Содержание

[1 ОБЗОР ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И ЗАДАЧ, РЕШАЕМЫХ С ПОМОЩЬЮ САПР, АСУ НА ПРЕДПРИЯТИИ 2](#_Toc139398102)

[2 ПОСТАНОВКА ЗАДАЧИ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ 4](#_Toc139398103)

[3 СТРУКТУРНЫЕ СХЕМЫ РАЗРАБАТЫВАЕМОГО ПРОГРАММНОГО ПРОДУКТА 5](#_Toc139398104)

[4 РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА ДЛЯ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ 10](#_Toc139398105)

[5 ИСПОЛЬЗОВАНИЕ СЕТЕВЫХ ТЕХНОЛОГИЙ НА ПРЕДПРИЯТИИ 22](#_Toc139398106)

[6 СПОСОБЫ И СРЕДСТВА ЗАЩИТЫ ИНФОРМАЦИИ 23](#_Toc139398107)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 24](#_Toc139398108)

[ПРИЛОЖЕНИЯ 25](#_Toc139398109)

# 1 ОБЗОР ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И ЗАДАЧ, РЕШАЕМЫХ С ПОМОЩЬЮ САПР, АСУ НА ПРЕДПРИЯТИИ

Один из наиболее важных аспектов эффективного функционирования предприятий в современном мире - использование специализированного программного обеспечения для автоматизации бизнес-процессов. Например, системы автоматизированного проектирования (САПР) и автоматизации управления производством (АСУ) играют важную роль в повышении эффективности работы предприятий и оптимизации ресурсов.

Основными факторами, обеспечивающими возможность успешного создания автоматизированных информационно-управляющих систем, являются: развитие теории управления, математических методов и моделей, создание технических средств для сбора, передачи, фиксации, обработки и вывода информации, автоматического управления, телеобработки данных, диалогового режима и т.п.

САПР-системы позволяют проектным организациям проектировать и моделировать различные объекты и системы. Они обеспечивают возможность визуализации и анализа проектов, а также упрощают процесс внесения изменений и управления проектами. САПР может использоваться в разных отраслях, например в строительстве, машиностроении, электротехнике и других.

АСУ позволяют автоматизировать и оптимизировать управление и контроль различными производственными процессами, включая управление запасами, производственными линиями, планирование производства и многое другое. Автоматизация помогает снизить затраты на производство, улучшить качество продукции и повысить продуктивность.

Задачи, решаемые с помощью САПР и АСУ, могут включать:

1. Проектирование и моделирование объектов и систем;

2. Создание чертежей и технической документации;

3. Оптимизация процессов производства и сокращение времени и затрат;

4. Управление запасами и оптимизация инвентаризации;

5. Планирование производства и сокращение времени настройки;

6. Управление качеством продукции и контроль качества;

7. Автоматизация управления оборудованием и многое другое.

Использование специализированного программного обеспечения, такого как САПР и АСУ, может значительно повысить эффективность работы предприятий и снизить затраты. Эти системы имеют много преимуществ, одно из которых - возможность интеграции с другими системами предприятия.

Компания «EPAM Systems» является одной из крупнейших IT-компаний в мире, и они предлагают разнообразные IT-решения и услуги. Компания специализируется на проектировании и разработке заказного программного обеспечения, является специалистом по консалтингу, любой сложности для любой сферы бизнеса, создает уникальные продукты для решения конкретных задач и привлекает клиентов, повышает узнаваемости бренда, компании, увеличением продаж.

В компании используется различное программное обеспечение в своей работе. Некоторые из основных инструментов и технологий, применяемых EPAM, включают:

- Java, .NET, Python, Ruby, JavaScript, C++ и другие языки программирования для разработки приложений;

- Фреймворки и библиотеки, такие как Spring, Hibernate, React, Angular, Node.js и другие;

- Системы управления базами данных, такие как Oracle, MySQL, PostgreSQL, MongoDB и другие;

- Средства CI/CD (Continuous Integration/Continuous Delivery), такие как Jenkins, TeamCity, GitLab CI и другие;

- Системы контроля версий, такие как Git и SVN;

- Среды разработки, такие как IntelliJ IDEA, Eclipse, Visual Studio и другие;

- Веб-серверы, такие как Apache HTTP Server, Nginx и другие;

- Облачные платформы, такие как Amazon Web Services (AWS), Microsoft Azure и Google Cloud Platform;

- RPA (Robotic Process Automation) инструменты, такие как UiPath, Blue Prism и Automation Anywhere.

Конкретные инструменты и технологии, которые используются, могут варьироваться в зависимости от проекта и требований клиента.

# 2 ПОСТАНОВКА ЗАДАЧИ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ

Требуется создать сайт для обучающего центра «Training Center», связанного с проведением треннигов(практических курсов) онлайн. Данный сайт должен иметь приятный, интуитивно-понятный вид, наличие собственного профиля пользователя, возможность авторизоваться/зарегистрироваться в системе для прохождения тренингов и оставить комментарий.

Сведения о наставниках – это имена менторов, место и опыт работы, занимаемая должность. Раздел «О Нас» – это информационная вкладка об портале.

Тренировочный центр позволяет более оптимально организовать выполнение практических заданий пользователями, повысить навыки специалистов и сократить затраты на обучение.

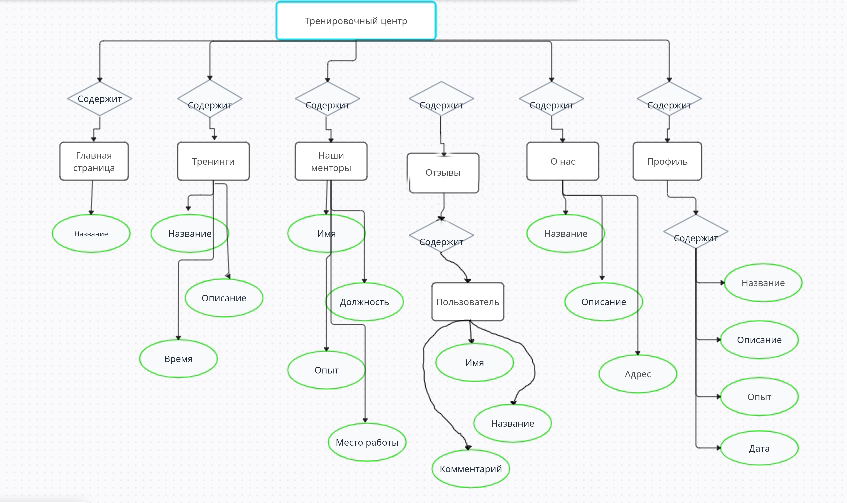
Целью проекта является разработка практической и информационной системы «Тренировочный Центр «EPAM Systems»», для чего необходимо решить ряд задач:

* формирование необходимого функционала;
* формирование и хранение базы данных;
* возможность зарегистрироваться/войти в систему;
* прохождение практических заданий;
* возможность оставить отзыв;
* предоставление данных, хранящихся в БД(информация о пользователе);

# 3 СТРУКТУРНЫЕ СХЕМЫ РАЗРАБАТЫВАЕМОГО ПРОГРАММНОГО ПРОДУКТА

Концептуальное моделирование — это деятельность по формальному описанию некоторых аспектов окружающего нас физического и социального мира в целях понимания и коммуникации.

