Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информатики

Отчет по лабораторной работе №3

Проектирование и разработка ПО на языке UML

Студенты гр. 953504, 953502:

Харкевич А.П., Плахотникова Ю.П.

Руководитель: Гриценко Н. Ю.

Минск 2022

|  |  |
| --- | --- |
| **СОДЕРЖАНИЕ** |  |
| [Постановка задачи............................................................................................](#page3) | | | [3](#page3) | |
| [Разработка функциональной модели ..............................................................](#page4) | | | [4](#page4) | |
| [Разработка модели данных ..............................................................................](#page5) | | | [5](#page5) | |
| [Проектирование приложения ..........................................................................](#page6) | | | [6](#page6) | |
| [Диаграммы классов..................................................................................](#page6) | [6](#page6) |
| [Диаграмма состояний ..............................................................................](#page7) | 7 |
| [Диаграмма активности.............................................................................](#page9) | [9](#page9) |
| Диаграмма компонентов........................................................................ | 11 |
| [Реализация приложения .................................................................................](#page13) | | | [13](#page13) | |
| [Список использованных источников ................................................................](#page14) | | | | [14](#page14) | |
| [ПРИЛОЖЕНИЕ А Иллюстрации работы программного средства .................](#page15) | | | | [15](#page15) | |
| [ПРИЛОЖЕНИЕ Б Фрагменты программного кода..........................................](#page17) | | | | 17 | |

1. **ПОСТАНОВКА ЗАДАЧИ**

Предметная область: **«Сервис по прогнозу спортивных данных»**.

Разработать программное средство, механизмы взаимодействия с базой данных и нейронной сетью.

Тип СУБД – Oracle.

Среда проектирования – Oracle SQL Developer, Visual Studio Code.

Среда разработки – Visual Studio Code

**2РАЗРАБОТКА ФУНКЦИОНАЛЬНОЙ МОДЕЛИ**

Функциональную модель предметной области представим в виде диаграммы вариантов использования в нотации UML, представляющей систему

* виде набора варианта использования и актеров, взаимодействующих с ними.
  + рамках предметной области можно выделить гостя и авторизованного пользователя.

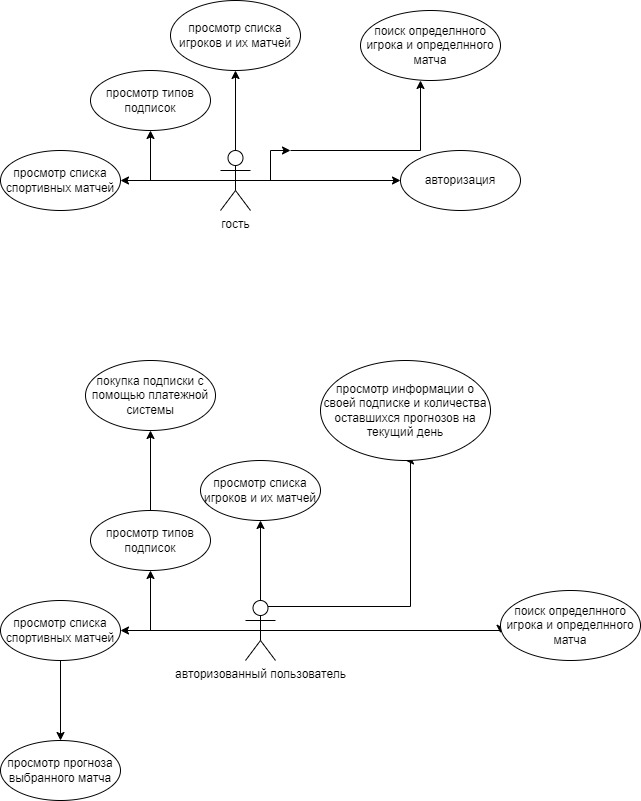
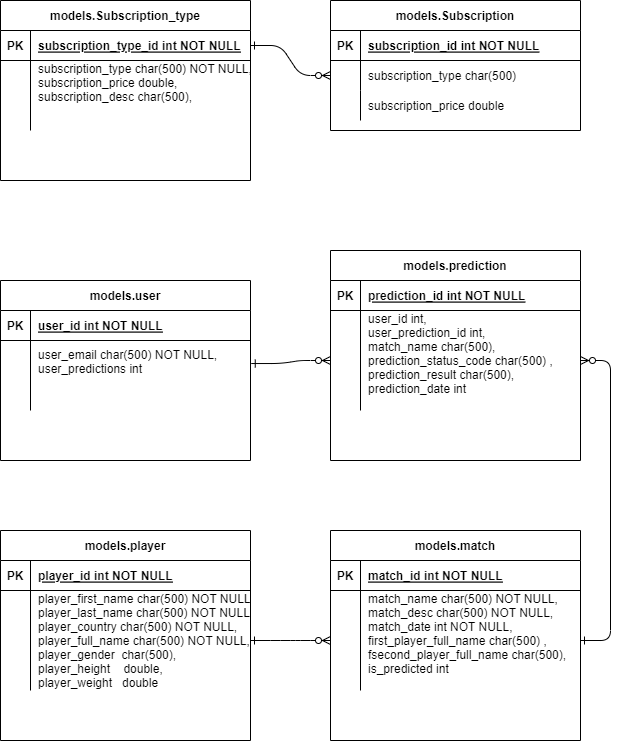


