

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«Дальневосточный федеральный университет»**

**ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ**

**Департамент программной инженерии и искусственного интеллекта**

**РАЗРАБОТКА ONLINE-СЕРВИСА ДЛЯ ПРЕДМЕТНОЙ ОБЛАСТИ  
«ОРГАНИЗАЦИЯ ТУРНИРОВ ПО СЁГИ»**

ПРОЕКТ

по дисциплине «Сетевые и интернет технологии»

по образовательной программе подготовки бакалавров по направлению

09.03.04 «Программная инженерия»

Выполнил:

студент гр. Б9120-09.03.04прогин

Заяц А.Е.

Руководитель:

доцент департамента ПИиИИ, к. т. н., доцент

Озерова Г. П.

г. Владивосток

2024

Оглавление

[Введение 3](#_Toc156644719)

[1 Проектирование online-сервиса 4](#_Toc156644720)

[1.1 Описание предметной области 4](#_Toc156644721)

[1.2 Исследования: метод персонажа 4](#_Toc156644722)

[1.3 Сценарии использования online-сервиса 4](#_Toc156644723)

[1.3.1 Userflow для персонажа организатор турнира 4](#_Toc156644724)

[1.4 Прототипы интерфейса 10](#_Toc156644725)

[1.5 Проектирование базы данных 14](#_Toc156644726)

[1.5.1 Концептуальная модель данных 14](#_Toc156644727)

[1.5.2 Логическая модель данных 15](#_Toc156644728)

[1.5.3 Описание процессов обработки информации 16](#_Toc156644729)

[2 Реализация online-сервиса 18](#_Toc156644730)

[2.1 Функциональные возможности 18](#_Toc156644731)

[2.2 Выбор средств реализации 18](#_Toc156644732)

[Заключение 19](#_Toc156644733)

[ПРИЛОЖЕНИЕ А 20](#_Toc156644734)

**Введение**

В современном мире online-сервисы используются повсеместно. Online-сервисы – это сайты, которые предоставляют всевозможные услуги, что значительно облегчает работу и позволяет существенно сэкономить время.

Однако, не все сферы в достаточной мере автоматизированы. Так, например, аренда игровых приставок зачастую ведётся в бумажном варианте, это занимает довольно продолжительное время. Также проверка доступности конкретной приставки занимает много времени.

В связи с чем вышеописанными проблемами было решено создать систему, которая бы обеспечивала возможность аренды в сети игровых приставок.

Целью проекта является разработка online-сервиса для аренды игровых приставок.

Для достижения цели необходимо выполнить следующие задачи:

* описать предметную область «Аренда игровых приставок»;
* провести исследование методом персонажа предметной области «Аренда игровых приставок»;
* описать сценарии использования online-cервиса предметной области «Аренда игровых приставок»;
* разработать проект пользовательского интерфейса разрабатываемого online-сервиса;
* разработать проект базы данных;
* реализовать online-сервис для автоматизации предметной области «Аренда игровых приставок».

1. **Проектирование online-сервиса**
   1. **Описание предметной области**

Компания предоставляет услугу аренды игровых приставок.

Игровые приставки арендуют пользователя сайта. Каждый пользователь предоставляет следующие данные для аренды пристаки: имя, номер телефона.

У каждого пользователя есть счётчик количества арендованных игровых приставок на данный момент.

Игровая приставка имеет название, фирму производителя, цену аренды на один день и некоторое количество технических характеристик.

* 1. **Исследования: метод персонажа**

Администратор сайта:

* Характеристика:20 лет, мужчина, высшее образование, базовые навыки работы с компьютером
* Что делает? Добавляет пользователей на сайт, добавляет консоли на сайт, производит аренду игровой приставки конкретному пользователю.
  1. **Сценарии использования online-сервиса**
     1. **Userflow для персонажа администратор сайта**

1. Какая цель у администратора сайта при работе с системой?
   1. Администратор хочет добавить пользователя.

Обеспечить возможность добавление пользователя.

* 1. Администратор хочет добавить новую приставку на сайт.

Обеспечить возможность добавления приставок на сайт.

* 1. Администратор хочет изменить характеристику приставки.

Обеспечить возможность изменение характеристик приставки.

