# Тема

Веб-приложение «ДНД общежитие №1».

# Словесное описание предметной области и актуальность

В настоящих условиях, регистрация на игру и получение информации об предстоящей игре проходит через группу в вк. Т.к. это очень неудобно, то я хотел бы это облегчить, чтобы пользователи могли регистрироваться на данном ресурсе, а администратор мог их распределять по мастерам и играм.

# Описание данных, хранящихся в БД

БД должна хранить следующие данные:

* + Личную информацию об игроках;
  + Информацию о персонажах игрока;
  + Информацию об артефактах на конкретном персонаже;
  + Информацию о ролях пользователя;
  + Список Книг для игры в днд;
  + Список скиллов;
  + Информацию о скиллах, которые есть у конкретного персонажа
  + Новости о днд в общаге
  + Информацию о проведенных играх и будущих игр
  + Информацию о компаниях
  + Список мест проведения

# Роли пользователей приложения

Администратор;

Игрок, зарегистрированный в системе;

Мастер, зарегистрированный в системе;

# Развернутое описание функционала приложения для каждой из ролей

1. Посетителю доступно:

Только регистрация

1. Зарегистрированному пользователю доступно:

Информация о своем персонаже и его артефактах и скиллах, информация о компаниях, в которых он состоит, информация об его истории игр, список книг, просматривать новости. Зарегистрированный пользователь может добавлять персонажа, добавлять скиллы на персонажа. Изменять информацию о персонаже.

1. Мастеру доступно:

Все тоже, что и персонажу. А также мастер может создавать компании, добавлять в компании информацию о доступных книгах игрокам в данной компании, добавлять в компанию игрока. Планировать время игры, и также добавлять игроков в игры. Выдавать артефакты персонажам игроков.

1. Админу доступно все тоже, что и мастеру. А также админ может сам определять мастера в компанию, добавлять в список книг новые книги, в список мест проведения добавлять новые места. Добавлять в список скиллов новые скиллы. Выдавать роль мастера игрокам

# Срок предоставления курсовой работы

25.12.2020

# ER-диаграмма разработанной БД

Все данные сайта будут храниться в базе данных. Воспользуемся ER диаграммой для ее проектирования. ER-диаграмма изображена на рисунке 1.

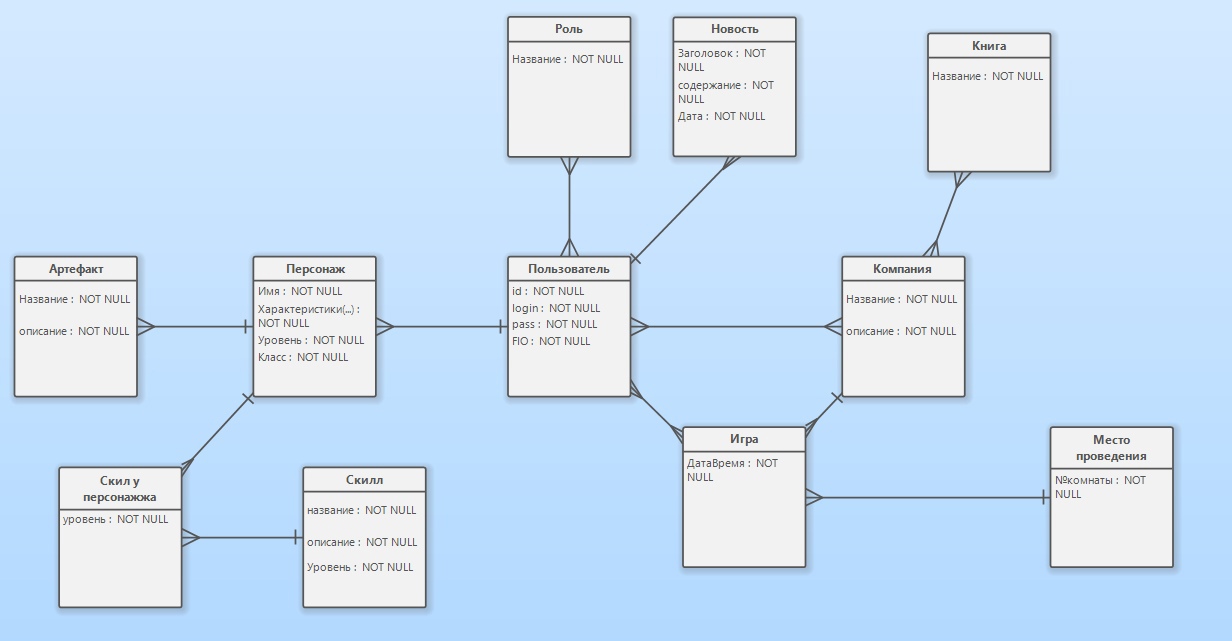


Рисунок 1 – Реляционная схема

# Описание выбранных технологий реализации.

Для разработки веб-сайта «Днд Общежите №1» необходимы следующие дополнительные инструментальные программные средства:

* язык программирования Python и его веб-фреймворк Flask,
* система управления базами данных PostgreSQL
* система объектно-реляционного связывания между Python и PostgreSQL SQLAlchemy

Flask — [фреймворк](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) для создания [веб-приложений](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) на языке программирования [Python](https://ru.wikipedia.org/wiki/Python), использующий набор инструментов Werkzeug, а также шаблонизатор [Jinja2](https://ru.wikipedia.org/wiki/Jinja). Относится к категории так называемых [микрофреймворков](https://ru.wikipedia.org/w/index.php?title=%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA&action=edit&redlink=1) — минималистичных каркасов веб- приложений, сознательно предоставляющих лишь самые базовые возможности.

Описание возможностей Flask:

1. Очень модульный Фреймворк с гибкой системой настроек.
2. Можно легко написать RBAC или использовать обычную [ACL](http://docs.pylonsproject.org/projects/pyramid/en/latest/narr/security.html?highlight=security).
3. Легко подключаются MongoDB и CauchDB.
4. Для шаблонизации можно подключать [Chameleon, Jinja2, Mako](http://docs.pylonsproject.org/projects/pyramid/en/latest/narr/templates.html).
5. Фреймворк позволяет делать всё что угодно и предоставляет простую систему работы с запросами и обслуживания безопасности.
6. Поддерживается Python 3.

PostgreSQL – это свободно распространяемая объектно-реляционная система управления базами данных (ORDBMS), наиболее развитая из открытых СУБД в мире. PostgreSQL базируется на языке SQL и поддерживает многие из возможностей стандарта SQL:2011.

Сильными сторонами PostgreSQL считаются:

1. Высокопроизводительные и надёжные механизмы [транзакций](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B0%D0%BA%D1%86%D0%B8%D1%8F_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) и [репликации](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BF%D0%BB%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_(%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0));
2. Расширяемая система встроенных языков программирования: в стандартной поставке поддерживаются [PL/pgSQL](https://ru.wikipedia.org/wiki/PL/pgSQL), [PL/Perl](https://ru.wikipedia.org/wiki/PL/Perl), [PL/Python](https://ru.wikipedia.org/wiki/PL/Python) и [PL/Tcl](https://ru.wikipedia.org/wiki/PL/Tcl); дополнительно можно использовать PL/Java, PL/PHP, PL/Py, PL/R, PL/Ruby, [PL/Scheme](https://ru.wikipedia.org/wiki/PL/Scheme), PL/sh и PL/V8, а также имеется поддержка загрузки модулей расширения на языке [C](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F));
3. [Наследование](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5));
4. Возможность индексирования геометрических объектов и наличие базирующегося на ней расширения [PostGIS](https://ru.wikipedia.org/wiki/PostGIS);
5. Встроенная поддержка [слабоструктурированных данных](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%B0%D0%B1%D0%BE%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5) в формате [JSON](https://ru.wikipedia.org/wiki/JSON) с возможностью их индексации;
6. Расширяемость (возможность создавать новые типы данных, типы индексов, языки программирования, модули расширения, подключать любые внешние источники данных).

SQLAlchemy — это программная библиотека на языке Python для работы с реляционными СУБД с применением технологии ORM. Служит для синхронизации объектов Python и записей реляционной базы данных. SQLAlchemy позволяет описывать структуры баз данных и способы взаимодействия с ними на языке Python без использования SQL.

1. Основные возможности SQLAlchemy:
2. Использование ORM не является обязательным
3. Устоявшаяся архитектура
4. Возможность использовать SQL, написанный вручную
5. Поддержка транзакций
6. Создание запросов с использованием функций и выражений Python
7. Модульность и расширяемость
8. Дополнительная возможность раздельного определения объектного отображения и классов
9. Поддержка составных индексов
10. Поддержка отношений между классами, в том числе «один-ко-многим» и «многие-ко-многим»
11. Поддержка ссылающихся на себя объектов
12. Предварительная и последующая обработка данных (параметров запроса, результата)

# Описание используемых методов взаимодействия с БД.

База данных подключается в файле \_\_init\_\_.py проекта. Для подключения необходимо импортировать psycopg2 в проект и создать переменную с помощью которой мы будем обращается к СУБД. Используя SQLAlchemy подключаемся к БД postrges, указывая имя пользователя, пароль, ip-адрес с портом, а также название подключаемой БД. Благодаря SQLAlchemy мы можем делать запросы, изменять БД, а так же сохранять в ней результаты запроса.

# Описание архитектуры приложения

Проект состоит из большого числа файлов. Рассмотрим основные из них.

* routes.py – основная логика работы веб-приложения – ответ на запросы пользователя по конкретным веб-адресам в виде html-страницы.
* models.py – представление сущностей БД в виде классов;
* init .py – инициализация приложения;
* forms.py – содержит, формы для упрощения работы c html-страницами.
* config.py – содержит значение секретного ключа. Flask и некоторые его расширения используют значение секретного ключа в качестве криптографического ключа, полезного для генерации подписей или токенов;
* test.py – запуск приложения.

Все html-шаблоны храниться в папке templates.