Цель процесса концептуального моделирования заключается в предоставлении заинтересованным сторонам модели (общего видения) программной системы (или группы взаимосвязанных систем), выполняемых ею функций, описания информационного пространства и взаимодействия с другими внешними системами. Концептуальная модель БД представлена на рисунке 3.1.



**Рисунок 3.1** — **Концептуальная модель БД**

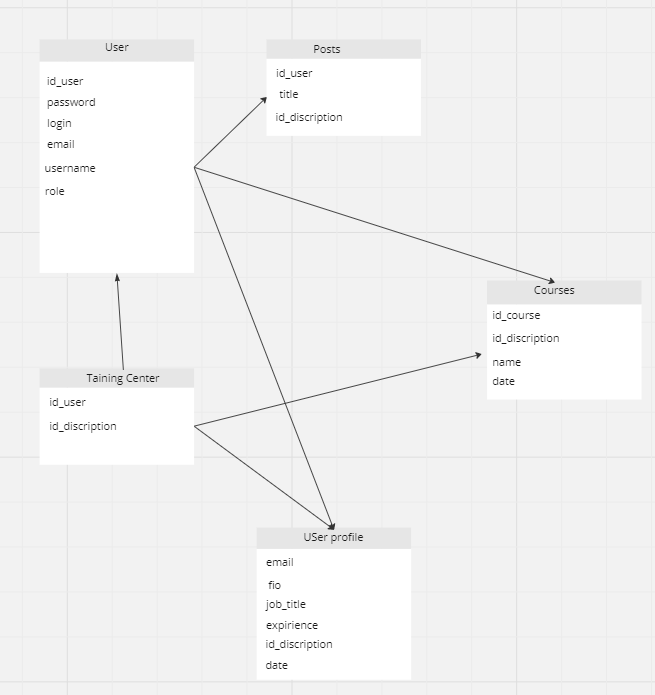
Цель второй фазы проектирования базы данных состоит в создании логической модели данных для исследуемой части предприятия.

Логическая модель, отражающая особенности представления о функционировании предприятия одновременно многих типов пользователей, называется глобальной логической моделью данных.

Процесс проектирования БД должен опираться на определенную модель данных (реляционная, сетевая, иерархическая), которая определяется типом предполагаемой для реализации информационной системы СУБД.

Концептуальное и логическое проектирование — это итеративные процессы, которые включают в себя ряд уточнений, продолжающиеся до тех пор, пока не будет получен наиболее соответствующий структуре предприятия продукт.

В данной БД все поля атомарны и нет одинаковых кортежей, отсутствуют зависимости неключевых полей от части составного ключа, неключевые атрибуты взаимно независимы. Исходя из этих особенностей, можно сделать вывод о том, что все отношения находятся в 3 нормальной форме. Логическая схема БД представлена на рисунке 3.2.

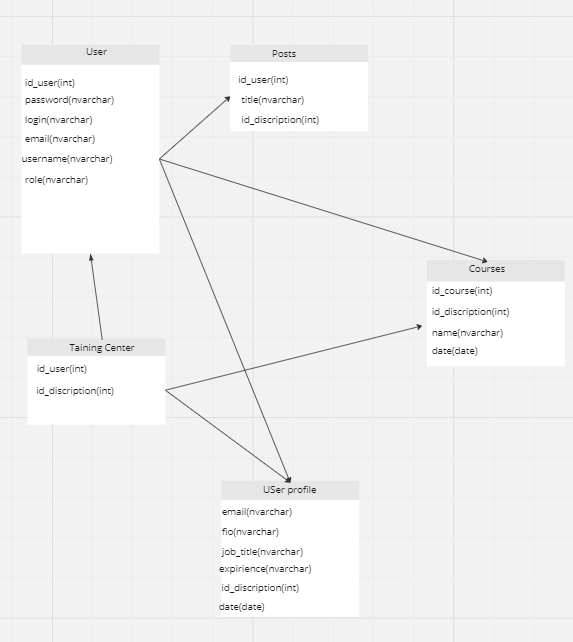


**Рисунок 3.2** — **Логическая модель БД**

Физическое проектирование базы данных – процесс подготовки описания реализации базы данных на вторичных запоминающих устройствах; на этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также все связанные с этим ограничения целостности и средства защиты.

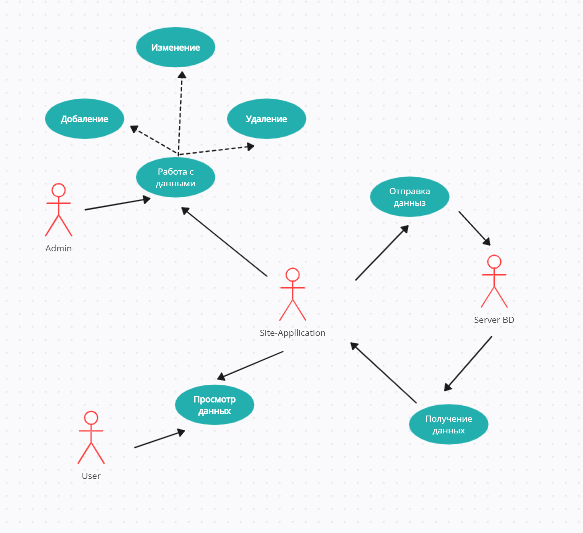
Как правило, основной целью физического проектирования базы данных является описание способа физической реализации логического проекта базы данных.

При построении физической модели каждая таблица приобрела идентификатор, и для доступа к любой таблице из одной БД нужно всего лишь добавить внешнее поле в таблицу. Физическая модель БД изображена на рисунке 3.3.