Рисунок 2.1 – Диаграмма вариантов использования

Согласно приведённой выше диаграмме у пользователя есть следующие возможности: вход/регистрация по почте, просмотр спортивных игроков и их ближайших матчей, просмотр всех ближайших матчей, поиск по игрокам и матчам, возможность просмотреть несколько типов подписок и выбрать наиболее подходящую, возможность спрогнозировать результат выбранного матча, возможность просмотреть информацию о своей подписке и количество оставшихся прогнозов текущего дня и возможность купить подписку.

1. **РАЗРАБОТКА МОДЕЛИ ДАННЫХ**

Разработка модели данных включает разработку классов, обеспечивающих взаимодействие с базой данных (рисунок 3.1).

 Рисунок 3.1 – Диаграмма классов модели данных

Модель данных включает классы, соответствующие сущностям базы данных, которые обеспечивают доступ и обновление данных получаемых из БД.

1. **ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ**

**4.1 Диаграммы классов**

Разработаем диаграммы классов, взаимодействия с пользователем.

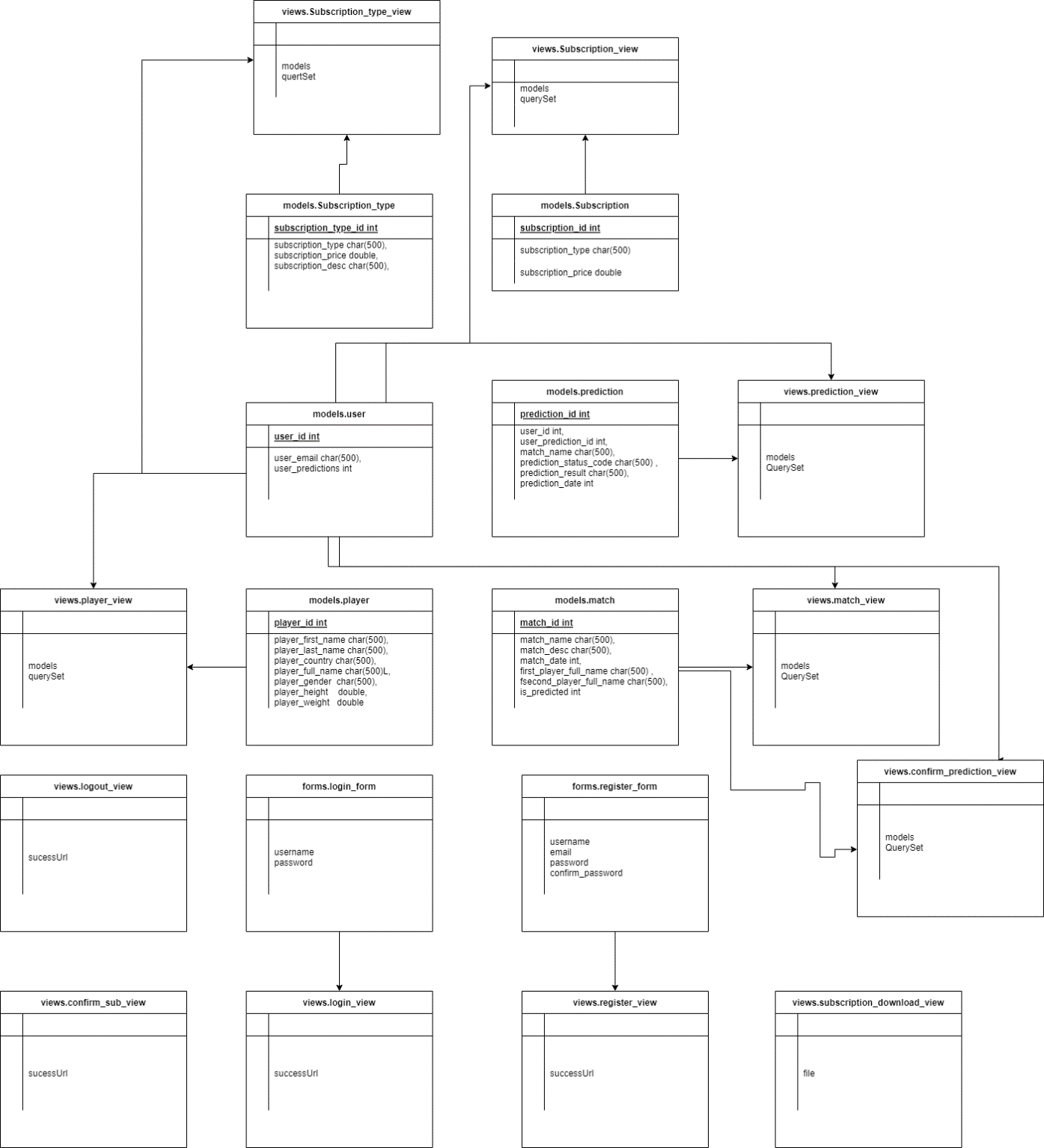
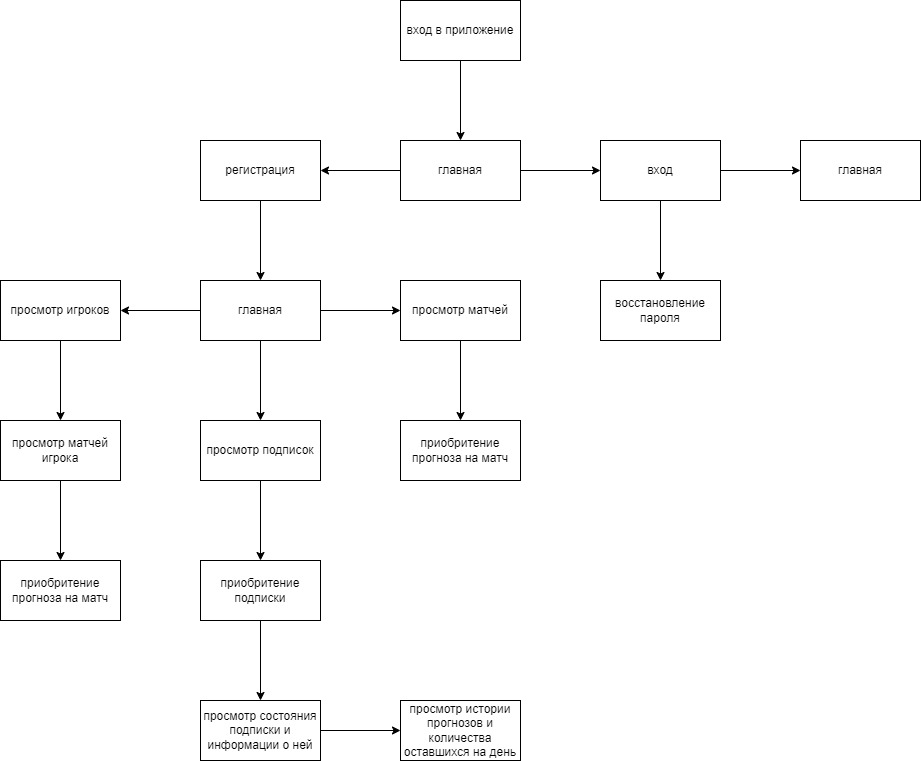


Рисунок 4.1 – Диаграмма классов для взаимодействия с пользователем приложения

**4.2 Диаграмма состояний**

Далее разработаем диаграмму состояний приложения (рисунок 4.2).

 Рисунок 4.2 – Диаграмма состояний приложения

Пользователь заходит в приложение для просмотра ближайших спортивных матчей или для покупки подписки на прогнозы по ближайшим спортивным матчам и переходит при этом на главную страницу.

Пользователь попадает на главную страницу, где видит описание приложения и список подписок, которые он может приобрести.

Он может перейти на вкладку матчи и посмотреть список ближайших матчей фильтруя их по дате и по названию. Он может перейти на вкладку спортсмены, отфильтровать спортсменов по имени. При нажатии на определенного спортсмена он перейдет на вкладку подробной информации о спортсмене, где будет указан список его ближайших матчей.

При нажатии на определенный матч он может узнать прогноз результата матча, для этого ему нужно авторизироваться и приобрести подписку.

Если пользователь не имеет аккаунта, он может зарегистрироваться, для этого ему достаточно нажать кнопку регистрации, ввести email, пароль и подтвердить email, после чего он перейдет на главную страницу.

Если он уже имеет аккаунт он может войти и перейдет после этого на главную страницу. Если пользователь забыл пароль он сможет восстановить его и войти.

Если пользователь не приобрел подписку и захочет узнать прогноз результата матча ему будет предложено перейти на главную страницу, для приобретения подписки.