* 1. Администратор хочет иметь возможность поиска по приставкам.

Обеспечить возможность поиска по приставкам.

* 1. Администратор хочет выдавать в аренду приставку пользователю.

Обеспечить возможность записи аренды приставки пользователем.

1. Какие шаги должен предпринять Администратор для достижения целей при работе с системой?
   1. Администратор, который хочет добавить пользователя. Цель: добавить пользователя (Рис.1).
   2. Администратор, который хочет добавить приставку. Цель: добавить приставку (Рис.2).
   3. Администратор, который хочет выдать в аренду приставку пользователю. Цель: выдать в аренду приставку (Рис.3).
   4. Администратор, который хочет записать возврат приставки от пользователя. Цель: записать возврат (Рис.4).
   5. Администратор, который хочет совершить поиск по пользователям. Цель: найти пользователя (Рис.5).
   6. Администратор, который хочет совершить поиск по приставкам. Цель: найти приставку (Рис.6).

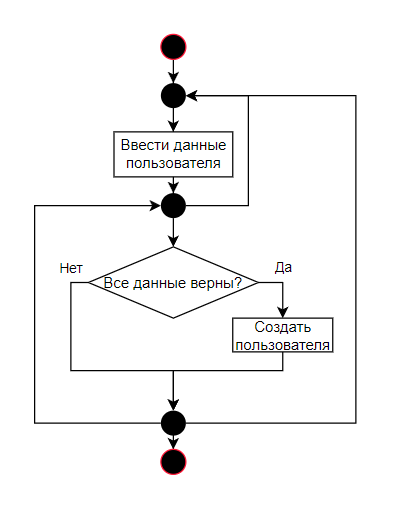