Html-шаблоны:

* addbook.html – шаблон добавления книги;
* addbookcompany.html – шаблон добавления книги в компанию
* addcompany.html – шаблон создания компании
* addhero.html – шаблон создания героя(тоже что и персонаж)
* addnews.html – шаблон создания новости/объявления
* addplace.html – шаблон добавления нового места проведения
* addplay.html – шаблон создания прошедшей/предстоящей игры
* addplayercompany.html – шаблон добавления игрока в компанию
* addplayerplay.html – шаблон отметки игрока о присутствии на игре
* addskill.html – шаблон добавления нового скилла на сайт
* addskillhero.html – шаблон добавления нового скилла персонажу
* admin.html – шаблон панели администратора
* base.html – базовый шаблон
* editor.html – шаблон панели Мастера
* giveartefact.html – шаблон выдачи артефакта герою
* giverole.html – шаблон выдачи роли мастера игроку
* hero.html – шаблон информации о герое
* index.html – шаблон главная страница сайта
* infocompany.html – шаблон информации о компании
* login.html – шаблон авторизации
* mycompany.html – шаблон просмотра компаний игрока
* register.html – шаблон регистрации
* storygame.html – шаблон истории игр
* user.html – шаблон профиля пользователя
* viewbook.html – шаблон отображения списка книг

# Листинг приложения

## Файл routes.py:

from datetime import date

from html.entities import name2codepoint

from logging import log

from os import name

import re

from flask import render\_template, flash, redirect, url\_for

from sqlalchemy.orm import query

from sqlalchemy.sql.functions import current\_time, current\_timestamp

from werkzeug.utils import validate\_arguments

from wtforms.fields import choices

from app import app

from app.forms import \*

from .models import Book, Customers, Hero, RoleUser

from app import Dnd

from flask import (

    Blueprint,

    render\_template,

    request,

    flash,

    abort,

    redirect,

    url\_for,

    current\_app,

)

from flask\_login import current\_user, login\_user,logout\_user, login\_required

from werkzeug.urls import url\_parse

@app.route('/')

@app.route('/index')

@login\_required

def index():

    news=News.query.all()

    return render\_template('index.html', title='Home', news=news)

@app.route('/login', methods=['GET', 'POST'])

def login():

    if current\_user.is\_authenticated:

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

    form = LoginForm()

    if form.validate\_on\_submit():

        user = Customers.query.filter\_by(login=form.username.data).first()

        if user is None or not user.check\_password(form.password.data):

            flash('Invalid username or password')

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

        login\_user(user, remember=form.remember\_me.data)

        next\_page = request.args.get('next')

        if not next\_page or url\_parse(next\_page).netloc != '':

            next\_page = url\_for('index')

        return redirect(next\_page)

    return render\_template('login.html', title='Sign In', form=form)

@app.route('/register', methods=['GET', 'POST'])

def register():

    if current\_user.is\_authenticated:

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

    form = RegistrationForm()

    if form.validate\_on\_submit():

        user = Customers(login=form.username.data,forename=form.forename.data,surname=form.surname.data)

        user.set\_password(form.password.data)

        Dnd.session.add(user)

        role=RoleUser(login=form.username.data,namerole='Player')

        Dnd.session.add(role)

        Dnd.session.commit()

        flash('Congratulations, you are now a registered user!')

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

    return render\_template('register.html', title='Register', form=form)

@app.route('/user/<login>/addhero', methods=['GET', 'POST'])

def addhero(login):

    form = AddHeroForm()

    if form.validate\_on\_submit():

        hero = Hero(namehero=form.namehero.data,

        strength=form.strength.data,

        dexterity=form.dexterity.data,

        wisdom=form.wisdom.data,

        physique=form.physique.data,

        intelligence=form.intelligence.data,

        classhero=form.classhero.data,

        charisma=form.charisma.data,

        herolevel=form.herolevel.data,

        login=login)

        print(hero)

        Dnd.session.add(hero)

        Dnd.session.commit()

        flash('Congratulations, you are add new hero!')

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

    return render\_template('addHero.html', title='Addhero', form=form)

@app.route('/user/<login>', methods=['GET','POST'])

@login\_required

def user(login):

    form = ViewMoreInfoHeroForm()

    user = Customers.query.filter\_by(login=login).first\_or\_404()

    heroes = Dnd.session.query(Hero.namehero).filter\_by(login=login)

    all\_heroes=heroes.all()

    heroes\_user=[i[0] for i in all\_heroes]

    form.namehero.choices=heroes\_user

    namehero=form.namehero.data

    print(namehero)

    if form.validate\_on\_submit():

        return redirect(url\_for('infohero',login=login,namehero=namehero))

    return render\_template('user.html',title='Profile', user=user, form=form )

@app.route('/user/<login>/<namehero>', methods=['GET','POST'])

@login\_required

def infohero(namehero,login):

    print('///////////////')

    print(namehero)

    #hero\_user=Hero.query.filter\_by(namehero=namehero).all()

    hero\_user=Hero.query.get\_or\_404(namehero)

    skills=Dnd.session.query(SkillHero.skill).filter\_by(namehero=namehero).all()

    artefacts=Dnd.session.query(Artefact.artefactname).filter\_by(namehero=namehero).all()

    print(skills)

    form=InfoHero()

    if form.validate\_on\_submit():

        hero\_user.strength= form.strength.data

        hero\_user.dexterity= form.dexterity.data

        hero\_user.physique=form.physique.data

        hero\_user.intelligence=form.intelligence.data

        hero\_user.wisdom=form.wisdom.data

        hero\_user.charisma=form.charisma.data

        hero\_user.herolevel=form.herolevel.data

        Dnd.session.commit()

        return redirect(url\_for('user',login=login))

    form.strength.data=hero\_user.strength

    form.dexterity.data=hero\_user.dexterity

    form.physique.data=hero\_user.physique

    form.intelligence.data=hero\_user.intelligence

    form.wisdom.data=hero\_user.wisdom

    form.charisma.data=hero\_user.charisma

    form.herolevel.data=hero\_user.herolevel

    return render\_template('hero.html',title='Hero',form=form,hero=hero\_user,skills=skills,artefacts=artefacts)

@app.route('/user/<login>/<namehero>/addskillhero', methods=['GET','POST'])

def addskillhero(namehero,login):

    form=AddSkillHeroForm()

    skills = Dnd.session.query(Skill.skill)

    all\_skills=skills.all()

    skill\_user=[i[0] for i in all\_skills]

    form.skill.choices=skill\_user

    skillhero= SkillHero.query.filter\_by(namehero=namehero,skill=form.skill.data).first()

    if skillhero is not None:

        flash('Данный пользователь уже добавлен в данную игру')

        return redirect(url\_for('addskillhero',login=login,namehero=namehero))

    if form.validate\_on\_submit():

        skill = SkillHero(namehero=namehero,skill=form.skill.data)

        Dnd.session.add(skill)

        Dnd.session.commit()

        flash('Congratulations, you are add new Skill!')

        return redirect(url\_for('user',login=login))

    return render\_template('addskillhero.html', title='AddSkillHero',form=form,login=login,namehero=namehero)

@app.route('/admin',methods=['GET','POST'])

@login\_required

def admin():

    admin=RoleUser.query.filter\_by(login=current\_user.login).first\_or\_404()

    if admin.namerole =='Admin':

        form=EditorForm()

        company = Dnd.session.query(Company.namecompany)

        all\_company=company.all()

        companys\_user=[i[0] for i in all\_company]

        form.namecompany.choices=companys\_user

        if form.validate\_on\_submit():

            print(form.namecompany.data)

            return redirect(url\_for('infocompany',namecompany=form.namecompany.data))

        return render\_template('admin.html',form=form)

    else:

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

@app.route('/logout')

def logout():

    logout\_user()

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

@app.route('/admin/addplace', methods=['GET', 'POST'])

def addplace():

    form = AddPlaceForm()

    if form.validate\_on\_submit():

        place = Place(numberroom=form.place.data)

        Dnd.session.add(place)

        Dnd.session.commit()

        flash('Congratulations, you are add new Place!')

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

    return render\_template('addplace.html', title='Addplace',form=form)

@app.route('/admin/addskill', methods=['GET', 'POST'])

def addskill():

    form = AddSkillForm()

    if form.validate\_on\_submit():

        skill = Skill(skill=form.skill.data,discription=form.discription.data,skilllevel=form.skilllevel.data)

        Dnd.session.add(skill)

        Dnd.session.commit()

        flash('Congratulations, you are add new Skill!')

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

    return render\_template('addskill.html', title='Addskill', form=form)

@app.route('/admin/addbook', methods=['GET', 'POST'])

def addbook():

    form = AddBookForm()

    if form.validate\_on\_submit():

        book = Book(namebook=form.namebook.data)

        Dnd.session.add(book)

        Dnd.session.commit()

        flash('Congratulations, you are add new book!')

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

    return render\_template('addbook.html', title='Addbook', form=form)

@app.route('/admin/addnews', methods=['GET', 'POST'])

def addnews():

    form = AddNewsForm()

    date=datetime.utcnow()

    if form.validate\_on\_submit():

        news = News(title=form.title.data,discription=form.discription.data,login=current\_user.login,datet=date)

        Dnd.session.add(news)

        Dnd.session.commit()

        flash('Congratulations, you are add new News!')

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

    return render\_template('addnews.html', title='Addnews', form=form)

@app.route('/admin/giverole', methods=['GET', 'POST'])

def giverole():

    form = GiveRoleForm()

    user\_ids = Dnd.session.query(Customers.login)

    all\_ids = user\_ids.all()

    users=[i[0] for i in all\_ids]

    form.user.choices=users

    if form.validate\_on\_submit():

        book = RoleUser(login=form.user.data,namerole=form.namerole.data)

        Dnd.session.add(book)

        Dnd.session.commit()

        flash('Congratulations, you are giverole!', book.login)

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

    return render\_template('giverole.html', title='Giverole', form=form)

@app.route('/admin/addcompanyadmin',methods=['GET','POST'])

def addcompanyadmin():

    form=AddCompanyForm()

    user\_ids = Dnd.session.query(RoleUser.login).filter\_by(namerole='Dangeon\_master')

    #user\_ids = Dnd.session.query(Customers.login)

    all\_ids = user\_ids.all()

    users=[i[0] for i in all\_ids]

    form.dangeonmaster.choices=users

    if form.validate\_on\_submit():

        s=form.namecompany.data

        print(s)

        print('/////////////////')

        book=Company(namecompany=s,discription=form.discription.data,dungeonmaster=form.dangeonmaster.data)

        Dnd.session.add(book)

        Dnd.session.commit()

        flash('Congratulations, you are create company!')