При желании пользователя приобрести подписку, он будет переведен на модальное окно, где ему будет предложено ознакомиться с договором на подписку, подтвердить ознакомление с ним, выбрать платежную систему и ввести данные своей карты.

После приобретения своей подписки пользователь сможет увидеть свой тип подписок, а также число прогнозов, которые остались у него на текущие сутки. Теперь при нажатии на определённый матч пользователь сможет увидеть прогноз этого матча. Прогнозы результатов матча и их состояния он сможет отслеживать на вкладке прогнозы. Вкладка доступна только авторизованный пользователям.

**4.3 Диаграмма активности**

Разработаем диаграмму активности для варианта использования

“Открыть список матчей” и “Добавить прогноз” (рисунок 4.3).

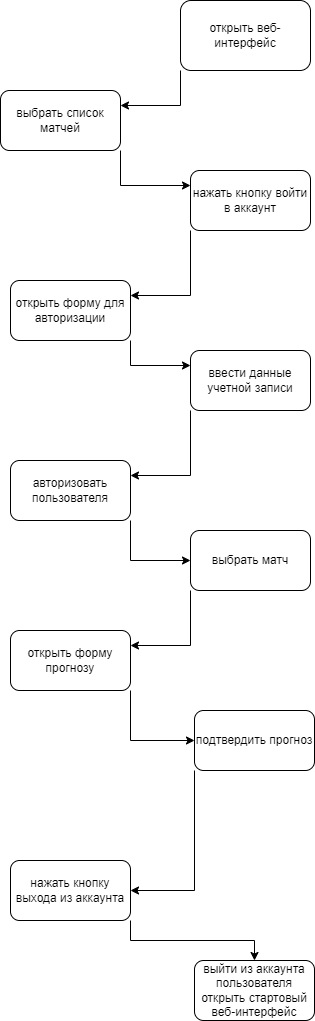


Рисунок 4.3 – Диаграмма активности варианта использования “Открыть список матчей” и “Добавить прогноз”

Активность предполагает состояние «Открыт веб-интерфейс» и включает следующие действия:

* Переход в список матчей
* Переход и заполнение формы входа в аккаунт
* Переход к прогнозу матча и подтверждение добавления прогноза
* Выход из аккаунта

**4.4 Диаграмма компонентов**

Разработаем диаграмму компонентов системы (рисунок 4.4) этапа разработки.

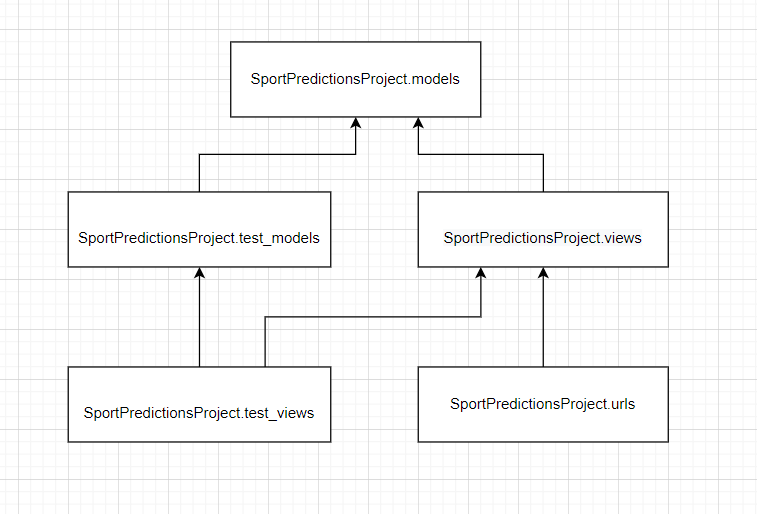


Рисунок 4.4 – Диаграмма компонентов

На диаграмме представлены следующие компоненты

* tests - компонента, содержащая ресурсы для тестирования приложения
* urls - компонента, является входной точкой в приложение, решающая зависимости путей сайта
* views - компонента, содержащая ресурсы для взаимодействия с UI приложения
* model- компонента, содержащая ресурсы для взаимодействия с БД приложения

1. **РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ**

Выполним разработку проекта приложения с использованием среды разработки Visual Studio Code.