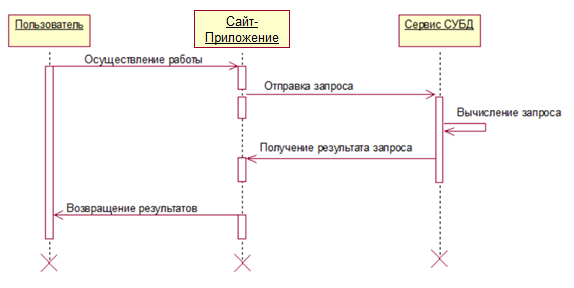
**Рисунок 3.3** — **Физическая модель БД**

Диаграмма Use Case (рисунок 3.4):



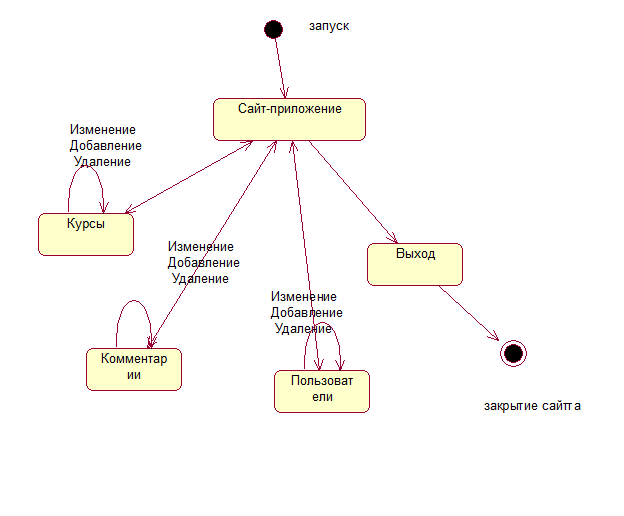
**Рисунок 3.4** — **Диаграмма Use Case**

Диаграмма Sequence (рисунок 3.5):



**Рисунок 3.5** — **Диаграмма Sequence**

Диаграмма Statechart (рисунок 3.6):



**Рисунок 3.6** — **Диаграмма StateChart**

# 4 РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА ДЛЯ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

Для создания вёрстки проекта используется HTML в связке с CSS.

HTML (HyperText Markup Language) – это специальный язык разметки, который применяется при создании сайтов в интернете. Браузеры прекрасно понимают html и могут интерпретировать в понятном для человека виде .

CSS (Cascading Style Sheets) — технология описания внешнего вида документа, написанного языком разметки . CSS отвечает за оформление и внешний вид, а HTML — за содержание и логическую структуру документа.

Так же для возможности открытия различных окон и работы графиков используется JS.

JavaScript — это кроссплатформенный, интерпретируемый язык программирования, который используется для создания интерактивных веб-страниц и приложений .

Для вывода самих задач и подзадач в HTML используется PHP в связке с фреймворком 1С-Битрикс.

Python —высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ.

Python Django —свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. Проект поддерживается организацией Django Software Foundation. Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми.

Windows Presentation Foundation — аналог WinForms, система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая подсистема в составе .NET Framework, использующая язык С# & XML.

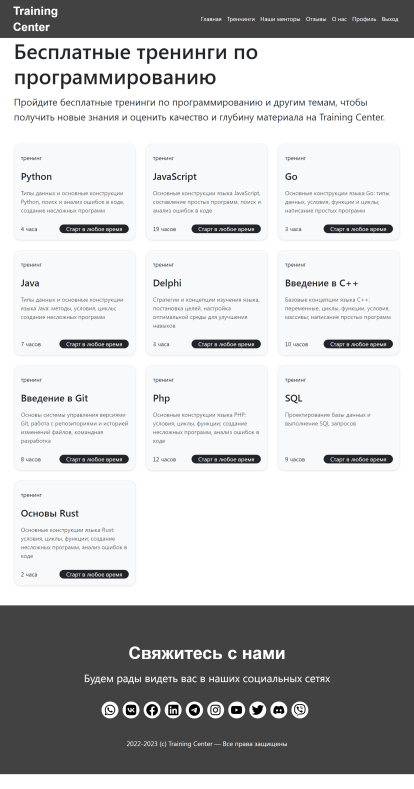
Информационные блоки - модуль, позволяющий каталогизировать и управлять различными типами (блоками) однородной информации. С помощью информационных блоков может быть реализована публикация различных типов динамической информации.

При запуске сайта пользователя встречает главная страница (рисунок 4.1).



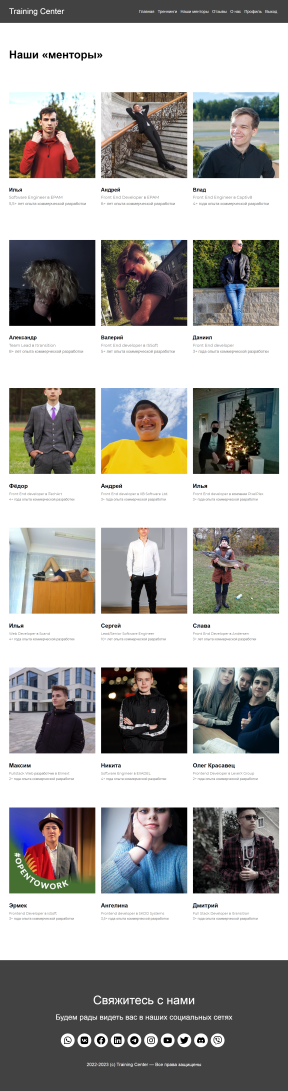
**Рисунок 4.1** — **Главная страница**

При просмотре главной страницы мы можем видеть главное меню сайта и его наполнение, разделы которых отображены ниже (рисунок 4.2).

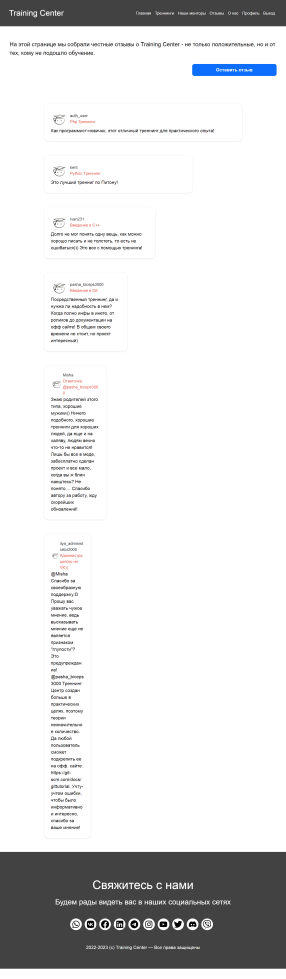


**Рисунок 4.2** — **«Тренннги»**

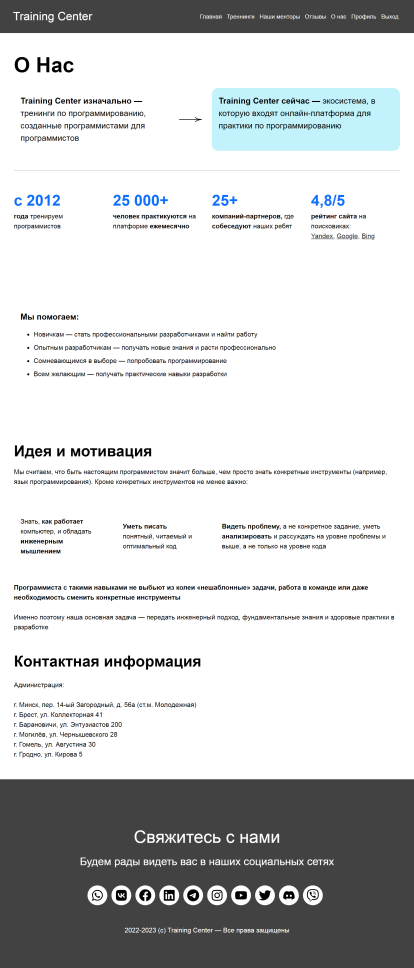
После вкладки «Треннинги» можно увидеть вкладки и других разделов (рисунок 4.3).