Рисунок 1 – Userflow диаграмма для п. 2.1 Администратора

Изображение выглядит как текст, диаграмма, линия

Автоматически созданное описание

Рисунок 2 – Userflow диаграмма для п. 2.2 Администратора

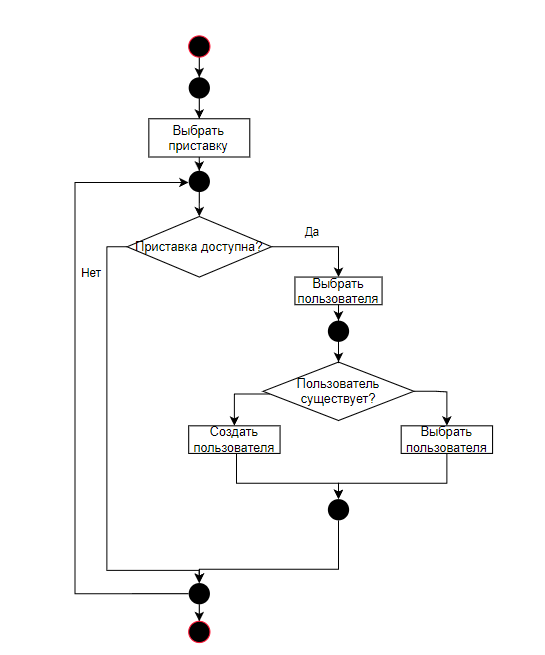


Рисунок 3 – Userflow диаграмма для п. 2.3 Администратора

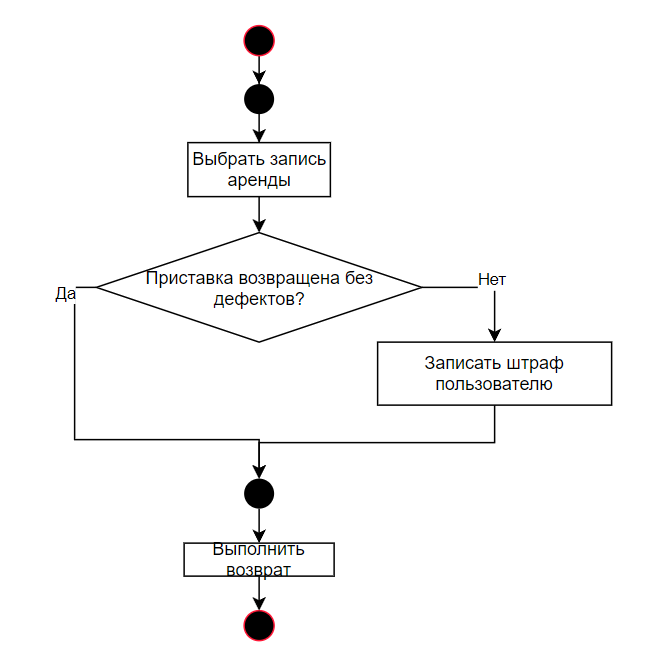


Рисунок 4 – Userflow диаграмма для п. 2.4 Администратора

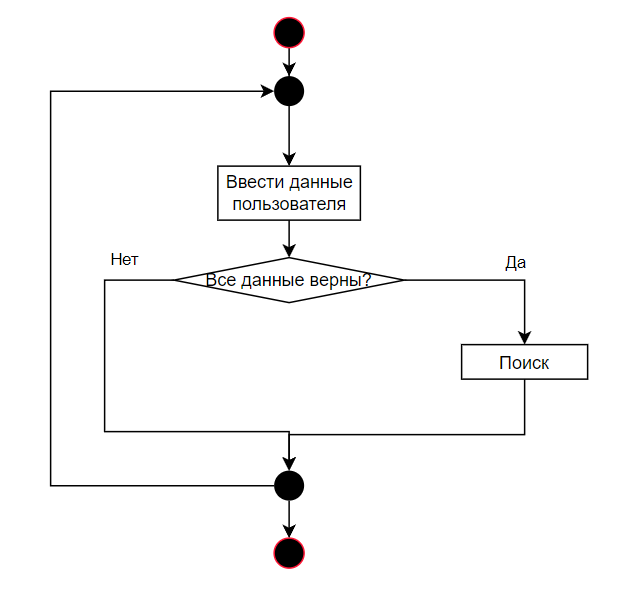


Рисунок 5 – Userflow диаграмма для п. 2.5 Администратора

Изображение выглядит как текст, диаграмма, линия, зарисовка

Автоматически созданное описание

Рисунок 6 – Userflow диаграмма для п. 2.6 Администратора

* 1. **Прототипы интерфейса**

Рассматривая User-flow персонажа «Администратор сайта», выделим все необходимые функции:

1. Отображение списка пользователей;
2. Отображение списка приставок;
3. Поиск по приставкам;
4. Поиск по пользователям;
5. Регистрация пользователя;
6. Выдача приставки в аренду;
7. Возврат приставки от пользователя.

Спроектируем прототип интерфейса (Рис. 9-15).