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

    return render\_template('addcompany.html', title='AddCompany', form=form)

@app.route('/admin/addplayeradmin',methods=['GET','POST'])

def addplayeradmin():

    form = AddPlayerInCompanyForm()

    company = Dnd.session.query(Company.namecompany)

    all\_company=company.all()

    company\_user=[i[0] for i in all\_company]

    form.namecompany.choices=company\_user

    user\_ids = Dnd.session.query(Customers.login)

    all\_ids = user\_ids.all()

    users=[i[0] for i in all\_ids]

    form.player.choices=users

    playercompany=CustomersCompany.query.filter\_by(namecompany=form.namecompany.data, login=form.player.data).first()

    if playercompany is not None:

        flash('Данный игрок уже добавлена в компанию')

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

    if form.validate\_on\_submit():

        player = CustomersCompany(login=form.player.data,namecompany=form.namecompany.data)

        Dnd.session.add(player)

        Dnd.session.commit()

        flash('Congratulations, you are add new player in company!')

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

    return render\_template('addplayercompany.html', title='AddPlayer', form=form)

@app.route('/admin/infocompany/<namecompany>',methods=['GET','POST'])

def infocompanyadmin(namecompany):

    form=EditCompanyForm()

    discription\_company=Company.query.get\_or\_404(namecompany)

    players=Dnd.session.query(CustomersCompany.login).filter\_by(namecompany=namecompany).all()

    if form.validate\_on\_submit():

        discription\_company.discription=form.discription.data

        Dnd.session.commit()

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

    form.discription.data=discription\_company.discription

    return render\_template('infocompany.html', title='EditCompany',players=players, form=form,namecompany=namecompany,dangeonmaster=discription\_company.dungeonmaster)

@app.route('/editor',methods=['GET','POST'])

@login\_required

def editor():

    admin=RoleUser.query.filter\_by(login=current\_user.login,namerole='Dangeon\_master').first()

    if admin==None:

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

    else:

        form=EditorForm()

        company = Dnd.session.query(Company.namecompany).filter\_by(dungeonmaster=current\_user.login)

        all\_company=company.all()

        companys\_user=[i[0] for i in all\_company]

        form.namecompany.choices=companys\_user

        if form.validate\_on\_submit():

            print(form.namecompany.data)

            return redirect(url\_for('infocompany',namecompany=form.namecompany.data))

        return render\_template('editor.html',title='Editor',form=form)

@app.route('/editor/addcompanyeditor',methods=['GET','POST'])

def addcompanyeditor():

    form=AddCompanyForm()

    user\_ids = Dnd.session.query(RoleUser.login).filter\_by(login=current\_user.login,namerole='Dangeon\_master')

    #user\_ids = Dnd.session.query(Customers.login)

    all\_ids = user\_ids.all()

    users=[i[0] for i in all\_ids]

    form.dangeonmaster.choices=users

    if form.validate\_on\_submit():

        s=form.namecompany.data

        print(s)

        print('/////////////////')

        book=Company(namecompany=s,discription=form.discription.data,dungeonmaster=form.dangeonmaster.data)

        Dnd.session.add(book)

        Dnd.session.commit()

        flash('Congratulations, you are create company!')

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

    return render\_template('addcompany.html', title='AddCompany', form=form)

@app.route('/editor/addbook', methods=['GET', 'POST'])

def addbookcompany():

    form = AddBookCompanyForm()

    login=current\_user.login

    company = Dnd.session.query(Company.namecompany).filter\_by(dungeonmaster=login)

    all\_company=company.all()

    company\_user=[i[0] for i in all\_company]

    form.namecompany.choices=company\_user

    book = Dnd.session.query(Book.namebook)

    all\_books=book.all()

    book\_user=[i[0] for i in all\_books]

    form.namebook.choices=book\_user

    bookcompany=BookCompany.query.filter\_by(namecompany=form.namecompany.data, namebook=form.namebook.data).first()

    if bookcompany is not None:

        flash('Данная книга уже добавлена в компанию')

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

    if form.validate\_on\_submit():

        book = BookCompany(namebook=form.namebook.data,namecompany=form.namecompany.data)

        Dnd.session.add(book)

        Dnd.session.commit()

        flash('Congratulations, you are add new book in company!')

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

    return render\_template('addbookcompany.html', title='Addbook', form=form)

@app.route('/editor/addplayer', methods=['GET', 'POST'])

def addplayercompany():

    form = AddPlayerInCompanyForm()

    login=current\_user.login

    company = Dnd.session.query(Company.namecompany).filter\_by(dungeonmaster=login)

    all\_company=company.all()

    company\_user=[i[0] for i in all\_company]

    form.namecompany.choices=company\_user

    user\_ids = Dnd.session.query(Customers.login)

    all\_ids = user\_ids.all()

    users=[i[0] for i in all\_ids]

    form.player.choices=users

    playercompany=CustomersCompany.query.filter\_by(namecompany=form.namecompany.data, login=form.player.data).first()

    if playercompany is not None:

        flash('Данный игрок уже добавлена в компанию')

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

    if form.validate\_on\_submit():

        player = CustomersCompany(login=form.player.data,namecompany=form.namecompany.data)

        Dnd.session.add(player)

        Dnd.session.commit()

        flash('Congratulations, you are add new player in company!')

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

    return render\_template('addplayercompany.html', title='AddPlayer', form=form)

@app.route('/editor/addplay',methods=['GET','POST'])

def addplay():

    form = AddPlayForm()

    login=current\_user.login

    room = Dnd.session.query(Place.numberroom)

    all\_room=room.all()

    rooms=[i[0] for i in all\_room]

    form.numberroom.choices=rooms

    company = Dnd.session.query(Company.namecompany).filter\_by(dungeonmaster=login)

    all\_company=company.all()

    company\_user=[i[0] for i in all\_company]

    form.namecompany.choices=company\_user

    playcompany=Play.query.filter\_by(namecompany=form.namecompany.data, numberroom=form.numberroom.data,datet=form.datet.data).first()

    if playcompany is not None:

        flash('Данная игра с этой комнатой в данное время уже добавлена в таблицу ')

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

    if form.validate\_on\_submit():

        play = Play(namecompany=form.namecompany.data,numberroom=form.numberroom.data,datet=form.datet.data)

        Dnd.session.add(play)

        Dnd.session.commit()

        flash('Congratulations, you are add play!')

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

    return render\_template('addplay.html', title='AddPlay', form=form)

@app.route('/editor/addplayerplay',methods=['GET','POST'])

def addplayerplay():

    form = AddPlayerPlayForm()

    user\_ids = Dnd.session.query(Customers.login)

    all\_ids = user\_ids.all()

    users=[i[0] for i in all\_ids]

    form.login.choices=users

    idgame\_ids = Dnd.session.query(Play.id)

    all\_idagames = idgame\_ids.all()

    idgames=[i[0] for i in all\_idagames]

    form.idgame.choices=idgames

    playerplay= CustomersPlay.query.filter\_by(login=form.login.data,idgame=form.idgame.data).first()

    if playerplay is not None:

        flash('Данный пользователь уже добавлен в данную игру')

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

    if form.validate\_on\_submit():

        playerplays = CustomersPlay(login=form.login.data,idgame=form.idgame.data)

        Dnd.session.add(playerplays)

        Dnd.session.commit()

        flash('Congratulations, you are add player id play!')

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

    return render\_template('addplayerplay.html', title='AddPlay', form=form)

@app.route('/editor/giveartefact',methods=['GET','POST'])

def giveartefact():

    form=GiveArtefact()

    hero = Dnd.session.query(Hero.namehero)

    all\_hero=hero.all()

    heroes=[i[0] for i in all\_hero]

    form.namehero.choices=heroes

    if form.validate\_on\_submit():

        artefact = Artefact(artefactname=form.artefact.data,namehero=form.namehero.data,discription=form.discription.data)

        Dnd.session.add(artefact)

        Dnd.session.commit()

        flash('Congratulations, you are give arttefact!')

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

    return render\_template('giveartefact.html', title='GiveArtefact',form=form)

@app.route('/editor/infocompany/<namecompany>',methods=['GET','POST'])

def infocompany(namecompany):

    form=EditCompanyForm()

    discription\_company=Company.query.get\_or\_404(namecompany)

    players=Dnd.session.query(CustomersCompany.login).filter\_by(namecompany=namecompany).all()

    if form.validate\_on\_submit():

        discription\_company.discription=form.discription.data

        Dnd.session.commit()

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

    form.discription.data=discription\_company.discription

    return render\_template('infocompany.html', title='EditCompany',players=players, form=form,namecompany=namecompany,dangeonmaster=discription\_company.dungeonmaster)

@app.route('/viewbook',methods=['GET','POST'])

def viewbook():

    books=Dnd.session.query(Book.namebook)

    return render\_template('viewbook.html', title='Viewbook',books=books)

@app.route('/user/<login>/storygame',methods=['GET','POST'])

def storygame(login):

    idgames=Dnd.session.query(CustomersPlay.idgame).filter\_by(login=login).all()

    print(idgames)

    idgame=[i[0] for i in idgames]

    namegame=[]

    for id in idgame:

        print(id)

        game=Dnd.session.query(Play.namecompany,Play.datet,Play.numberroom).filter\_by(id=id).first()

        print(game)

        namegame.append(game)

    print(namegame)

    return render\_template('storygame.html', title='storygame',login=login,namegame=namegame)

@app.route('/user/<login>/company')

def mycompany(login):

    companys=Dnd.session.query(CustomersCompany.namecompany).filter\_by(login=login).all()

    company=[i[0] for i in companys]

    namegame=[]

    for id in company:

        print(id)

        game=Dnd.session.query(Company.namecompany,Company.dungeonmaster, Company.discription).filter\_by(namecompany=id).first()

        print(game)

        namegame.append(game)

    print(namegame)

    return render\_template('mycompany.html', title='Mycompany',namegame=namegame)