**Рисунок 4.3** — **«Наши менторы»**



**Рисунок 4.4** — «**Отзывы»**



**Рисунок 4.5** — **«О нас»**

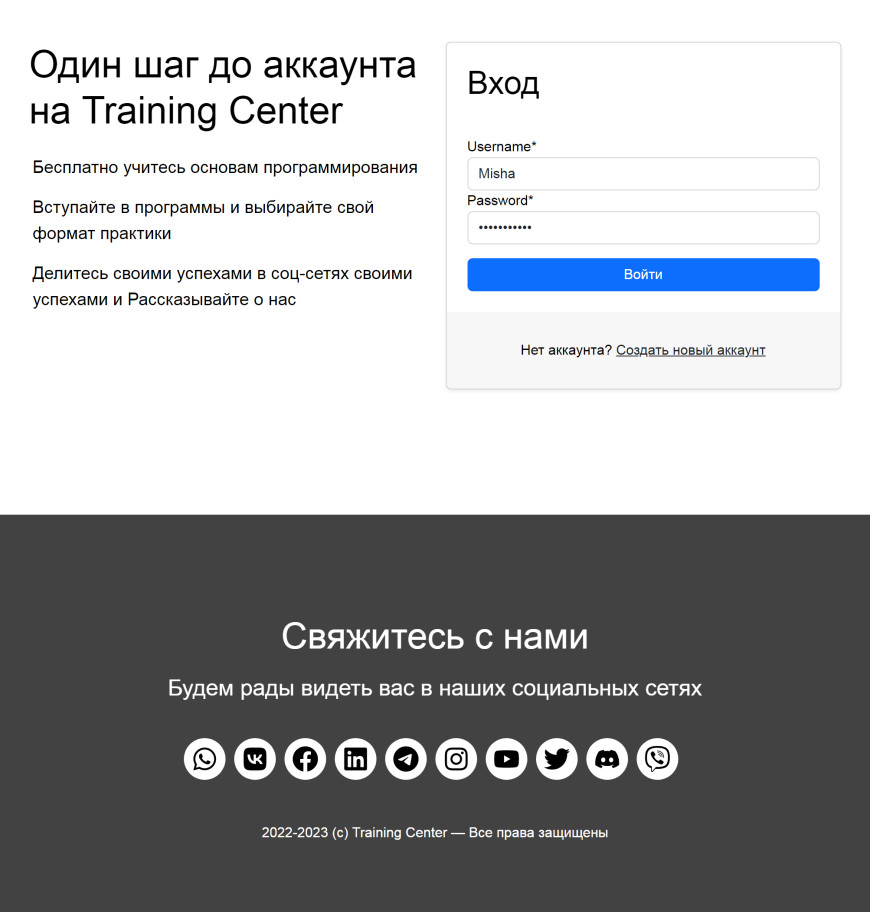
В шапке каждой из страниц присутствует кнопка «Профиль» (рисунок 4.6).



**Рисунок 4.6** — **Вкладка Профиль**

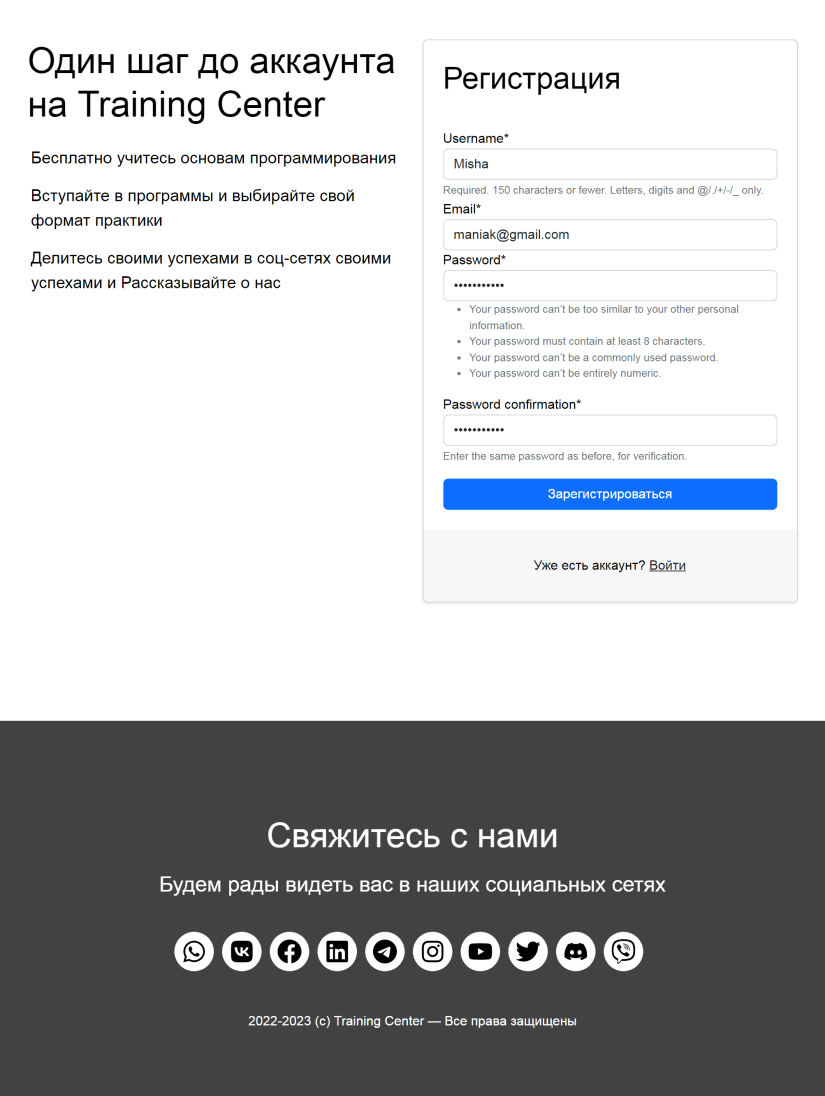
При нажатии на кнопку «Личный кабинет» мы переходим на страницу

проверки пользователя на авторизацию. Если пользователь не авторизирован, ему покажется окно входа в аккаунт или же вариант с регистрацией (рисунок 4.7).