Структура файлов проекта, разработанная в соответствии с классами, выявленными в ходе проектирования, приведена на рисунке 5.1.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5.1 – Проект приложения в Visual Studio Code

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

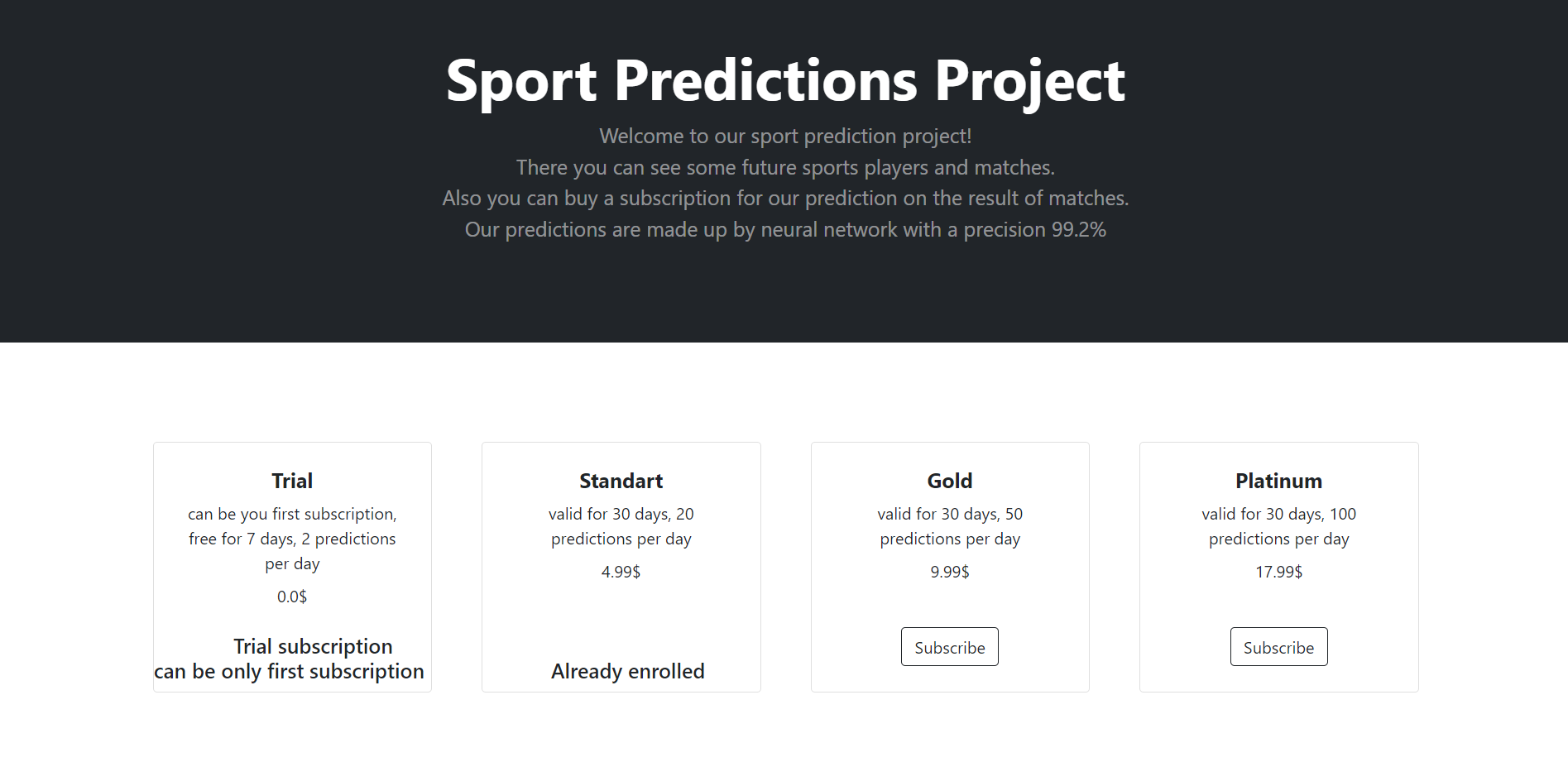
1. Документация Django [Электронный ресурс]

<https://www.djangoproject.com/>

1. Документация Bootstrap [Электронный ресурс] <https://getbootstrap.com/>
2. Документация Oracle [Электронный ресурс] https://docs.oracle.com/en/database/oracle/oracle-database/

**ПРИЛОЖЕНИЕ А**

**Иллюстрации работы программного средства**



Изображение выглядит как текст, снимок экрана, внутренний

Автоматически созданное описание

**ПРИЛОЖЕНИЕ Б**

**Фрагменты программного кода**

1. Листинг models.py

from django.db import models

import cx\_Oracle

from django.db import connection

def get\_subscription\_types():

cursor = connection.cursor()

ref\_cursor = cursor.var(cx\_Oracle.CURSOR).var

result = cursor.callproc("SP\_GET\_SUBSCRIPTION\_TYPES", (2, ref\_cursor))

raw\_context = result[1].fetchall()

subscription\_ids = []

subscription\_types = []

subscription\_descs = []

subscription\_prices = []