## Файл models.py:

import datetime

from logging import log

from flask\_login.utils import login\_user

from sqlalchemy.sql.functions import current\_timestamp

from werkzeug.security import generate\_password\_hash, check\_password\_hash

from app import Dnd

from flask\_login import UserMixin

from sqlalchemy import func

from app import login

from sqlalchemy.orm import relationship

from datetime import datetime

class Customers(UserMixin,Dnd.Model):

    \_\_tablename\_\_ = 'customers'

    id = Dnd.Column(Dnd.Integer, primary\_key=True)

    login = Dnd.Column(Dnd.String(),unique=True)

    password = Dnd.Column(Dnd.String())

    surname = Dnd.Column(Dnd.String())

    forename = Dnd.Column(Dnd.String())

    #hero=relationship("hero", backref="customers")

    def \_\_init\_\_(self, login, surname, forename):

        self.login = login

        self.surname =  surname

        self.forename = forename

    def set\_password(self, password):

        self.password = generate\_password\_hash(password)

    def check\_password(self, passw):

        return check\_password\_hash(self.password, passw)

    def \_\_repr\_\_(self):

        return f""

class Roles(Dnd.Model):

    \_\_tablename\_\_ = 'roles'

    namerole = Dnd.Column(Dnd.String(), primary\_key=True)

    def \_\_init\_\_(self, namerole):

        self.namerole = namerole

    def \_\_repr\_\_(self):

        return f""

class  RoleUser(Dnd.Model):

    \_\_tablename\_\_ = 'roleuser'

    id = Dnd.Column(Dnd.Integer, primary\_key=True)

    namerole = Dnd.Column(Dnd.String(), Dnd.ForeignKey('roles.namerole'), nullable=False, index=True)

    login= Dnd.Column(Dnd.String(), Dnd.ForeignKey('customers.login'), nullable=False, index=True)

    def \_\_init\_\_(self, login, namerole):

        self.login = login

        self.namerole =  namerole

    def \_\_repr\_\_(self):

        return f""

class Hero(Dnd.Model):

    \_\_tablename\_\_ = 'hero'

    namehero = Dnd.Column(Dnd.String(), primary\_key=True)

    strength = Dnd.Column(Dnd.Integer,nullable=False)

    dexterity = Dnd.Column(Dnd.Integer,nullable=False)

    physique = Dnd.Column(Dnd.Integer,nullable=False)

    intelligence = Dnd.Column(Dnd.Integer,nullable=False)

    wisdom = Dnd.Column(Dnd.Integer,nullable=False)

    charisma = Dnd.Column(Dnd.Integer,nullable=False)

    classhero = Dnd.Column(Dnd.String(),nullable=False)

    #forename = Dnd.Column(Dnd.String())

    login= Dnd.Column(Dnd.String(), Dnd.ForeignKey('customers.login'), nullable=False, index=True)

    herolevel = Dnd.Column(Dnd.Integer,nullable=False)

    def \_\_init\_\_(self, namehero,strength,dexterity,physique,intelligence,wisdom,charisma,classhero,herolevel,login):

        self.namehero = namehero

        self.strength =  strength

        self.dexterity =  dexterity

        self.physique =  physique

        self.intelligence =  intelligence

        self.wisdom =  wisdom

        self.charisma = charisma

        self.classhero =  classhero

        self.herolevel=herolevel

        self.login=login

    def \_\_repr\_\_(self):

        return f""

class Artefact(Dnd.Model):

    \_\_tablename\_\_ = 'artefact'

    artefactname = Dnd.Column(Dnd.String(), primary\_key=True)

    discription = Dnd.Column(Dnd.String(),nullable=False)

    namehero= Dnd.Column(Dnd.String(), Dnd.ForeignKey('hero.namehero'), nullable=False, index=True)

    def \_\_init\_\_(self,artefactname, discription, namehero):

        self.artefactname = artefactname

        self.discription =  discription

        self.namehero =namehero

    def \_\_repr\_\_(self):

        return f""

class Skill(Dnd.Model):

    \_\_tablename\_\_ = 'skill'

    skill = Dnd.Column(Dnd.String(), primary\_key=True)

    discription = Dnd.Column(Dnd.String(),nullable=False)

    skilllevel= Dnd.Column(Dnd.Integer, nullable=False)

    def \_\_init\_\_(self,skill, discription, skilllevel):

        self.skill = skill

        self.discription =  discription

        self.skilllevel =skilllevel

    def \_\_repr\_\_(self):

        return f""

class SkillHero(Dnd.Model):

    \_\_tablename\_\_ = 'skillhero'

    id = Dnd.Column(Dnd.Integer, primary\_key=True)

    namehero= Dnd.Column(Dnd.String(), Dnd.ForeignKey('hero.namehero'), nullable=False, index=True)

    skill = Dnd.Column(Dnd.String(), Dnd.ForeignKey('skill.skill'), nullable=False, index=True)

    def \_\_init\_\_(self,skill,  namehero):

        self.skill = skill

        self.namehero =namehero

    def \_\_repr\_\_(self):

        return f""

class Place(Dnd.Model):

    \_\_tablename\_\_ = 'place'

    numberroom=Dnd.Column(Dnd.String, primary\_key=True)

    def \_\_init\_\_(self,numberroom ):

        self.numberroom = numberroom

    def \_\_repr\_\_(self):

        return f""

class Company(Dnd.Model):

    \_\_tablename\_\_ = 'company'

    namecompany=Dnd.Column(Dnd.String(), primary\_key=True)

    discription = Dnd.Column(Dnd.String())

    dungeonmaster = Dnd.Column(Dnd.String(),Dnd.ForeignKey('customers.login'), nullable=False, index=True)

    def \_\_init\_\_(self,namecompany,discription,dungeonmaster):

        self.namecompany = namecompany

        self.discription = discription

        self.dungeonmaster=dungeonmaster

    def \_\_repr\_\_(self):

        return f""

class Play(Dnd.Model):

    \_\_tablename\_\_ = 'play'

    id = Dnd.Column(Dnd.Integer, primary\_key=True)

    numberroom= Dnd.Column(Dnd.String, Dnd.ForeignKey('place.numberroom'), nullable=False, index=True)

    datet = Dnd.Column(Dnd.Date(), nullable=False)

    namecompany= Dnd.Column(Dnd.String(), Dnd.ForeignKey('company.namecompany'), nullable=False, index=True)

    def \_\_init\_\_(self,datet,  numberroom, namecompany):

        self.datet = datet

        self.namecompany =namecompany

        self.numberroom =numberroom

    def \_\_repr\_\_(self):

        return f""

class Book(Dnd.Model):

    \_\_tablename\_\_ = 'book'

    namebook=Dnd.Column(Dnd.String(), primary\_key=True)

    def \_\_init\_\_(self,namebook ):

        self.namebook = namebook

    def \_\_repr\_\_(self):

        return f""

class BookCompany(Dnd.Model):

    \_\_tablename\_\_ = 'bookcompany'

    id = Dnd.Column(Dnd.Integer, primary\_key=True)

    namebook= Dnd.Column(Dnd.String(), Dnd.ForeignKey('book.namebook'), nullable=False, index=True)

    namecompany= Dnd.Column(Dnd.String(), Dnd.ForeignKey('company.namecompany'), nullable=False, index=True)

    def \_\_init\_\_(self,  namebook, namecompany):

        self.namecompany =namecompany

        self.namebook =namebook

    def \_\_repr\_\_(self):

        return f""

class News(Dnd.Model):

    \_\_tablename\_\_ = 'news'

    title = Dnd.Column(Dnd.String(), primary\_key=True)

    login= Dnd.Column(Dnd.String(), Dnd.ForeignKey('customers.login'), nullable=False, index=True)

    datet = Dnd.Column(Dnd.Date(),default=datetime.utcnow())

    discription = Dnd.Column(Dnd.String(), nullable=False)

    def \_\_init\_\_(self,  title, login,datet,discription):

        self.login =login

        self.title =title

        self.datet=datet

        self.discription=discription

    def \_\_repr\_\_(self):

        return f""

class CustomersCompany(Dnd.Model):

    \_\_tablename\_\_ = 'customerscompany'

    id = Dnd.Column(Dnd.Integer, primary\_key=True)

    login= Dnd.Column(Dnd.String(), Dnd.ForeignKey('customers.login'), nullable=False, index=True)

    namecompany= Dnd.Column(Dnd.String(), Dnd.ForeignKey('company.namecompany'), nullable=False, index=True)

    def \_\_init\_\_(self,  login, namecompany):

        self.login =login

        self.namecompany =namecompany

    def \_\_repr\_\_(self):

        return f""

class CustomersPlay(Dnd.Model):

    \_\_tablename\_\_ = 'customersplay'

    id = Dnd.Column(Dnd.Integer, primary\_key=True)

    login= Dnd.Column(Dnd.String(), Dnd.ForeignKey('customers.login'), nullable=False, index=True)

    idgame= Dnd.Column(Dnd.Integer, Dnd.ForeignKey('play.id'), nullable=False, index=True)

    def \_\_init\_\_(self,  login, idgame):

        self.login =login

        self.idgame =idgame

    def \_\_repr\_\_(self):

        return f""

@login.user\_loader

def load\_user(id):

    return Customers.query.get(int(id))

## Файл forms.py:

from flask\_wtf import FlaskForm

from sqlalchemy.orm import defaultload

from sqlalchemy.sql.expression import select

from sqlalchemy.sql.functions import user

from werkzeug.utils import validate\_arguments

from wtforms import StringField, PasswordField, BooleanField, SubmitField, IntegerField,SelectField,DateField

from wtforms import validators,TextAreaField,URLField

from wtforms.validators import ValidationError, DataRequired, EqualTo

from wtforms.validators import DataRequired,Length

from app.models import \*

class LoginForm(FlaskForm):

    username = StringField('Username', validators=[DataRequired()])

    password = PasswordField('Password', validators=[DataRequired()])

    remember\_me = BooleanField('Remember Me')

    submit = SubmitField('Sign In')

class RegistrationForm(FlaskForm):

    username = StringField('Username', validators=[DataRequired()])

    password = PasswordField('Password', validators=[DataRequired()])

    password2 = PasswordField(

        'Repeat Password', validators=[DataRequired(), EqualTo('password')])

    forename = StringField('Фамилия', validators=[DataRequired()])

    surname = StringField('Имя', validators=[DataRequired()])

    submit = SubmitField('Register')

    def validate\_username(self, username):

        user = Customers.query.filter\_by(login=username.data).first()

        if user is not None:

            raise ValidationError('Please use a different username.')