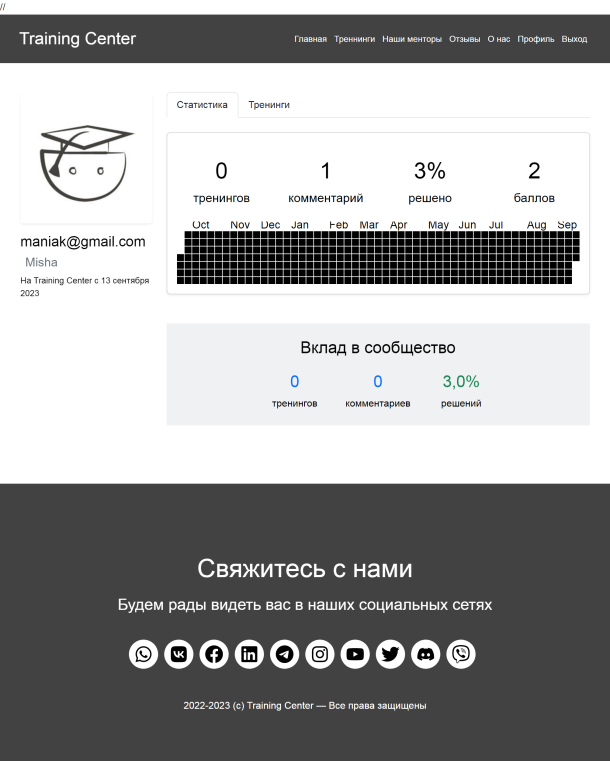
**Рисунок 4.7** — **Форма авторизации**

Страница регистрации пользователя (рисунок 4.8).

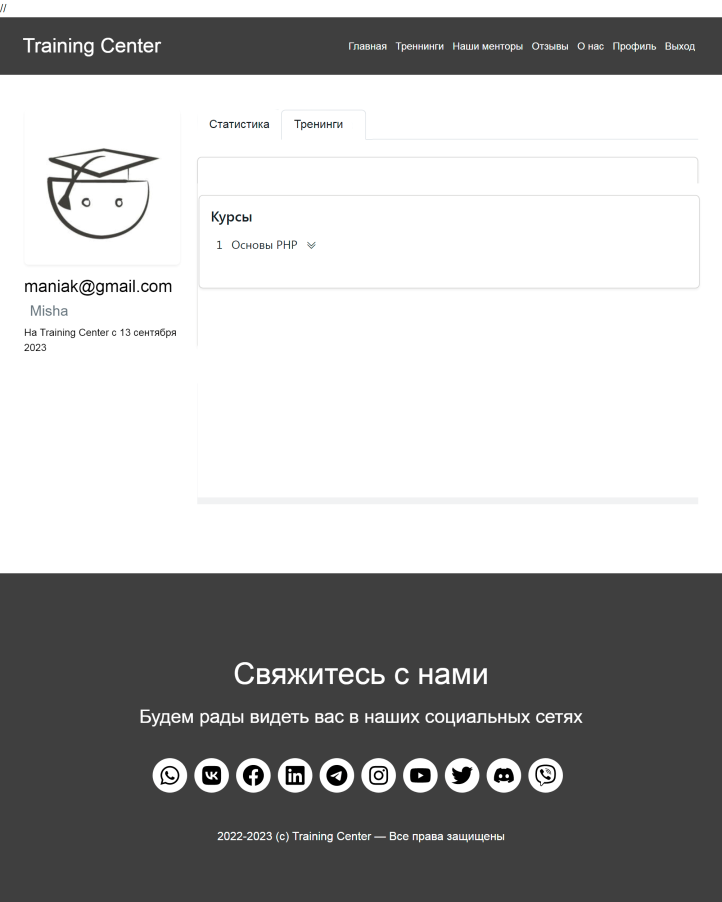
****

**Рисунок 4.8 —Форма регистрации**

После успешной авторизации мы попадаем в профиль, где можем увидеть информацию о пользователе, например: «Статистика», «Тренинги» где имеются доступные текущие практические задания (рисунок 4.9, 4.10).