for i in range(len(raw\_context)):

subscription\_ids.append(raw\_context[i][0])

subscription\_types.append(raw\_context[i][1])

subscription\_descs.append(raw\_context[i][2])

subscription\_prices.append(raw\_context[i][3])

return subscription\_ids, subscription\_types, subscription\_descs, subscription\_prices

def get\_players():

cursor = connection.cursor()

ref\_cursor = cursor.var(cx\_Oracle.CURSOR).var

result = cursor.callproc("SP\_GET\_PLAYERS", (2, ref\_cursor))

raw\_context = result[1].fetchall()

return raw\_context

def get\_matches(user\_id):

cursor = connection.cursor()

ref\_cursor = cursor.var(cx\_Oracle.CURSOR).var

result = cursor.callproc("SP\_GET\_MATCHES", (user\_id, ref\_cursor))

raw\_context = result[1].fetchall()

return raw\_context

def get\_current\_user\_email(user\_id):

cursor = connection.cursor()

ref\_cursor = cursor.var(cx\_Oracle.CURSOR).var

result = cursor.callproc("SP\_GET\_USER\_EMAIL", (user\_id, ref\_cursor))

raw\_context = result[1].fetchall()

return raw\_context

def get\_predictions(user\_id):

cursor = connection.cursor()

ref\_cursor = cursor.var(cx\_Oracle.CURSOR).var

result = cursor.callproc("SP\_GET\_PREDICTIONS", (user\_id, ref\_cursor))

raw\_context = result[1].fetchall()

return raw\_context

class TSubscriptionType(models.Model):

type\_id = models.FloatField(primary\_key=True)

type\_name = models.CharField(max\_length=500, blank=True, null=True)

type\_price = models.DecimalField(max\_digits=5, decimal\_places=2, blank=True, null=True)

class Meta:

managed = True

db\_table = 't\_subscription\_type'

def get\_subscription\_types\_by\_id(id):

cursor = connection.cursor()

ref\_cursor = cursor.var(cx\_Oracle.CURSOR).var

result = cursor.callproc("SP\_GET\_SUBSCRIPTION\_TYPES\_BY\_ID", (id, ref\_cursor))

raw\_context = result[1].fetchall()

subscription\_ids = []

subscription\_types = []

subscription\_prices = []

for i in range(len(raw\_context)):

subscription\_ids.append(raw\_context[i][0])

subscription\_types.append(raw\_context[i][1])

subscription\_prices.append(raw\_context[i][2])

return subscription\_ids, subscription\_types, subscription\_prices

def insert\_new\_user\_sub(id, user\_id):

print(user\_id)

cursor = connection.cursor()

days = 0

if id == 1:

days = 7

else:

days = 30

result = cursor.callproc("SP\_LOAD\_NEW\_SUBSCRIPTION", (user\_id, id, days))

return 'success'

def get\_matches\_by\_name(name):

cursor = connection.cursor()

ref\_cursor = cursor.var(cx\_Oracle.CURSOR).var

result = cursor.callproc("SP\_GET\_MATCHES\_BY\_NAME", (name, ref\_cursor))

raw\_context = result[1].fetchall()

return raw\_context

def insert\_new\_user\_prediction(name, user\_id):

cursor = connection.cursor()

result = cursor.callproc("SP\_LOAD\_NEW\_USER\_PREDICTION", (name, user\_id))

return 'success'

def get\_subs\_analysis():

cursor = connection.cursor()

ref\_cursor = cursor.var(cx\_Oracle.CURSOR).var

result = cursor.callproc("SP\_GET\_SUBSCRIPTION\_ANALYSIS", (2, ref\_cursor))

raw\_context = result[1].fetchall()

return raw\_context

def get\_current\_user\_predictions\_amount(user\_id):

cursor = connection.cursor()

ref\_cursor = cursor.var(cx\_Oracle.CURSOR).var

result = cursor.callproc("SP\_GET\_CURRENT\_USER\_PREDICTIONS\_AMOUNT", (user\_id, ref\_cursor))

raw\_context = result[1].fetchall()

return raw\_context

logger.info('Profile saved')

def get\_current\_user\_subscription(user\_id):

cursor = connection.cursor()

ref\_cursor = cursor.var(cx\_Oracle.CURSOR).var

result = cursor.callproc("SP\_GET\_CURRENT\_USER\_SUBSCRIPTION", (user\_id, ref\_cursor))

raw\_context = result[1].fetchall()

return raw\_context

def get\_matches\_by\_player\_id(player\_id):

cursor = connection.cursor()

ref\_cursor = cursor.var(cx\_Oracle.CURSOR).var

result = cursor.callproc("SP\_GET\_MATCHES\_BY\_PLAYER\_ID", (player\_id, ref\_cursor))