class AddHeroForm(FlaskForm):

    namehero = StringField('HeroName', validators=[DataRequired()])

    strength = IntegerField('Strength', validators=[DataRequired()])

    dexterity =  IntegerField('dexterity', validators=[DataRequired()])

    physique =  IntegerField('physique', validators=[DataRequired()])

    intelligence =  IntegerField('intelligence', validators=[DataRequired()])

    wisdom = IntegerField('wisdom', validators=[DataRequired()])

    charisma =  IntegerField('charisma', validators=[DataRequired()])

    classhero = SelectField('classhero', validators=[DataRequired()], choices=[('bard', 'Бард'),

    ('Barbarian', 'Варвар'), ('Fighter', 'Воин'),("Wizard","Волшебник"),("Druid","Друид"),("Cleric","Жнец"),

    ('Artificer', 'Изобретатель'), ('Warlock', 'Колдун'),("Monk","Монах"),("Paladin","Паладин"),

    ("Rogue","Плут"),("Ranger","Следопыт"),("Sorcerer","Чародей")

    ])

    herolevel =  IntegerField('Herolevel', validators=[DataRequired()])

    submit = SubmitField('CreateHero')

    def validate\_namehero(self, namehero):

        user = Hero.query.filter\_by(namehero=namehero.data).first()

        if user is not None:

            raise ValidationError('Please use a different heroname.')

class AddBookForm(FlaskForm):

    namebook = StringField('Namebook', validators=[DataRequired()])

    submit = SubmitField('CreateBook')

    def validate\_namebook(self, namebook):

        book = Book.query.filter\_by(namebook=namebook.data).first()

        if book is not None:

            raise ValidationError('Please use a different namebook.')

class AddSkillForm(FlaskForm):

    skill = StringField('Skill', validators=[DataRequired()])

    discription = TextAreaField('discription', validators=[DataRequired(),Length(max=1500)])

    skilllevel=IntegerField('dexterity', validators=[DataRequired()])

    submit = SubmitField('CreateSkill')

    def validate\_skill(self, skill):

        skill = Skill.query.filter\_by(skill=skill.data).first()

        if skill is not None:

            raise ValidationError('Please use a different skill.')

class AddPlaceForm(FlaskForm):

    place = IntegerField('Place', validators=[DataRequired()])

    submit = SubmitField('CreatePlace')

    def validate\_namebook(self, place):

        places = Place.query.filter\_by(place=place.data).first()

        if places is not None:

            raise ValidationError('Please use a different place.')

class AddNewsForm(FlaskForm):

    title=StringField("Title",validators=[DataRequired()])

    discription = TextAreaField('discription', validators=[DataRequired(),Length(max=1500)])

    submit = SubmitField('CreateNews')

    def validate\_title(self, title):

        titeles = News.query.filter\_by(title=title.data).first()

        if titeles is not None:

            raise ValidationError('Please use a different title.')

class GiveRoleForm(FlaskForm):

    namerole=SelectField('namerole',validators=[DataRequired()], choices=[('Dangeon\_master', 'Мастер')])

    user =  SelectField("Выберите имя пользователя, которому вы хотите выдать роль",validators=[DataRequired()])

    submit = SubmitField('GiveRole')

    def validate\_user(self,user):

        titeles = Customers.query.filter\_by(login=user.data).first()

        if titeles is None:

            raise ValidationError('Такого пользователя не существует')

        roleusers=RoleUser.query.filter\_by(login=user.data,namerole='Dangeon\_master').first()

        if roleusers is not None:

            raise ValidationError('Данный пользователь уже является мастером')

class ViewMoreInfoHeroForm(FlaskForm):

    namehero=SelectField('Выберите персонажа, о котором хотите юзнать больше',validators=[DataRequired()])

    submit=SubmitField('SeeMoreinfo')

    def validate\_namehero(self,namehero):

        heroes = Hero.query.filter\_by(namehero=namehero.data).first()

        if heroes is None:

            raise ValidationError('Такого героя не существует')

class InfoHero(FlaskForm):

    strength = IntegerField('Strength', validators=[DataRequired()])

    dexterity =  IntegerField('dexterity', validators=[DataRequired()])

    physique =  IntegerField('physique', validators=[DataRequired()])

    intelligence =  IntegerField('intelligence', validators=[DataRequired()])

    wisdom = IntegerField('wisdom', validators=[DataRequired()])

    charisma =  IntegerField('charisma', validators=[DataRequired()])

    herolevel =  IntegerField('Herolevel', validators=[DataRequired()])

    submit = SubmitField('EditHero')

class AddCompanyForm(FlaskForm):

    namecompany = StringField('Company', validators=[DataRequired()])

    dangeonmaster=SelectField('Dangeon\_master',validators=[DataRequired()])

    discription = TextAreaField('discription', validators=[DataRequired(),Length(max=500)])

    submit = SubmitField('CreateCompany')

    def validate\_dangeonmaster(self,dangeonmaster):

        titeles = RoleUser.query.filter\_by(login=dangeonmaster.data,namerole='Dangeon\_master').first()

        if titeles is None:

            raise ValidationError('Такого пользователя не существует')

    def validate\_namecompany(self,namecompany):

        company = Company.query.filter\_by(namecompany=namecompany.data).first()

        if company is not None:

            raise ValidationError('Такая компания уже существует')

class AddBookCompanyForm(FlaskForm):

    namecompany = SelectField('Company', validators=[DataRequired()])

    namebook = SelectField('Namebook', validators=[DataRequired()])

    submit = SubmitField('AddBook')

    def validate\_namecompany(self,namecompany):

        company = Company.query.filter\_by(namecompany=namecompany.data).first()

        if company is None:

            raise ValidationError('Такая компания не существует')

        #namebook=namebook.data

        #bookcompany=BookCompany.query.filter\_by(namecompany=namecompany.data, namebook=namebook.data).first()

        #if bookcompany is not None:

        #    raise ValidationError('Данный пользователь уже является мастером')

class AddPlayerInCompanyForm(FlaskForm):

    namecompany = SelectField('Company', validators=[DataRequired()])

    player =  SelectField("Выберите имя пользователя, которого вы хотите добавить в компанию",validators=[DataRequired()])

    submit = SubmitField('AddPlayer')

    def validate\_user(self,player):

        titeles = Customers.query.filter\_by(login=user.data).first()

        if titeles is None:

            raise ValidationError('Такого пользователя не существует')

    def validate\_namecompany(self,namecompany):

        company = Company.query.filter\_by(namecompany=namecompany.data).first()

        if company is None:

            raise ValidationError('Такая компания не существует')

class AddPlayForm(FlaskForm):

    namecompany = SelectField('Company', validators=[DataRequired()])

    numberroom =  SelectField("Выберите место проведения",validators=[DataRequired()])

    datet=DateField('Выберите дату',validators=[DataRequired()])

    submit = SubmitField('AddPlay')

    def validate\_numberroom(self,numberroom):

        titeles = Place.query.filter\_by(numberroom=numberroom.data).first()

        if titeles is None:

            raise ValidationError('Такого пользователя не существует')

    def validate\_namecompany(self,namecompany):

        company = Company.query.filter\_by(namecompany=namecompany.data).first()

        if company is None:

            raise ValidationError('Такая компания не существует')

class AddPlayerPlayForm(FlaskForm):

    login = SelectField('Customers', validators=[DataRequired()])

    idgame =  SelectField("Выберите № игры",validators=[DataRequired()])

    submit = SubmitField('AddPlayer')

    def validate\_login(self,login):

        titeles = Customers.query.filter\_by(login=login.data).first()

        if titeles is None:

            raise ValidationError('Такого пользователя не существует')

    def validate\_idgame(self,idgame):

        idgames = Play.query.filter\_by(id=idgame.data).first()

        if idgames is None:

            raise ValidationError('Такая компания не существует')

class GiveArtefact(FlaskForm):

    namehero = SelectField('Имя персонажа', validators=[DataRequired()])

    artefact =  StringField("Artefact",validators=[DataRequired()])

    discription = TextAreaField('discription',validators=[DataRequired()])

    submit = SubmitField('GiveArtefact')

    def validate\_namehero(self,namehero):

        titeles = Hero.query.filter\_by(namehero=namehero.data).first()

        if titeles is None:

            raise ValidationError('Такого пользователя не существует')

    def validate\_artefact(self,artefact):

        artefacts = Artefact.query.filter\_by(artefactname=artefact.data).first()

        if artefacts is not None:

            raise ValidationError('Такой артефакт уже выдан')

class AddSkillHeroForm(FlaskForm):

    skill=SelectField('Skill',validators=[DataRequired()])

    submit = SubmitField('AddSkill')

    def validate\_skill(self, skill):

        skills = Skill.query.filter\_by(skill=skill.data).first()

        if skills is not None:

            raise ValidationError('Такой скилл уже есть у персонажа')

class EditorForm(FlaskForm):

    namecompany=SelectField('Company',validators=[DataRequired()])

    submit=SubmitField('SeeMoreInfoCompany')

    def validate\_namecompany(self, namecompany):

        skills = Company.query.filter\_by(namecompany=namecompany.data).first()

        if skills is None:

            raise ValidationError('Такой компании нет')

class EditCompanyForm(FlaskForm):

    discription = TextAreaField('discription', validators=[DataRequired(),Length(max=500)])

    submit=SubmitField('Update\_company')

class Admin(FlaskForm):

    admin=URLField('Admin')

## Файл \_init\_.py:

from flask\_login import LoginManager

from flask import Flask

from config import Config

from flask\_migrate import Migrate

from flask\_sqlalchemy import SQLAlchemy

app = Flask(\_\_name\_\_)

app.config.from\_object(Config)

app.config['SQLALCHEMY\_DATABASE\_URI'] = "postgresql+psycopg2://postgres:15780@localhost:5432/Dnd3"