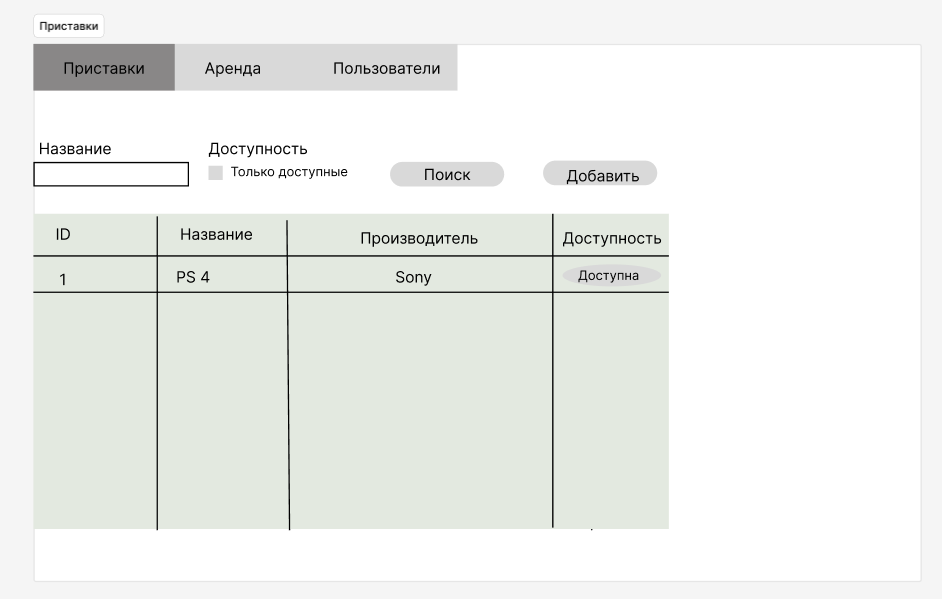


Рисунок 9 – Прототип окна списка приставок

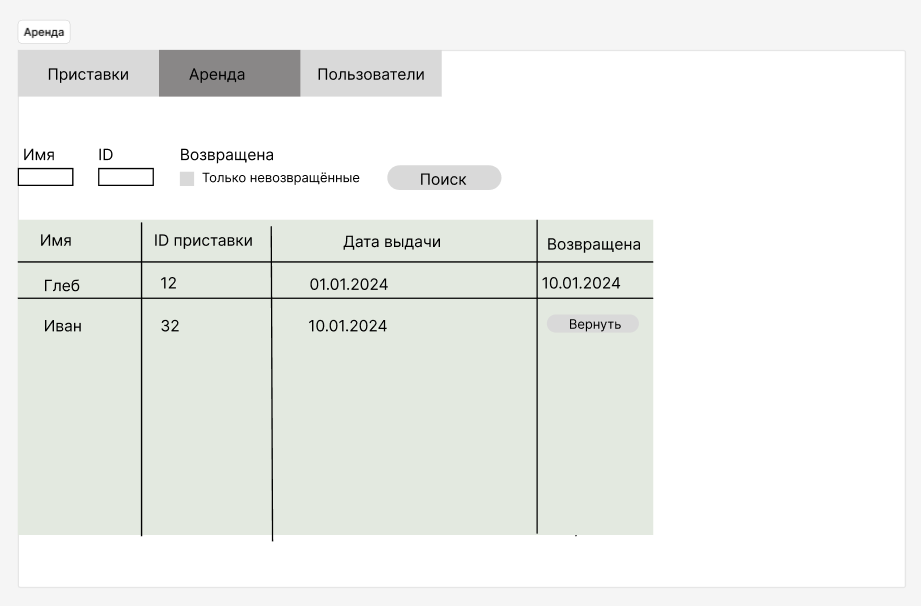
Рисунок 10 – Прототип окна аренда



Рисунок 11 – Прототип окна списка пользователей

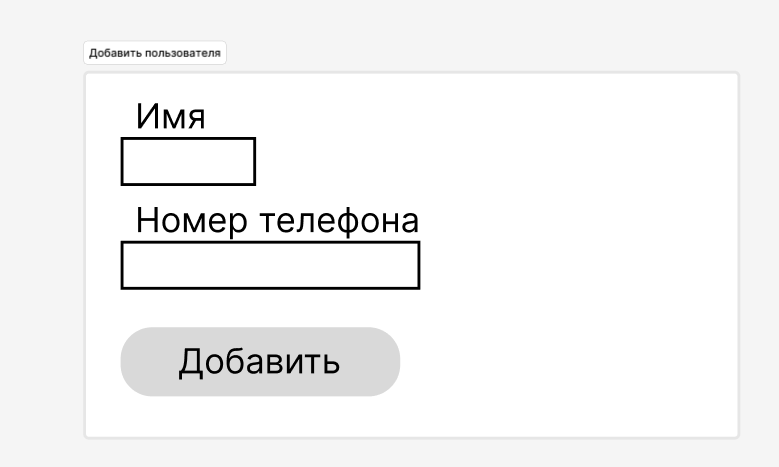


Рисунок 12 – Прототип окна добавления пользователя

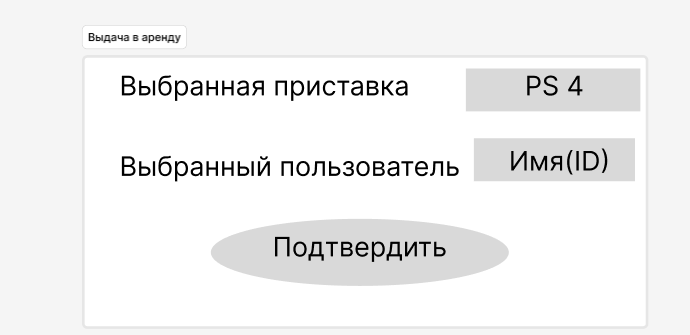


Рисунок 13 – Прототип окна выдача приставки

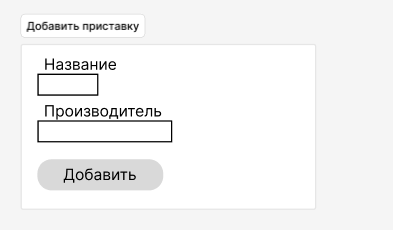


Рисунок 14 – Прототип окна добавления приставки

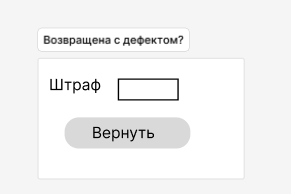


Рисунок 15 – Прототип окна подтверждение возврата

* 1. **Проектирование базы данных**
     1. **Концептуальная модель данных**

На основе анализа предметной области «Аренда игровых приставок», были выделены следующие информационные объекты, которые необходимо хранить в базе данных:

ПРИСТАВКА, ПОЛЬЗОВАТЕЛЬ, АРЕНДА, ПРОИЗВОДИТЕЛЬ.