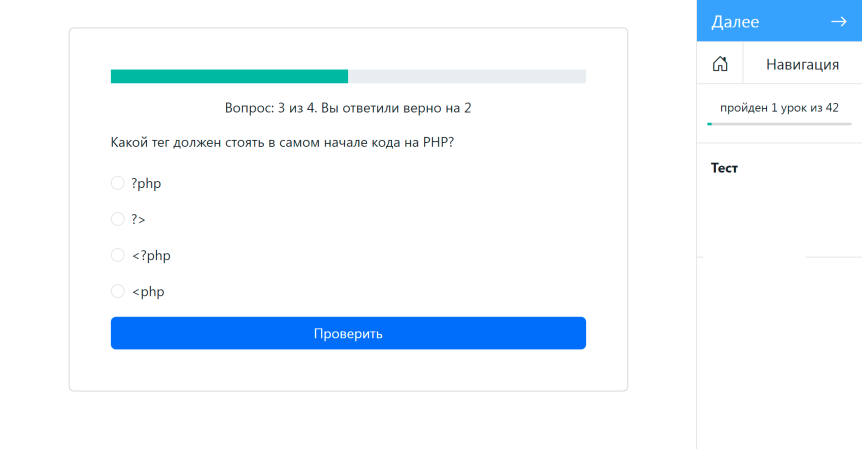


**Рисунок 4.9** — **Страница профиля-статистика**

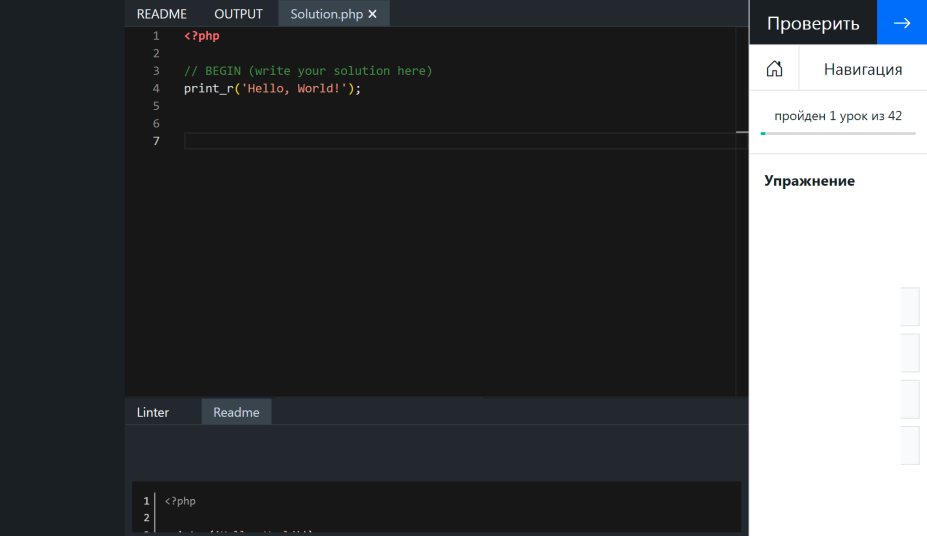


**Рисунок 4.10** — **Страница профиля-тренинги**

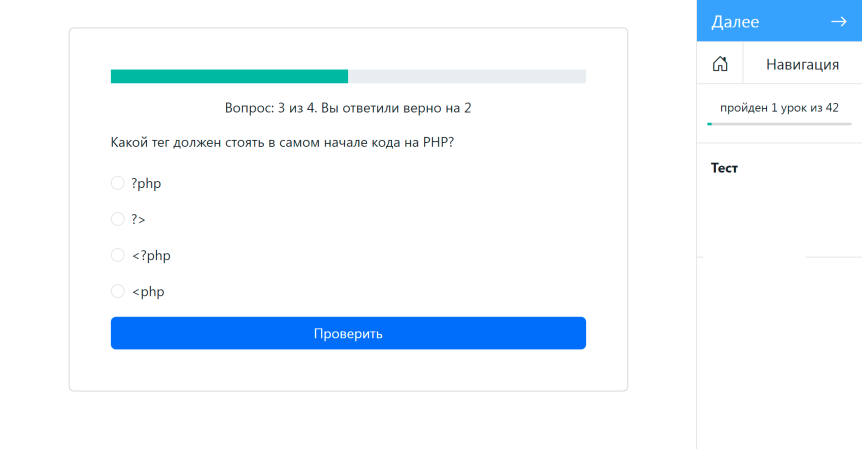
Прохождение практического тренинга с результатом прохождения (рисунок 4.11, 4.12, 4.13).



**Рисунок 4.11** — **Тренинг выбор варианта**

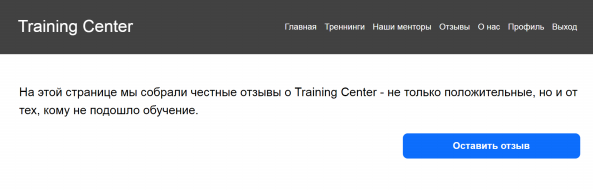


**Рисунок 4.12** — **Тренинг тест**

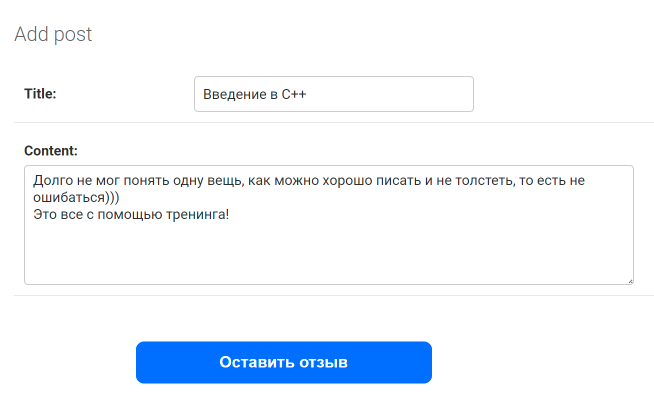


**Рисунок 4.13** — **Тренинг результат**

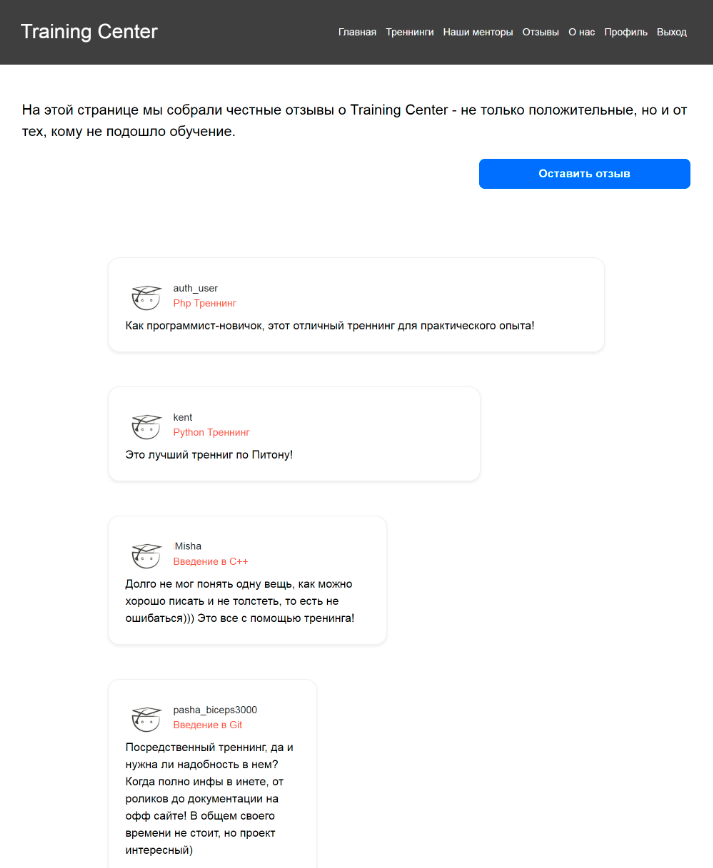
Возможность пользователю оставить комментарий, если он авторизован (рисунок 4.14, 4.15, 4.16).