raw\_context = result[1].fetchall()

return raw\_context

def get\_players\_by\_id(player\_id):

cursor = connection.cursor()

ref\_cursor = cursor.var(cx\_Oracle.CURSOR).var

result = cursor.callproc("SP\_GET\_PLAYERS\_BY\_ID", (player\_id, ref\_cursor))

raw\_context = result[1].fetchall()

return raw\_context

1. Листинг views.py

from django.shortcuts import redirect, render

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

from django.http import HttpResponseRedirect

from django.contrib.auth import authenticate, login, logout

from myapp.models import get\_subscription\_types, get\_players, get\_matches, get\_current\_user\_email, get\_predictions, get\_subscription\_types\_by\_id, insert\_new\_user\_sub, get\_matches\_by\_name, insert\_new\_user\_prediction,get\_subs\_analysis, get\_current\_user\_predictions\_amount, get\_current\_user\_subscription, get\_matches\_by\_player\_id, get\_players\_by\_id

from django.contrib.auth.models import User

def test\_view(request):

subscription\_ids, subscription\_types, subscription\_descs, subscription\_prices = get\_subscription\_types()

subscriptions = []

for i in range(len(subscription\_ids)):

subscriptions.append({ 'subscription\_id': subscription\_ids[i],

'subscription\_type' : subscription\_types[i],

'subscription\_price': subscription\_prices[i],

'subscription\_desc': subscription\_descs[i]})

raw\_user\_context = get\_current\_user\_email(request.user.id)

user ={}

for i in range(len(raw\_user\_context)):

user['id'] = raw\_user\_context[i][0]

user['email'] = raw\_user\_context[i][1]

raw\_user\_context = get\_current\_user\_predictions\_amount(request.user.id)

raw\_sub\_context = get\_current\_user\_subscription(request.user.id)

subscription\_id = raw\_sub\_context[0][0]

user['predictions'] = raw\_user\_context[0][0]

context = { 'subscriptions' : subscriptions,

'user' : user,

'subscription\_id': subscription\_id}

print(context)

return render(request, 'base.html', context)

def players\_view(request):

raw\_context = get\_players()

players = []

for i in range(len(raw\_context)):

players.append({ 'player\_id': raw\_context[i][0],

'player\_first\_name': raw\_context[i][1],

'player\_last\_name': raw\_context[i][2],

'player\_full\_name': raw\_context[i][3],

'player\_country': raw\_context[i][4],

'player\_gender': raw\_context[i][5],

'player\_height': raw\_context[i][6],

'player\_weight': raw\_context[i][7]

})

raw\_user\_context = get\_current\_user\_email(request.user.id)

user ={}

for i in range(len(raw\_user\_context)):

user['id'] = raw\_user\_context[i][0]

user['email'] = raw\_user\_context[i][1]

raw\_user\_context = get\_current\_user\_predictions\_amount(request.user.id)

user['predictions'] = raw\_user\_context[0][0]

context = { 'players' : players,

'user' : user }

return render(request, 'players.html', context)

def matches\_view(request):

raw\_context = get\_matches(request.user.id)

matches = []

for i in range(len(raw\_context)):

matches.append({ 'match\_id': raw\_context[i][0],

'match\_name': raw\_context[i][1],

'match\_desc': raw\_context[i][2],

'match\_date': raw\_context[i][3],

'first\_player\_full\_name': raw\_context[i][4],

'second\_player\_full\_name': raw\_context[i][5],

'is\_predicted': raw\_context[i][6]

})

raw\_user\_context = get\_current\_user\_email(request.user.id)

user ={}

for i in range(len(raw\_user\_context)):

user['id'] = raw\_user\_context[i][0]

user['email'] = raw\_user\_context[i][1]

raw\_user\_context = get\_current\_user\_predictions\_amount(request.user.id)

user['predictions'] = raw\_user\_context[0][0]

context = { 'matches' : matches,

'user' : user }

return render(request, 'matches.html', context)

@login\_required(login\_url='/')

def predictions\_view(request):

raw\_context = get\_predictions(request.user.id)

predictions = []

for i in range(len(raw\_context)):

predictions.append({ 'prediction\_id': raw\_context[i][0],

'user\_id': raw\_context[i][1],

'user\_prediction\_id': raw\_context[i][2],

'match\_name': raw\_context[i][3],

'prediction\_status\_code': raw\_context[i][4],

'prediction\_result': raw\_context[i][5],

'prediction\_date': raw\_context[i][6]