Каждый из выделенных информационных объектов имеет следующие атрибуты:

1. ПРИСТАВКА – Название, производитель;
2. ПОЛЬЗОВАТЕЛЬ - Имя, номер телефона, штраф;
3. АРЕНДА – Пользователь, Приставка, Дата выдачи, Дата возврата;
4. ПРОИЗВОДИТЕЛЬ – Название;

На рисунке 14 приведена концептуальная модель предметной области «Аренда игровых приставок» в виде ER-диаграммы.

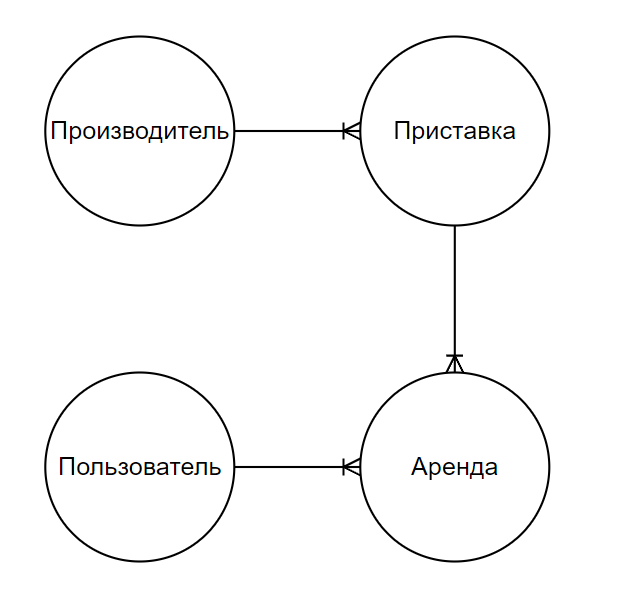


Рисунок 16 – Концептуальная модель ПрО «Аренда игровых приставок»

* + 1. **Логическая модель данных**

На рисунке 17 приведена логическая модель базы данных «Аренда игровых приставок».

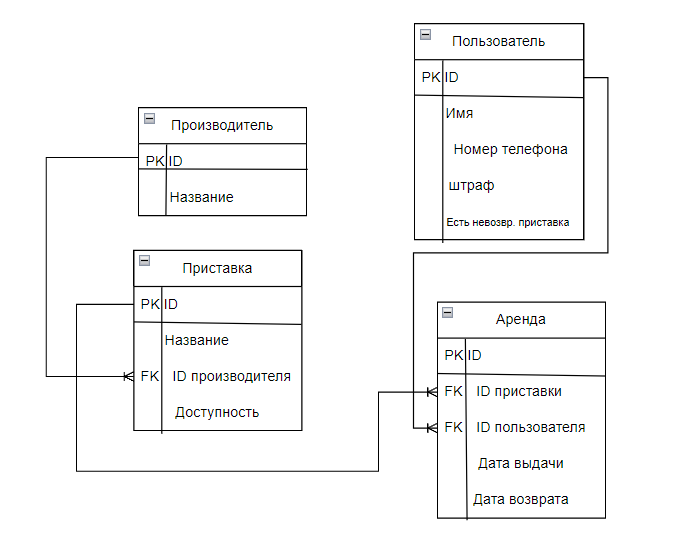


Рисунок 17 – Логическая модель ПрО «Аренда игровых приставок»

1. **Реализация online-сервиса**

Online-cервис будет реализован с использованием схемы разделения данных приложения и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо (Рис. 18).