**Рисунок 4.14** — **Кнопка «Оставить отзыв»**

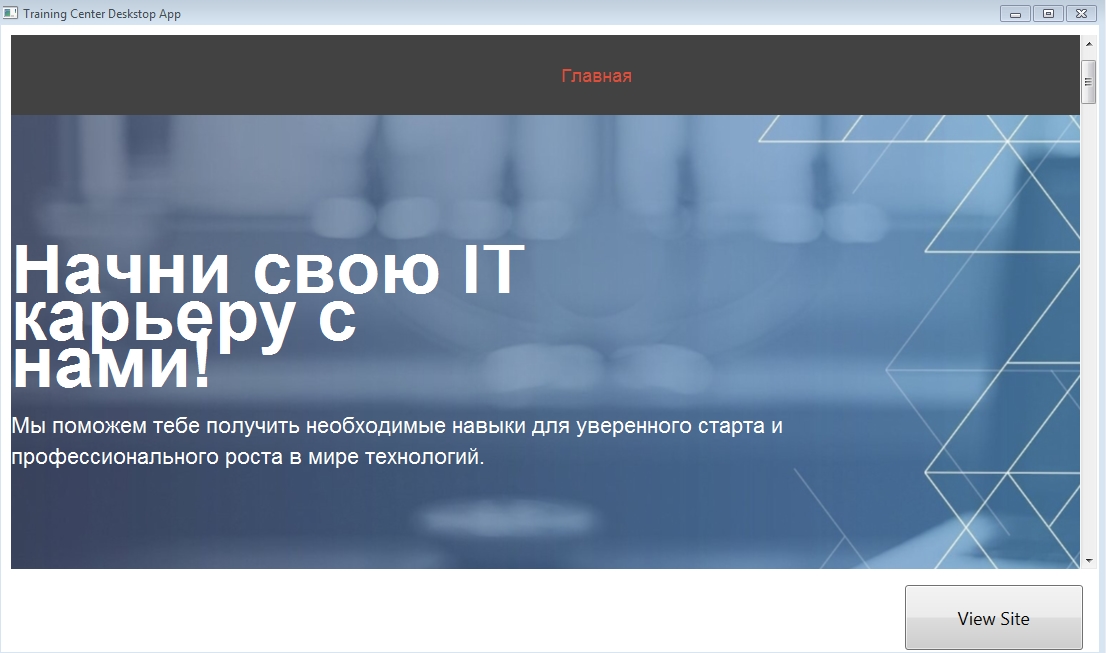


**Рисунок 4.15** — **Заполнение отзыва**

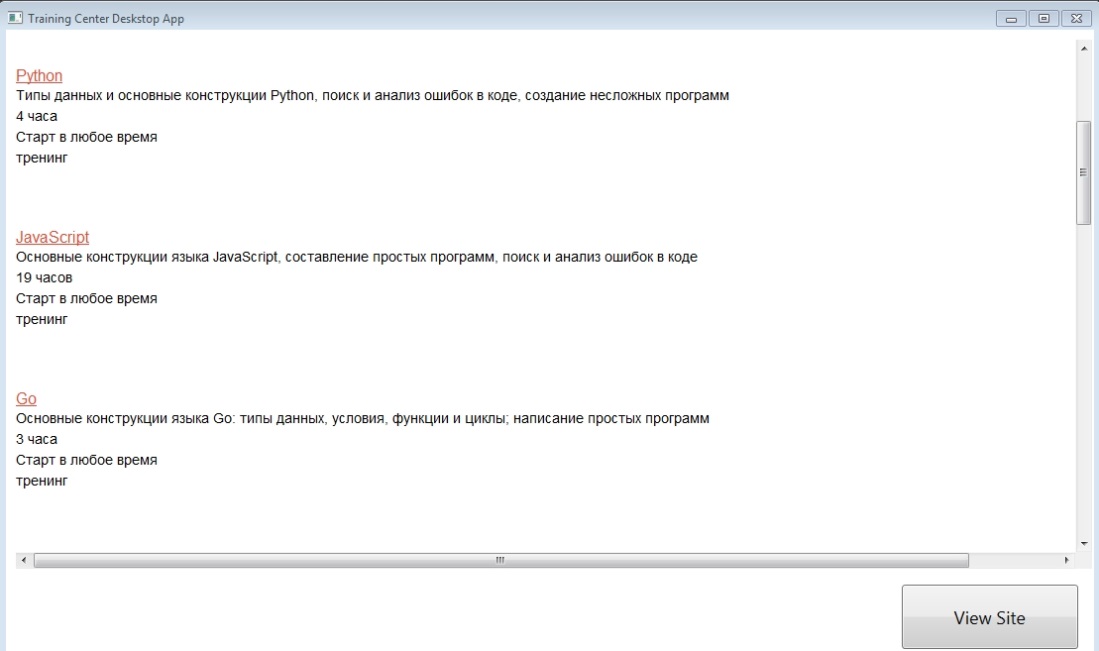


**Рисунок 4.16** — **Комментарий на странице «Отзывы»**

Также у пользователь есть возможность зайти с приложения, страница браузера транслируется прямиком в приложении (рисунок 4.17, 4.18).



**Рисунок 4.17** — **Приложение «Главное меню»**



**Рисунок 4.18** — **Приложение «Тренинги»**

# 5 ИСПОЛЬЗОВАНИЕ СЕТЕВЫХ ТЕХНОЛОГИЙ НА ПРЕДПРИЯТИИ

В контексте использования сетевых технологий на предприятии ЕПАМ, можно выделить несколько основных аспектов:

1) Коммуникации и связь: Сетевые технологии позволяют обеспечить эффективную коммуникацию и связь между сотрудниками ЕПАМ внутри организации и с внешними стейкхолдерами. Это может включать в себя использование электронной почты, программ для видеоконференций, систем для обмена документами и другие инструменты для коммуникации.

2) Сетевая инфраструктура и сервера: ЕПАМ, вероятно, будет иметь собственную сетевую инфраструктуру, состоящую из серверов, коммутаторов, маршрутизаторов и другого оборудования. Сетевые технологии используются для обеспечения надежной и безопасной работы инфраструктуры, а также для обеспечения доступа к ресурсам предприятия из разных местоположений.

3) Облачные технологии: Облачные технологии могут быть также важной составляющей сетевой инфраструктуры ЕПАМ. Они позволяют хранить и обрабатывать данные в облаке, что упрощает доступ к информации и сотрудничество между сотрудниками в разных локациях.

4) Безопасность сети: Сетевые технологии также играют важную роль в обеспечении безопасности предприятия ЕПАМ. Это может включать использование брэндмауэров, систем обнаружения вторжений, антивирусных программ и других механизмов, чтобы защитить сеть и данные от несанкционированного доступа.

5) Интеграция систем: ЕПАМ, вероятно, использует различные системы и платформы для управления бизнес-процессами и проектами. Сетевые технологии могут помочь в интеграции этих систем и обеспечении их взаимодействия, что способствует эффективной работе и снижению дублирования данных и усилий.

Все эти аспекты сетевых технологий на предприятии ЕПАМ помогают обеспечить эффективную коммуникацию, совместную работу, доступ к информации и защиту данных.

# 6 СПОСОБЫ И СРЕДСТВА ЗАЩИТЫ ИНФОРМАЦИИ

Для защиты информации Framework Django работает с встроенной базой данных SqLite и имеет свои встроенные механизмы защиты, например модули по защите от DdoS-атаки, вирусов, ботов, включая XSS и CSRF атаки.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Концепции проектирования БД [Электронный ресурс]. — Режим доступа: http://artemvolkov.ulcraft.com/liektsiia\_3 — Дата доступа: 10.09.2023
2. Физическая модель базы данных [Электронный ресурс]. — Режим доступа: https://clck.ru/35hier — Дата доступа: 10.09.2023
3. Язык программирования Python [Электронный ресурс]. — Режим доступа: https://clck.ru/35himV — Дата доступа: 11.09.2023
4. Сетевые технологии [Электронный ресурс]. — Режим доступа: clck.ru/34tEcz — Дата доступа: 11.09.2023
5. Технология WPF [Электронный ресурс]. — Режим доступа: clck.ru/35hixX — Дата доступа: 12.09.2023

