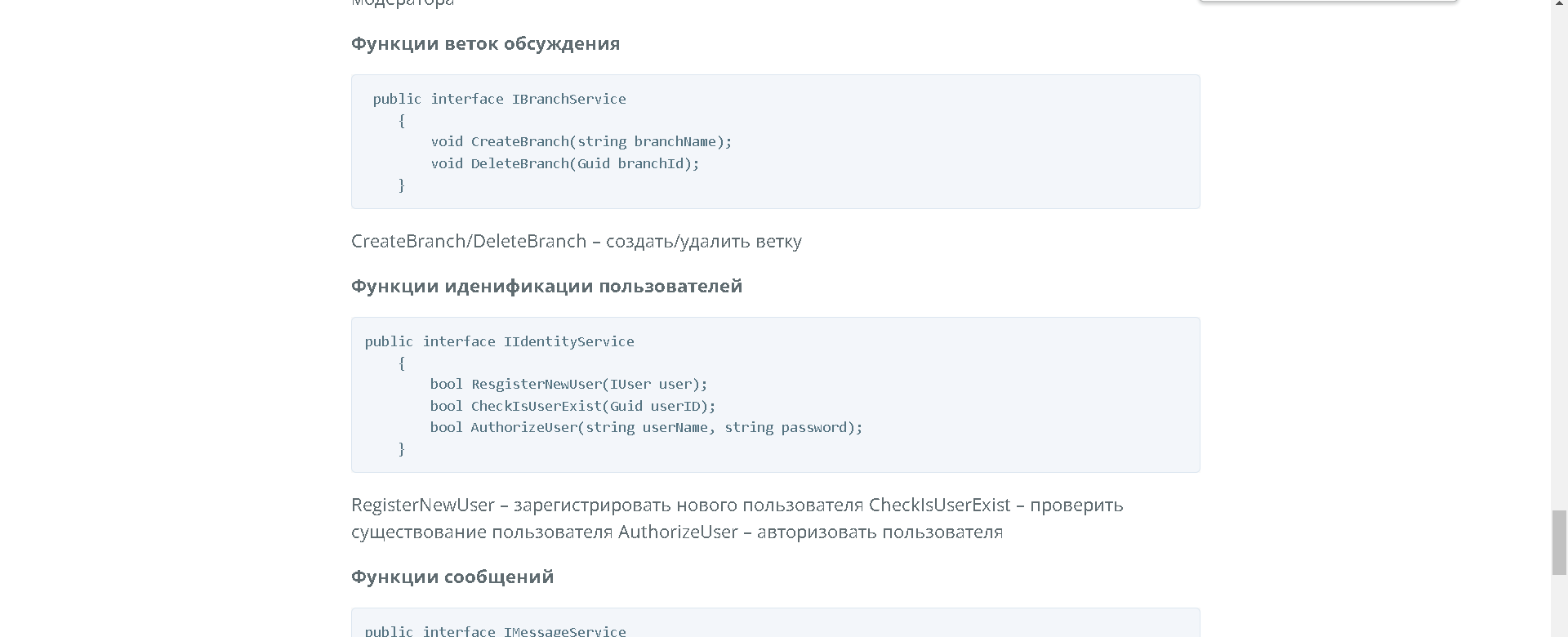
Демонстрация работы веб-сайта

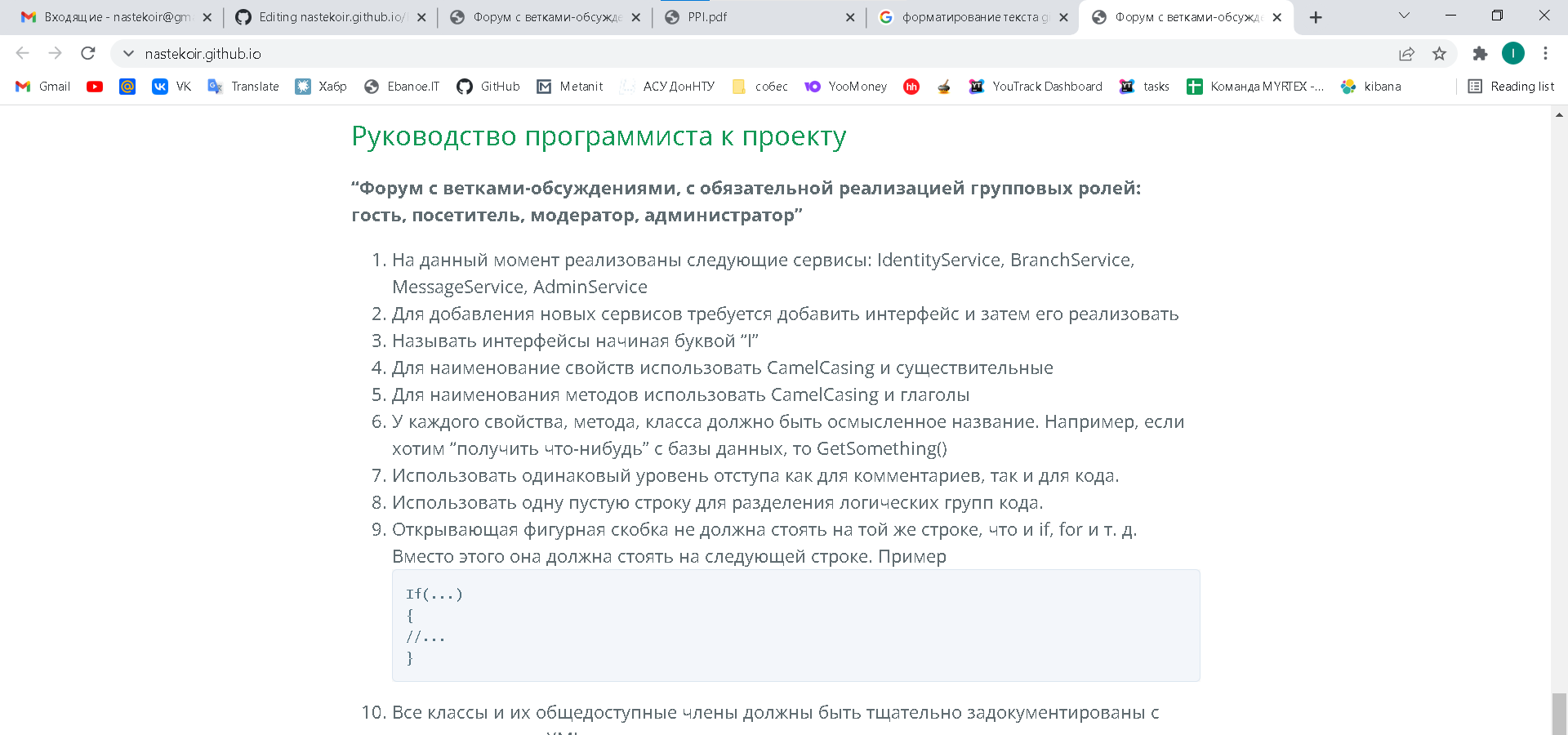
<https://nastekoir.github.io/>

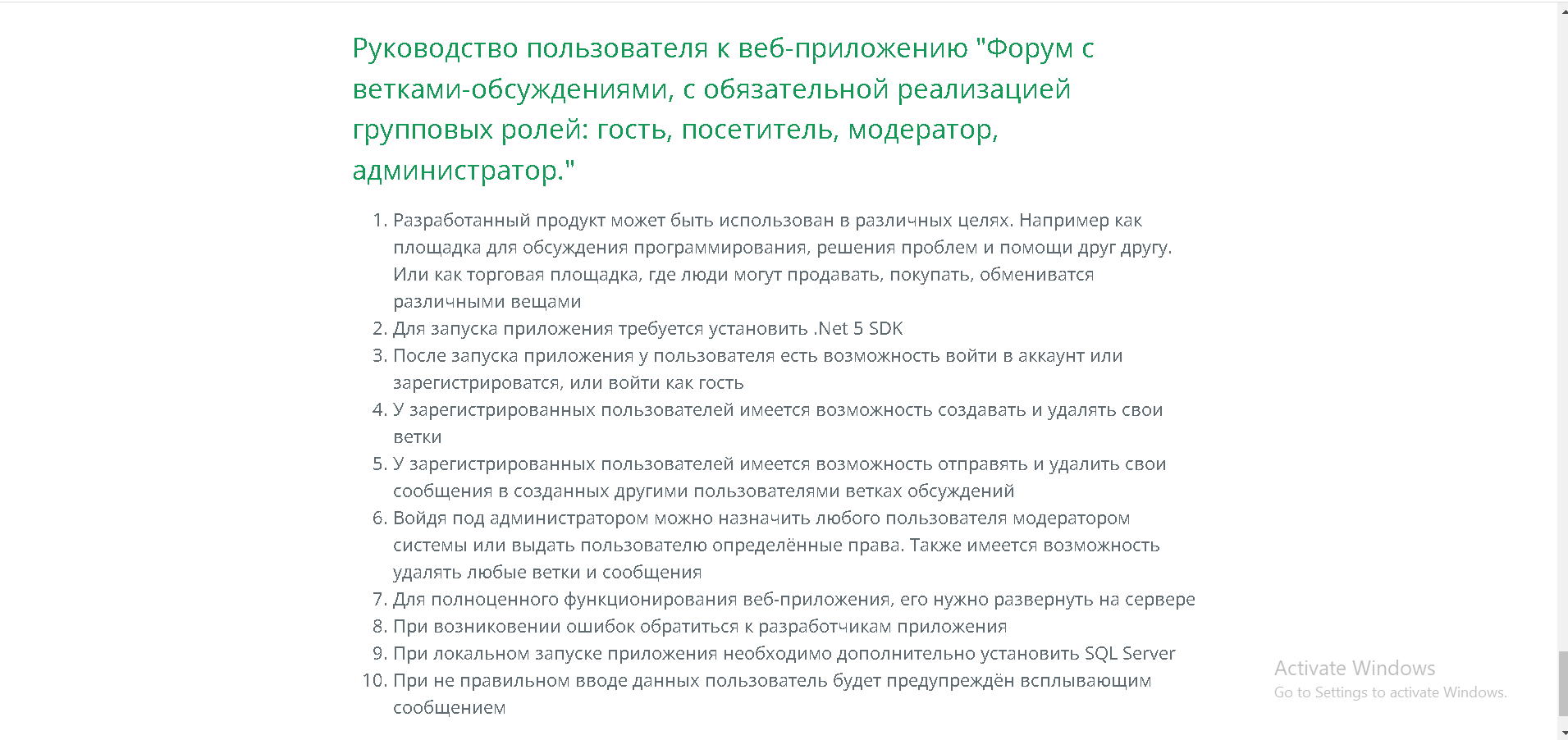
[](https://nastekoir.github.io/)

[](https://nastekoir.github.io/)

[](https://nastekoir.github.io/)

[](https://nastekoir.github.io/)

[](https://nastekoir.github.io/)

[](https://nastekoir.github.io/)