Листинги модели, шаблона и страницы «Список игроков» приведён в Приложении А.

* Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.
* Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.
* Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

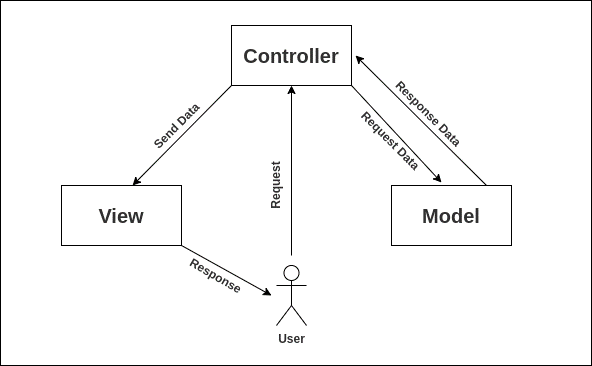


Рисунок 18 – Модель-Вид-Контроллер

* 1. **Функциональные возможности**

1. Осуществлять поиск пользователей по заданным характеристикам.
2. Осуществлять поиск приставок по заданным характеристикам.
3. Позволять администратору добавлять пользователя.
4. Позволять администратору выдавать в аренду приставки.
5. Позволять администратору принимать возврат приставки.
   1. **Выбор средств реализации**

В качестве средств реализации были выбраны:

1. Flask – фреймворк для создания веб-приложений на языке программирования Python.
2. Jinja2 – шаблонизатор для языка программирования Python.
   1. **Структура модулей приложения**

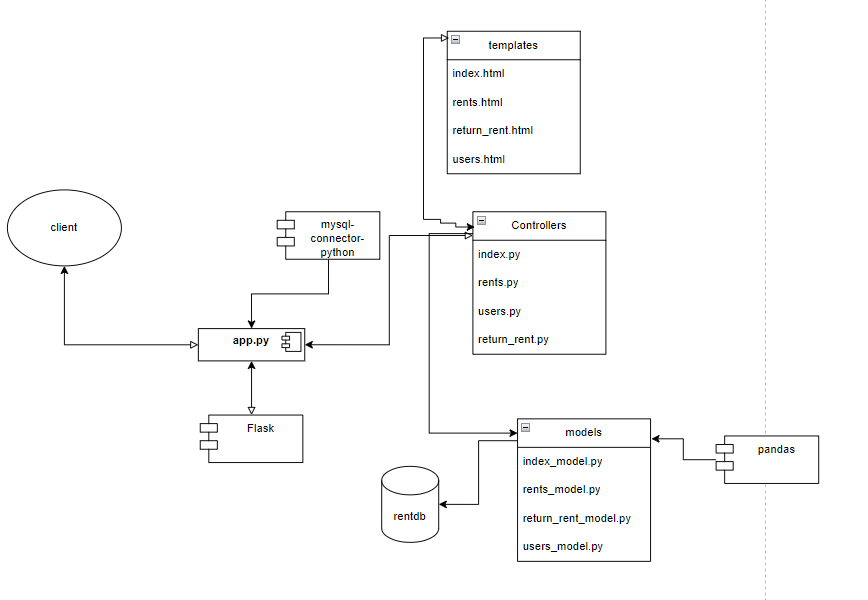


Рисунок 19– Структура модулей приложения

**Заключение**

В рамках проекта был разработан online-сервис для организации аренды игровых приставок, в результате чего были решены все поставленные задачи:

* описана предметная область «Аренда игровых приставок»;
* проведено исследование методом персонажа предметной области «Аренда игровых приставок»;
* описаны сценарии использования разрабатываемого online-cервиса;
* создан проект пользовательского интерфейса;
* создан проект базы данных;
* реализован online-сервис для автоматизации предметной области «Аренда игровых приставок».