Dnd = SQLAlchemy(app)

migrate = Migrate(app, Dnd)

title='Dnd'

login = LoginManager(app)

login.login\_view = 'login'

from app import routes, models

## Файл config.py:

import os

class Config(object):

    SECRET\_KEY = os.environ.get('SECRET\_KEY') or 'you-will-never-guess'

## Файл addbook.html:

{% extends "base.html" %}

{% block content %}

    <h1>Добавление Книги</h1>

    <form action="" method="post">

        {{ form.hidden\_tag() }}

        <p>

            {{ form.namebook.label }}<br>

            {{ form.namebook(size=32) }}<br>

            {% for error in form.namebook.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

    <p>{{ form.submit() }}</p>

</form>

{% endblock %}

## Файл addbookcompany.html:

{% extends "base.html" %}

{% block content %}

    <h1>Добавление Книги в компанию</h1>

    <a href="{{ url\_for('editor') }}">Назад</a>

    <form action="" method="post">

        {{ form.hidden\_tag() }}

        <p>

            {{ form.namebook.label }}<br>

            {{ form.namebook(size=1) }}<br>

            {% for error in form.namebook.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.namecompany.label }}<br>

            {{ form.namecompany(size=1) }}<br>

            {% for error in form.namecompany.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

    <p>{{ form.submit() }}</p>

</form>

{% endblock %}

## Файл addcompany.html:

{% extends "base.html" %}

{% block content %}

    <h1>Создание компании</h1>

    <form action="" method="post">

        {{ form.hidden\_tag() }}

        <p>

            {{ form.namecompany.label }}<br>

            {{ form.namecompany(size=32) }}<br>

            {% for error in form.namecompany.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.dangeonmaster.label }}<br>

            {{ form.dangeonmaster(size=1) }}<br>

            {% for error in form.dangeonmaster.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.discription.label }}<br>

            {{ form.discription(cols=90, rows=30) }}<br>

            {% for error in form.discription.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>{{ form.submit() }}</p>

    </form>

{% endblock %}

## Файл addhero.html:

{% extends "base.html" %}

{% block content %}

<a href="{{ url\_for('user', login=current\_user.login) }}">Назад</a>

    <h1>Добавление персонажа</h1>

    <form action="" method="post">

        {{ form.hidden\_tag() }}

        <p>

            {{ form.namehero.label }}<br>

            {{ form.namehero(size=32) }}<br>

            {% for error in form.namehero.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.classhero.label }}<br>

            {{ form.classhero(size=1) }}<br>

            {% for error in form.classhero.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.herolevel.label }}<br>

            {{ form.herolevel(max=20, min=1) }}<br>

            {% for error in form.herolevel.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.strength.label }}<br>

            {{ form.strength(max=24, min=1) }}<br>

            {% for error in form.strength.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.dexterity.label }}<br>

            {{ form.dexterity(max=24, min=1) }}<br>

            {% for error in form.dexterity.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.physique.label }}<br>

            {{ form.physique(max=24, min=1) }}<br>

            {% for error in form.physique.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.intelligence.label }}<br>

            {{ form.intelligence(max=24, min=1) }}<br>

            {% for error in form.intelligence.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.wisdom.label }}<br>

            {{ form.wisdom(max=24, min=1) }}<br>

            {% for error in form.wisdom.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.charisma.label }}<br>

            {{ form.charisma(max=24, min=1) }}<br>

            {% for error in form.charisma.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>{{ form.submit() }}</p>

    </form>

{% endblock %}

## Файл addnews.html:

{% extends "base.html" %}

{% block content %}

    <h1>Добавление Новости</h1>

    <form action="" method="post">

        {{ form.hidden\_tag() }}

        <p>

            {{ form.title.label }}<br>

            {{ form.title(size=32) }}<br>

            {% for error in form.title.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.discription.label }}<br>

            {{ form.discription(cols=90, rows=30)}}<br>

            {% for error in form.discription.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

    <p>{{ form.submit() }}</p>

</form>

{% endblock %}

## Файл addplace.html:

{% extends "base.html" %}

{% block content %}

    <h1>Добавление места проведения</h1>

    <form action="" method="post">

        {{ form.hidden\_tag() }}

        <p>

            {{ form.place.label }}<br>

            {{ form.place(min=0) }}<br>

            {% for error in form.place.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

    <p>{{ form.submit() }}</p>

</form>

{% endblock %}

## Файл addplay.html:

{% extends "base.html" %}

{% block content %}

<a href="{{ url\_for('editor') }}">Назад</a>

    <h1>Добавление игры</h1>

    <form action="" method="post">

        {{ form.hidden\_tag() }}

        <p>

            {{ form.namecompany.label }}<br>

            {{ form.namecompany(size=1) }}<br>

            {% for error in form.namecompany.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.numberroom.label }}<br>

            {{ form.numberroom(size=1) }}<br>

            {% for error in form.numberroom.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.datet.label }}<br>

            {{ form.datet(size=1) }}<br>

            {% for error in form.datet.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

    <p>{{ form.submit() }}</p>

</form>

{% endblock %}

## Файл addplayercompany.html:

{% extends "base.html" %}

{% block content %}

    <h1>Добавление Игрока в компанию</h1>

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

    <form action="" method="post">

        {{ form.hidden\_tag() }}

        <p>

            {{ form.player.label }}<br>

            {{ form.player(size=1) }}<br>

            {% for error in form.player.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.namecompany.label }}<br>

            {{ form.namecompany(size=1) }}<br>

            {% for error in form.namecompany.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

    <p>{{ form.submit() }}</p>

</form>

{% endblock %}

## Файл addplayerplay.html:

{% extends "base.html" %}

{% block content %}

<a href="{{ url\_for('editor') }}">Назад</a>

    <h1>Добавление игрока в игру</h1>

    <form action="" method="post">

        {{ form.hidden\_tag() }}

        <p>

            {{ form.login.label }}<br>

            {{ form.login(size=1) }}<br>

            {% for error in form.login.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.idgame.label }}<br>

            {{ form.idgame(size=1) }}<br>

            {% for error in form.idgame.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

    <p>{{ form.submit() }}</p>

</form>

{% endblock %}

## Файл addskill.html:

{% extends "base.html" %}

{% block content %}

    <h1>Добавление способности</h1>

    <form action="" method="post">

        {{ form.hidden\_tag() }}

        <p>

            {{ form.skill.label }}<br>

            {{ form.skill(size=32) }}<br>

            {% for error in form.skill.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.skilllevel.label }}<br>

            {{ form.skilllevel(max=9, min=0) }}<br>

            {% for error in form.skilllevel.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.discription.label }}<br>

            {{ form.discription(cols=90, rows=30) }}<br>

            {% for error in form.discription.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

    <p>{{ form.submit() }}</p>

</form>

{% endblock %}

## Файл addskillhero.html:

{% extends "base.html" %}

{% block content %}

    <p><a href="{{ url\_for('user',login=current\_user.login) }}">Назад</a></p>

    <hr>

    <form action="" method="post">

        {{ form.hidden\_tag() }}

        <p>

            {{ form.skill.label }}<br>

            {{ form.skill(size=1) }}<br>

            {% for error in form.skill.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>{{ form.submit() }}</p>

    </form>

{% endblock %}

## Файл admin.html:

{% extends "base.html" %}

{% block content %}

    <div>

    DndPlay Create:

        <p><a href="{{ url\_for('addbook') }}">Добавить книгу</a></p>

        <p><a href="{{ url\_for('addskill') }}">Добавить скилл</a></p>

        <p><a href="{{ url\_for('addplace') }}">Добавить место</a></p>

        <p><a href="{{ url\_for('addnews') }}">Добавить новость</a></p>

        <p><a href="{{url\_for('giverole')}}">Выдать роль</a></p>

        <p><a href="{{url\_for('addcompanyadmin')}}">Добавить компанию</a></p>

        <p><a href="{{url\_for('addplayeradmin')}}">Добавить игрока в компанию</a></p>

    </div>

    <form action="" method="post">

        {{ form.hidden\_tag() }}

        <p>

            {{ form.namecompany.label }}<br>

            {{ form.namecompany(size=1) }}<br>

            {% for error in form.namecompany.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>{{ form.submit() }}</p>

    </form>

{% endblock %}

## Файл base.html:

<html>

    <head>

      {% if title %}

      <title>{{ title }} - DndPlay</title>

      {% else %}

      <title>Welcome to DndPlay</title>

      {% endif %}

    </head>

    <body>

        <a href="{{ url\_for('admin') }}">admin</a>

        <a href="{{ url\_for('editor') }}">editor</a>

        <div>

            DndPlay:

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

            {% if current\_user.is\_anonymous %}

            <a href="{{ url\_for('login') }}">Login</a>

            {% else %}

            <a href="{{url\_for('viewbook')}}">Посмотреть книги</a>

            <a href="{{ url\_for('user', login=current\_user.login) }}">Profile</a>

            <a href="{{ url\_for('logout') }}">Logout</a>

            {% endif %}

        </div>

        <hr> <hr>

        {% with messages = get\_flashed\_messages() %}

        {% if messages %}

        <ul>

            {% for message in messages %}

            <li>{{ message }}</li>

            {% endfor %}

        </ul>

        {% endif %}

        {% endwith %}

        {% block content %}{% endblock %}

    </body>

</html>

## Файл editor.html:

{% extends "base.html" %}

{% block content %}

    <div>

    DndPlay Create:

        <p><a href="{{url\_for('addcompanyeditor')}}">Добавить свою компанию</a></p>

        <p><a href="{{url\_for('addbookcompany')}}">Добавить книгу в компанию</a></p>

        <p><a href="{{url\_for('addplayercompany')}}">Добавить игрока в компанию</a></p>

        <p><a href="{{url\_for('addplay')}}">Добавить игру</a></p>

        <p><a href="{{url\_for('addplayerplay')}}">Добавить игрока в прошедшую игру</a></p>

        <p><a href="{{url\_for('giveartefact')}}">Выдать артефакт</a></p>

    </div>

    <form action="" method="post">

        {{ form.hidden\_tag() }}

        <p>

            {{ form.namecompany.label }}<br>

            {{ form.namecompany(size=1) }}<br>

            {% for error in form.namecompany.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>{{ form.submit() }}</p>