})

raw\_user\_context = get\_current\_user\_email(request.user.id)

user ={}

for i in range(len(raw\_user\_context)):

user['id'] = raw\_user\_context[i][0]

user['email'] = raw\_user\_context[i][1]

raw\_user\_context = get\_current\_user\_predictions\_amount(request.user.id)

user['predictions'] = raw\_user\_context[0][0]

isDisplay = 0

if len(predictions) > 0:

isDisplay = 1

context = { 'predictions' : predictions,

'user' : user,

'isDisplay': isDisplay}

return render(request, 'predictions.html', context)

def logout\_view(request):

logout(request)

return HttpResponseRedirect('/')

def login\_operation\_view(request):

print(request.POST)

username = request.POST['username']

password = request.POST['password']

user = authenticate(request, username=username, password=password)

if user is not None:

login(request, user)

return HttpResponseRedirect('/')

def login\_view(request):

return render(request, 'login.html', {})

def register\_view(request):

return render(request, 'register.html', {})

def register\_operation\_view(request):

print(request.POST)

username = request.POST['username']

email = request.POST['email']

password = request.POST['password']

confirm\_password = request.POST['confirm\_password']

if password == confirm\_password:

user = User.objects.create\_user(username, email, password)

user = authenticate(request, username=username, password=password)

if user is not None:

login(request, user)

return HttpResponseRedirect('/')

else:

return HttpResponseRedirect('/')

def subscriptions\_view(request, id):

subscription\_ids, subscription\_types, subscription\_prices = get\_subscription\_types\_by\_id(id)

subscriptions = []

for i in range(len(subscription\_ids)):

subscriptions.append({ 'subscription\_id': subscription\_ids[i],

'subscription\_type': subscription\_types[i],

'subscription\_price': subscription\_prices[i]})

raw\_user\_context = get\_current\_user\_email(request.user.id)

user ={}

for i in range(len(raw\_user\_context)):

user['id'] = raw\_user\_context[i][0]

user['email'] = raw\_user\_context[i][1]

raw\_user\_context = get\_current\_user\_predictions\_amount(request.user.id)

user['predictions'] = raw\_user\_context[0][0]

context = { 'subscriptions' : subscriptions,

'user' : user }

return render(request, 'subscriptions.html', context)

def confirm\_sub\_view(request, id):

result = insert\_new\_user\_sub(id, request.user.id)

return HttpResponseRedirect('/')

def prediction\_confirm\_view(request, name):

raw\_context = get\_matches\_by\_name(name)

matches = []

for i in range(len(raw\_context)):

matches.append({ 'match\_id': raw\_context[i][0],

'match\_name': raw\_context[i][1],

'match\_desc': raw\_context[i][2],

'match\_date': raw\_context[i][3],

'first\_player\_full\_name': raw\_context[i][4],

'second\_player\_full\_name': raw\_context[i][5]

})

raw\_user\_context = get\_current\_user\_email(request.user.id)

user ={}

for i in range(len(raw\_user\_context)):

user['id'] = raw\_user\_context[i][0]

user['email'] = raw\_user\_context[i][1]

raw\_user\_context = get\_current\_user\_predictions\_amount(request.user.id)

user['predictions'] = raw\_user\_context[0][0]

context = { 'matches' : matches,

'user' : user }

return render(request, 'prediction\_confirm.html', context)

def prediction\_confirm\_operation\_view(request, name):

result = insert\_new\_user\_prediction(name, request.user.id)

return HttpResponseRedirect('/')

def analysis\_view(request):

raw\_context = get\_subs\_analysis()

print(raw\_context)

return render(request, 'analysis.html', {'values': raw\_context})

def matches\_by\_player\_view(request, id):

raw\_context = get\_matches\_by\_player\_id(id)

matches = []

for i in range(len(raw\_context)):

matches.append({ 'match\_id': raw\_context[i][0],

'match\_name': raw\_context[i][1],

'match\_desc': raw\_context[i][2],

'match\_date': raw\_context[i][3],

'first\_player\_full\_name': raw\_context[i][4],

'second\_player\_full\_name': raw\_context[i][5],

'player\_full\_name': raw\_context[i][6]

})

raw\_user\_context = get\_current\_user\_email(request.user.id)

user ={}

for i in range(len(raw\_user\_context)):

user['id'] = raw\_user\_context[i][0]

user['email'] = raw\_user\_context[i][1]

raw\_user\_context = get\_current\_user\_predictions\_amount(request.user.id)

user['predictions'] = raw\_user\_context[0][0]

raw\_player\_context = get\_players\_by\_id(id)

player ={}

player['full\_name'] = raw\_player\_context[0][0]

isDisplay = 0

if len(matches) > 0:

isDisplay = 1

context = { 'matches' : matches,

'user' : user,

'player': player,

'isDisplay': isDisplay}

return render(request, 'players\_by\_id.html', context)