**ПРИЛОЖЕНИЕ А**

*Листинг rents.py*

from app import app

from flask import request, session, redirect, url\_for, render\_template

from utils import get\_db\_connection

from models.rents\_model import  get\_rent, get\_rent\_search, get\_user, get\_console, approve

from jinja2 import Environment, FileSystemLoader

@app.route('/rents',methods=['get'])

def rents():

    conn=get\_db\_connection()

    if request.values.get("console\_id")!=None:

        session["console\_id"] = request.values.get("console\_id")

    if request.values.get("user\_id")!=None:

        session["user\_id"] = request.values.get("user\_id")

    user\_name = request.values.get('rents\_user\_name')

    console\_id = request.values.get('rents\_console\_id')

    only\_returned = request.values.get('rents\_returned')

    user = get\_user(conn,session['user\_id'])

    console = get\_console(conn,session['console\_id'])

    df\_rents = get\_rent\_search(conn,user\_name,console\_id,only\_returned)

    html = render\_template(

        'rents.html',

        selected\_user=user,

        selected\_console=console,

        userName = user\_name,

        consoleId=console\_id,

        onlyReturned=only\_returned,

        rents=df\_rents,

        len=len

    )

    return html

@app.route('/rents\_post', methods=["post"])

def rents\_post():

    conn = get\_db\_connection()

    rent\_user\_id = request.values.get('rent\_user\_id')

    rent\_console\_id = request.values.get('rent\_console\_id')

    if rent\_user\_id==None or rent\_console\_id==None:

        return redirect(url\_for('rents'))

    approve(conn, rent\_user\_id,rent\_console\_id)

    return redirect(url\_for("rents"))

*Листинг rents\_model.py*

import sqlite3

import pandas

def get\_rent(conn):

        return pandas.read\_sql(

    '''

    SELECT \* FROM RENT

    ''', conn)

def get\_rent\_search(conn, name = '', id = None, only\_returned = False):

    if name==None:

        name=''

    if id=='':

        id=None

    command = '''

    SELECT RentID as RentID,

    USERS.Name as Имя, ConsoleID as ConsoleID,

    RentDate as "Начало аренды",

    ReturnDate as 'Конец аренды'

    from RENT

    join USERS USING(UserID)

    '''

    # WHERE Name LIKE  + name + "%"

    if name!='' or id!=None or only\_returned:

        command+=" WHERE "

        commands = []

        if name!='':

            commands.append('USERS.Name like "' + name +'%"')

        if only\_returned:

            commands.append("ReturnDate!=NULL")

        if id!=None:

            commands.append("ConsoleID==" + str(id))

        command+=' and '.join(commands)

    return pandas.read\_sql(command, conn)

def get\_user(conn,user\_id):

    uid\_limit = (pandas.read\_sql(

    '''

    select count(\*) as Количество from USERS

    ''',conn

    ))['Количество'][0]

    if user\_id!='' and user\_id!=None:

        if int(user\_id)<0 or int(user\_id)>=uid\_limit:

            return None

    return pandas.read\_sql('''

    select name, PhoneNumber

    from USERS

    where UserID=='''+str(user\_id),conn)

def get\_console(conn,console\_id):

    cid\_limit = (pandas.read\_sql(

    '''

    select count(\*) as Количество from CONSOLE

    ''',conn

    ))['Количество'][0]

    if console\_id!='' and console\_id!=None:

        if int(console\_id)<0 or int(console\_id)>=cid\_limit:

            return None

    return pandas.read\_sql('''

    select Console.Name as CN, Company.name as CMN

    from Console

    join Company using (CompanyID)

    where ConsoleID=='''+str(console\_id),conn)

def approve(conn : sqlite3.Connection, user\_id, console\_id):

    cur = conn.cursor()

    uid\_limit = (pandas.read\_sql(

    '''

    select count(\*) as Количество from RENT

    ''',conn

    ))['Количество'][0]

    if int(uid\_limit)==0:

        cur.execute("""

            INSERT INTO RENT

            VALUES (0,:ci, :ui,DATE("now"),NULL)