# ПРИЛОЖЕНИЯ

**ПРИЛОЖЕНИЕ А**

**Код программы**

from django.shortcuts import render, redirect

from django.contrib import messages

from django.contrib.auth.decorators import login\_required

from .forms import UserRegisterForm

def register(request):

if request.method == 'POST':

form = UserRegisterForm(request.POST)

if form.is\_valid():

form.save()

username = form.cleaned\_data.get('username')

messages.success(request,f'Ваш аккаунт успешно создан, добро пожаловать')

return redirect('login')

else:

form = UserRegisterForm()

return render(request, 'userspage/SignUp.html', {'form': form})

@login\_required

def profile(request):

return render(request, 'userspage/Profile.html');

form.ShowInTaskbar = true;

Close();

}

}

}

class Profile(models.Model):

user = models.OneToOneField(User, on\_delete=models.CASCADE)

image = models.ImageField(default='default.jpg', upload\_to='profile\_pics')

def \_\_str\_\_(self):

return f'{self.user.username} Profile }

MessageBox.Show("Пользователь зарегистрирован");

}

urlpatterns = [

path('admin/', admin.site.urls),

path('', views.register, name='sign\_up'),

# path('my\_profile', views.profile, name='profile'),

]

class UserRegisterForm(UserCreationForm):

email = forms.EmailField()

class Meta:

model = User

fields = ['username', 'email', 'password1', 'password2']

**ПРИЛОЖЕНИЕ Б**

**Код программы**

<Application x:Class="desktop\_app.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="clr-namespace:desktop\_app"

StartupUri="MainWindow.xaml">

<Application.Resources>

</Application.Resources>

</Application><Window x:Class="desktop\_app.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:desktop\_app"

mc:Ignorable="d"

Title="Training Center Deskstop App" Height="665.298" Width="1113.596">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="100\*"/>

<ColumnDefinition Width="66\*"/>

<ColumnDefinition Width="381\*"/>

<ColumnDefinition Width="112\*"/>

<ColumnDefinition Width="275\*"/>

</Grid.ColumnDefinitions>

<WebBrowser HorizontalAlignment="Left" Height="534" VerticalAlignment="Top" Width="1086" Grid.ColumnSpan="5" Margin="10,10,0,0" Source="http://127.0.0.1:8000/training\_course"/>

<Button Content="View Site" Grid.Column="4" HorizontalAlignment="Left" Margin="129,560,0,0" VerticalAlignment="Top" Width="178" Height="65" FontSize="18"/>

</Grid>

</Window>

**ПРИЛОЖЕНИЕ В**

**Код программы**

namespace Practice

from django.shortcuts import render

# Create your views here.

def index(request):

return render(request, 'mainpage/TestPage.html')

def mainPage(request):

return render(request, 'mainpage/MainPage.html')

def aboutUs(request):

return render(request, 'mainpage/AboutUs.html')

def ourTrainers(request):

return render(request, 'mainpage/OurTrainers.html')

def trainingCourse(request):

return render(request, 'mainpage/TrainingCourse.html')

# Sign up and log in

def logIn(request):

return render(request, 'mainpage/LogIn.html')

def signUp(request):

return render(request, 'mainpage/SignUp.html')

def remindPassword(request):

return render(request, 'mainpage/RemindPassword.html')

urlpatterns = [

path('', views.mainPage, name='home'),

path('about\_us', views.aboutUs, name='about'),

path('our\_trainers', views.ourTrainers, name='trainers'),

# path('students\_reviews', views.studentsReviews, name='reviews'),

path('training\_course', views.trainingCourse, name='trainings'),

#path('log\_in', views.logIn, name='login'),

#path('sign\_up', views.signUp, name='sign\_up'),

#path('remind\_password', views.remindPassword, name='remind\_password'),

]

class MainpageConfig(AppConfig):

default\_auto\_field = 'django.db.models.BigAutoField'

name = 'mainpage'

"""

Django settings for training project.

Generated by 'django-admin startproject' using Django 3.2.21.

For more information on this file, see

https://docs.djangoproject.com/en/3.2/topics/settings/

For the full list of settings and their values, see

https://docs.djangoproject.com/en/3.2/ref/settings/

"""

from pathlib import Path

# Build paths inside the project like this: BASE\_DIR / 'subdir'.

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = 'django-insecure-%0e+#la!4b6pav8onwaye^l&\_o2@ehn87baw@ue%f5#i0fvuhi'

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED\_HOSTS = []

# Application definition

INSTALLED\_APPS = [

'mainpage.apps.MainpageConfig',

'reviewpage.apps.ReviewPageConfig',

'userspage.apps.UserspageConfig',

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

]

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware',

'django.contrib.sessions.middleware.SessionMiddleware',

'django.middleware.common.CommonMiddleware',

'django.middleware.csrf.CsrfViewMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware',

'django.contrib.messages.middleware.MessageMiddleware',

'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'training.urls'

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [],

'APP\_DIRS': True,

'OPTIONS': {

'context\_processors': [

'django.template.context\_processors.debug',

'django.template.context\_processors.request',

'django.contrib.auth.context\_processors.auth',

'django.contrib.messages.context\_processors.messages',

],

},

},

]

WSGI\_APPLICATION = 'training.wsgi.application'

# Database

# https://docs.djangoproject.com/en/3.2/ref/settings/#databases

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': BASE\_DIR / 'db.sqlite3',

}

}

# Password validation

# https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators

AUTH\_PASSWORD\_VALIDATORS = [

{

'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

},

]

# Internationalization

# https://docs.djangoproject.com/en/3.2/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_L10N = True

USE\_TZ = True

LOGIN\_REDIRECT\_URL = 'home'

LOGIN\_URL = 'login'

"""training URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:

https://docs.djangoproject.com/en/3.2/topics/http/urls/

Examples:

Function views

1. Add an import: from my\_app import views

2. Add a URL to urlpatterns: path('', views.home, name='home')

Class-based views

1. Add an import: from other\_app.views import Home

2. Add a URL to urlpatterns: path('', Home.as\_view(), name='home')

Including another URLconf

1. Import the include() function: from django.urls import include, path

2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))

"""

from django.contrib import admin

from django.urls import path, include

from userspage import views as user\_views

from django.contrib.auth import views as auth\_views

from django.conf import settings

from django.conf.urls.static import static

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('mainpage.urls')),

path('students\_reviews/', include('reviewpage.urls')),

path('sign\_up/', user\_views.register, name='sign\_up'),

path('profile/', user\_views.profile, name='profile'),

path('login/', auth\_views.LoginView.as\_view(template\_name='userspage/LogIn.html'), name='login'),

path('logout/', auth\_views.LogoutView.as\_view(template\_name='mainpage/MainPage.html'), name='home-logout'),

# path('sign\_up/', include('userspage.sign\_up'),

#path('sign\_up/', user\_views.register, name='sign\_up'),

]

if settings.DEBUG:

urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)