        """, {"ui": user\_id, "ci" : console\_id})

    else:

        cur.execute("""

            INSERT INTO RENT

            VALUES (:i,:ci, :ui,DATE("now"),NULL)

        """, {"ui": user\_id, "ci" : console\_id, 'i':int(uid\_limit)})

    cur.execute("""

        UPDATE USERS

        SET HasRented = true

        WHERE UserID = :ui

    """, {"ui": user\_id})

    cur.execute("""

        UPDATE CONSOLE

        SET IsRented = true

        WHERE ConsoleID = :ci

    """, {"ci": console\_id})

    conn.commit()

    return cur.lastrowid

*Листинг rents.html*

<!DOCTYPE HTML>

<html>

<head>

    <link rel="stylesheet" type="text/css" href="/static/css/main.css" />

    <title> Аренда  </title>

    <!-- Создать макрос для вывода таблицы без ключевого столбца, Параметры: имя таблицы, dataFrame с таблицей, имя

        ключевого столбца -->

        {% macro render\_table(table\_name, relation, id) %}

        <table>

            <tr>

                {% for name in relation.dtypes.index %}

                {% if name != id%}

                <th>{{name}}</th>

                {% endif %}

                {% endfor %}

            </tr>

            <tbody>

                <!-- для всех строк таблицы -->

                {% for i in range(relation.\_\_len\_\_()) %}

                <tr>

                    <!-- выводим значение каждого столбца -->

                    {% for name in relation.dtypes.index %}

                    {% if relation.loc[i,name]==None %}

                    <td>

                        <form action="{{url\_for('return\_rent')}}">

                        <input type="hidden" name="rent\_return\_id" value={{relation.loc[i,'RentID']}}>

                        <input type="submit" value="Вернуть аренду">

                        </form>

                    </td>

                    {% else %}

                    <td> {{ relation.loc[i, name] }} </td>

                    {% endif %}

                    {% endfor %}

                </tr>

                {% endfor %}

            </tbody>

        </table>

        {% endmacro %}

</head>

<body>

    <div class="menu">

        <ul>

            <li ><a href="{{url\_for('index')}}">Консоли</a></li>

           <li class="active"><a href="{{url\_for('rents')}}">Аренда</a></li>

            <li ><a href="{{url\_for('users')}}">Пользователи</a></li>

        </ul>

    </div>

    <div class="block">

        <p>

            <div id="zatemnenie">

                <div id="okno">

                    <form action="{{url\_for('rents\_post')}}" method="post">

                        <p>

                            Выбранный пользователь:<br>

                            {% if not selected\_user.empty %}

                            <label>Имя: {{selected\_user['Name'][0]}}</label>

                            <label>Номер телефона: {{selected\_user['PhoneNumber'][0]}}</label>

                            {% else %}

                            <p>Пользователь не выбран!</p>

                            {% endif %}

                        </p>

                        <p>

                        Выбранная консоль:

                        {% if not selected\_console.empty  %}

                        <p>Название: {{selected\_console['CN'][0]}}</p>

                        <p>Компания: {{selected\_console['CMN'][0]}}</p>

                        {% else %}

                        <p>Консоль не выбран!</p>

                        {% endif %}

                        </p>

                        <a href="#" class="close">Отменить</a>

                        <input type="hidden" name="rent\_user\_id" value={{session['user\_id']}}>

                        <input type="hidden" name="rent\_console\_id" value={{session['console\_id']}}>

                        <input type="submit" value="Новая аренда">

                    </form>

                </div>

              </div>

              <form method="get" id="search">

                <div>

                    Имя:

                    <input type="text" name="rents\_user\_name" value={{userName if userName!=None else ""}}>

                </div>

                <div>

                    ID:

                    <input type="text" name="rents\_console\_id" value={{consoleId if consoleId!=None else ""}}>

                </div>

                <div>

                    Возвращена:

                    <input type="checkbox" name="rents\_returned" {{'checked="checked"' if onlyReturned else ""}}>

                    Только возвращённые

                </div>

                <input type="submit" value="Найти">

               </form>

            <a href="#zatemnenie">Добавить аренду</a>

        </p>

        {{ render\_table("Аренда",rents, "") }}

    </div>

</body>

</html>