    </form>

{% endblock %}

## Файл giveartefact.html:

{% extends "base.html" %}

{% block content %}

    <a href="{{ url\_for('editor') }}">Назад</a>

    <h1>Выдача артефакта персонажу</h1>

     <form action="" method="post">

            {{ form.hidden\_tag() }}

                <p>

                    {{ form.namehero.label }}<br>

                    {{ form.namehero(size=1) }}<br>

                    {% for error in form.namehero.errors %}

                    <span style="color: red;">[{{ error }}]</span>

                    {% endfor %}

                </p>

            <p>

                {{ form.artefact.label }}<br>

                    {{ form.artefact(size=32) }}<br>

                    {% for error in form.artefact.errors %}

                    <span style="color: red;">[{{ error }}]</span>

                    {% endfor %}

            </p>

            <p>

                {{ form.discription.label }}<br>

                {{ form.discription(cols=90, rows=30) }}<br>

                {% for error in form.discription.errors %}

                <span style="color: red;">[{{ error }}]</span>

                {% endfor %}

            </p>

            <p>{{ form.submit() }}</p>

    </form>

{% endblock %}

## Файл giverole.html:

{% extends "base.html" %}

<a href="{{ url\_for('admin') }}">Назад</a>

{% block content %}

    <h1>Добавление персонажа</h1>

     <form action="" method="post">

            {{ form.hidden\_tag() }}

                <p>

                    {{ form.namerole.label }}<br>

                    {{ form.namerole(size=1) }}<br>

                    {% for error in form.namerole.errors %}

                    <span style="color: red;">[{{ error }}]</span>

                    {% endfor %}

                </p>

            <p>

                {{ form.user.label }}<br>

                    {{ form.user(size=1) }}<br>

                    {% for error in form.user.errors %}

                    <span style="color: red;">[{{ error }}]</span>

                    {% endfor %}

            </p>

            <p>{{ form.submit() }}</p>

    </form>

    {% endblock %}

## Файл hero.html:

{% extends "base.html" %}

{% block content %}

<a href="{{ url\_for('user', login=current\_user.login) }}">Назад</a>

    <h1>{{hero.namehero}}</h1>

    <h2>{{hero.classhero}}</h2>

    <form action="" method="post">

        {{ form.hidden\_tag() }}

        <p>

            {{ form.herolevel.label }}<br>

            {{ form.herolevel(max=20, min=1) }}<br>

            {% for error in form.herolevel.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.strength.label }}<br>

            {{ form.strength(max=24, min=1) }}<br>

            {% for error in form.strength.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.dexterity.label }}<br>

            {{ form.dexterity(max=24, min=1) }}<br>

            {% for error in form.dexterity.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.physique.label }}<br>

            {{ form.physique(max=24, min=1) }}<br>

            {% for error in form.physique.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.intelligence.label }}<br>

            {{ form.intelligence(max=24, min=1) }}<br>

            {% for error in form.intelligence.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.wisdom.label }}<br>

            {{ form.wisdom(max=24, min=1) }}<br>

            {% for error in form.wisdom.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.charisma.label }}<br>

            {{ form.charisma(max=24, min=1) }}<br>

            {% for error in form.charisma.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>{{ form.submit() }}</p>

    </form>

    <h3>Skill's</h3>

    <div>

        {% for skill in skills%}

            <p>{{skill[0]}}</p>

        {%endfor%}

    </div>

    <p><a href="{{ url\_for('addskillhero', login=current\_user.login, namehero=hero.namehero) }}">Добавить скилл</a></p>

    <h3>Artefacts's</h3>

    <div>

        {% for artefact in artefacts%}

            <p>{{artefact[0]}}</p>

        {%endfor%}

    </div>

    {% endblock %}

## Файл index.html:

{% extends "base.html" %}

{% block content %}

    <h1>Hi, {{ current\_user.surname}}!</h1>

    <h1>Новости и объявления</h1>

    {% for new in news %}

    <h1>{{new.title}} </h1>

    <div><h2>Автор: {{ new.login }}</h2>

        <p>{{new.discription}}</p>

        <p>{{new.datet}}</p>

    </div>

    {% endfor %}

{% endblock %}

## Файл infocompany.html:

{% extends "base.html" %}

{% block content %}

    <p><a href="{{ url\_for('index')}}">На главную</a></p>

    <hr>

    <h1>Название компании :{{namecompany}}</h1>

    <h2>Мастер: {{dangeonmaster}}</h2>

    <form action="" method="post">

        {{ form.hidden\_tag() }}

        <p>

            {{ form.discription.label }}<br>

            {{ form.discription(cols=90, rows=30)}}<br>

            {% for error in form.discription.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>{{ form.submit() }}</p>

    </form>

    <h3>Players</h3>

    <div>

        {% for player in players%}

            <p>{{player[0]}}</p>

        {%endfor%}

    </div>

{% endblock %}

## Файл login.html:

{% extends "base.html" %}

{% block content %}

    <h1>Sign In</h1>

    <form action="" method="post"  novalidate>

        {{ form.hidden\_tag() }}

        <p>

            {{ form.username.label }}<br>

            {{ form.username(size=32) }}<br>

            {% for error in form.username.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.password.label }}<br>

            {{ form.password(size=32) }}<br>

            {% for error in form.password.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>{{ form.remember\_me() }} {{ form.remember\_me.label }}</p>

        <p>{{ form.submit() }}</p>

    </form>

    <p>New User? <a href="{{ url\_for('register') }}">Click to Register!</a></p>

 {% endblock %}

## Файл mycompany.html:

{% extends "base.html" %}

{% block content %}

{% for game in namegame %}

    <h1>Название компании: {{game[0]}}</h1>

    <h2>Мастер: {{game[1]}}</h2>

    <h3>Описание: {{game[2]}}</h3>

{%endfor%}

{% endblock %}

## Файл register.html:

{% extends "base.html" %}

{% block content %}

    <h1>Register</h1>

    <form action="" method="post">

        {{ form.hidden\_tag() }}

        <p>

            {{ form.username.label }}<br>

            {{ form.username(size=32) }}<br>

            {% for error in form.username.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.forename.label }}<br>

            {{ form.forename(size=32) }}<br>

            {% for error in form.forename.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.surname.label }}<br>

            {{ form.surname(size=32) }}<br>

            {% for error in form.surname.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.password.label }}<br>

            {{ form.password(size=32) }}<br>

            {% for error in form.password.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>

            {{ form.password2.label }}<br>

            {{ form.password2(size=32) }}<br>

            {% for error in form.password2.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>{{ form.submit() }}</p>

    </form>

{% endblock %}

## Файл storygame.html:

{% extends "base.html" %}

{% block content %}

{% for game in namegame %}

    <h1>Название компании: {{game[0]}}</h1>

    <h2>Дата: {{game[1]}}</h2>

    <h3>Место проведения: {{game[2]}}</h3>

{%endfor%}

{% endblock %}

## Файл user.html:

{% extends "base.html" %}

{% block content %}

    <h1>User: {{ user.login }}</h1>

    <p><a href="{{ url\_for('addhero',login=current\_user.login) }}">Добавление персонажа!</a></p>

    <p><a href="{{ url\_for('storygame',login=current\_user.login) }}">История игры!</a></p>

    <p><a href="{{ url\_for('mycompany',login=current\_user.login) }}">Компании в которых я состою!</a></p>

    <hr>

    <h2>Перонажи</h2>

    <form action="" method="post">

        {{ form.hidden\_tag() }}

        <p>

            {{ form.namehero.lable }}<br>

            {{ form.namehero(size=1) }}<br>

            {% for error in form.namehero.errors %}

            <span style="color: red;">[{{ error }}]</span>

            {% endfor %}

        </p>

        <p>{{ form.submit() }}</p>

    </form>

{% endblock %}

## Файл viewbook.html:

{% extends "base.html" %}

{% block content %}

<h3>Book's</h3>

    <div>

        {% for book in books%}

            <p>{{book[0]}}</p>

        {%endfor%}

    </div>

{% endblock %}

# Описание применения приложения со снимками экрана

Возможности пользователя:

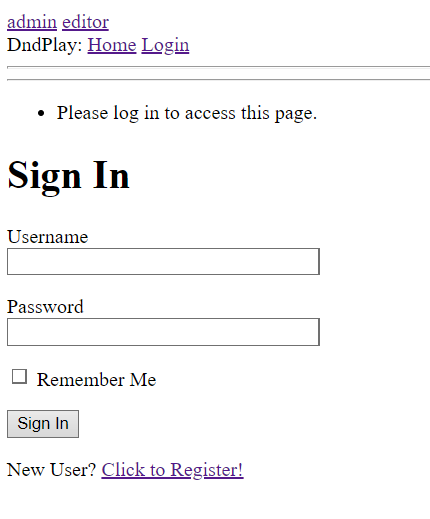


Рисунок 2 – Страница неавторизованного пользователя

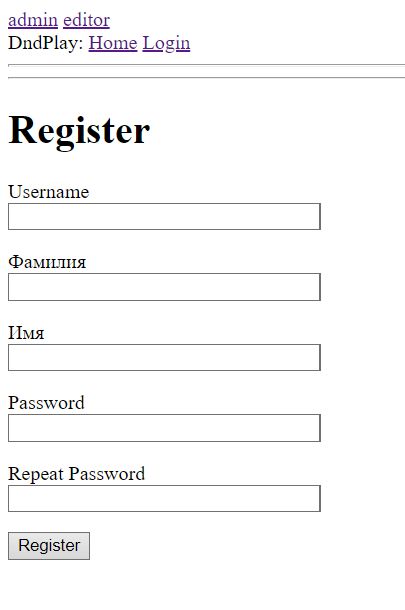


Рисунок 3 – Страница регистрации

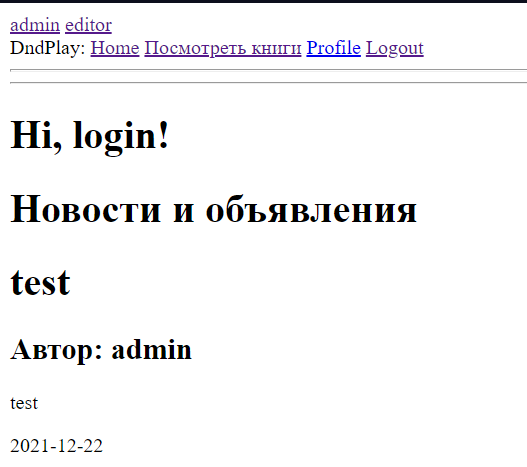


Рисунок 4 – Главная страница после авторизации

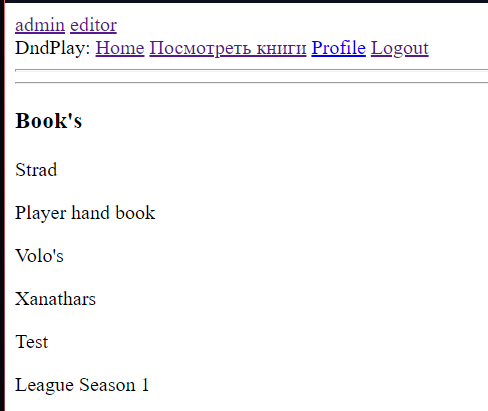


Рисунок 5 – Страница просмотра списка книг

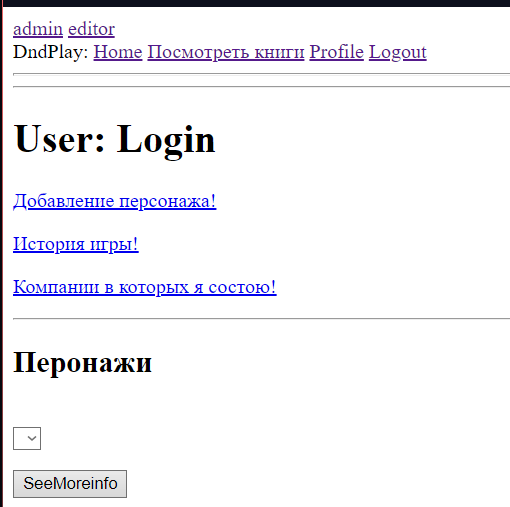


Рисунок 6 – Страница профиля пользователя

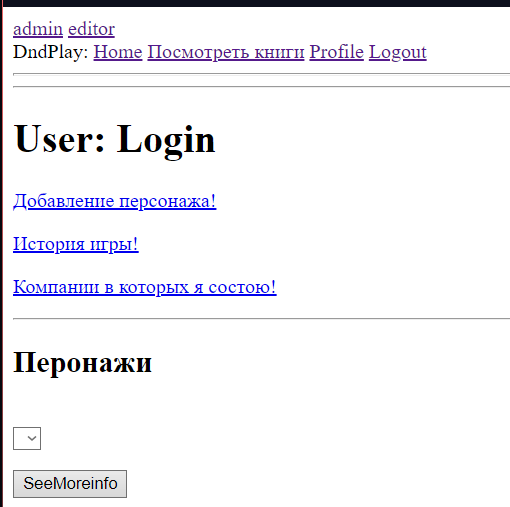


Рисунок 7 – Страница добавления персонажа

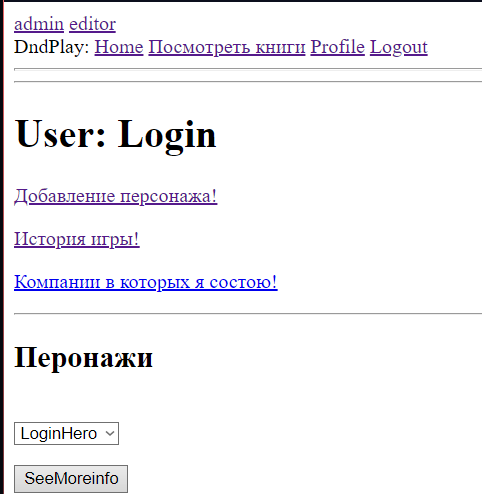


Рисунок 8 – Страница профиля пользователя после добавления персонажа

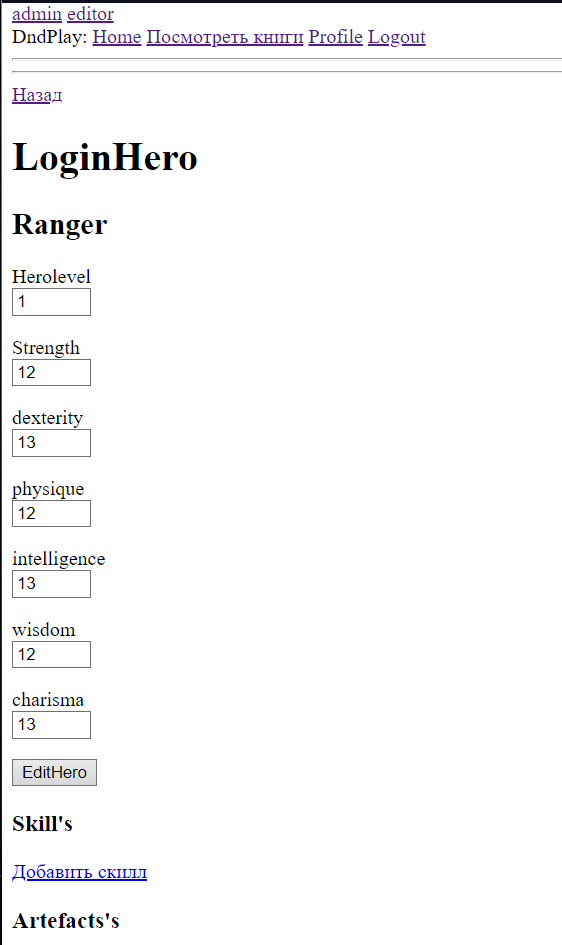


Рисунок 9 – Страница информации о герое

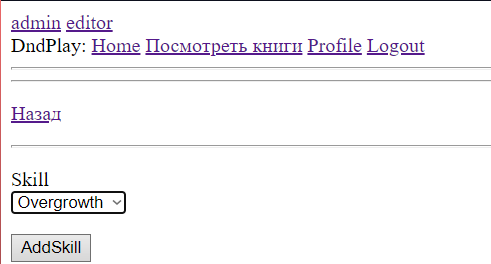


Рисунок 10 – Страница добавления скилла персонажу

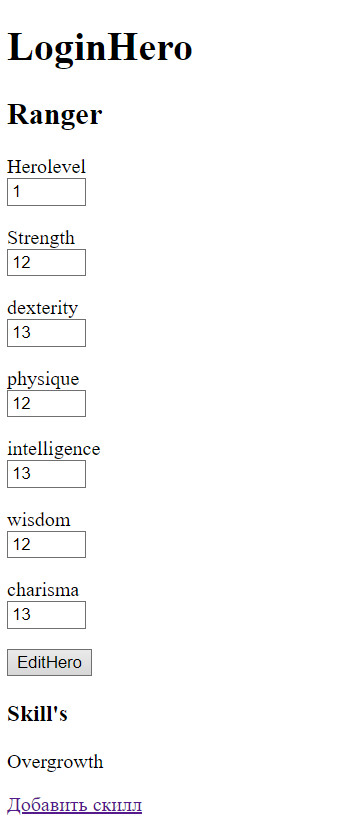


Рисунок 10 – Страница информации о персонаже после добавления скилла персонажу

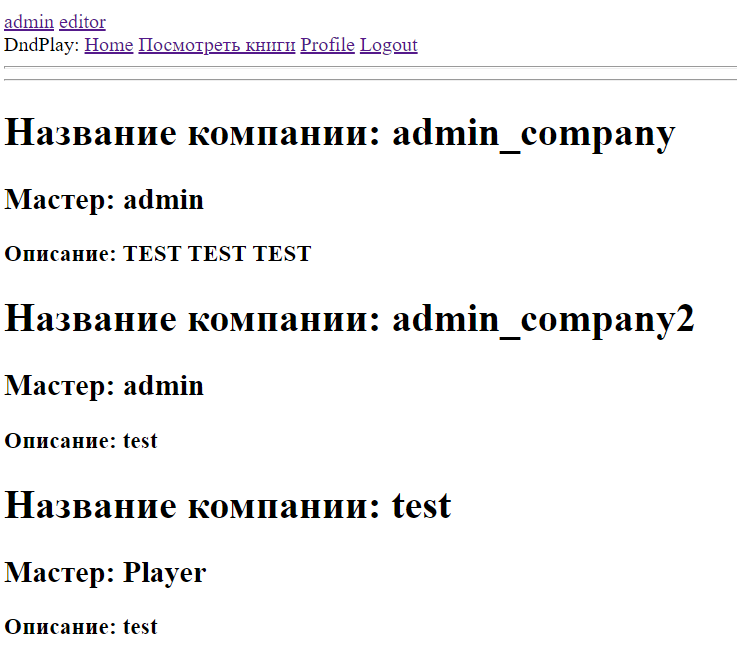


Рисунок 11 – Страница компаний, в которых состоит пользователь

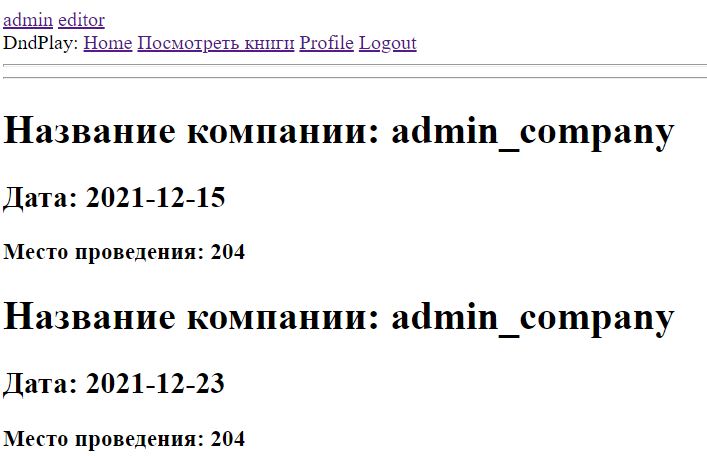


Рисунок 12 – Страница историй игр пользователя

## Возможности мастера:

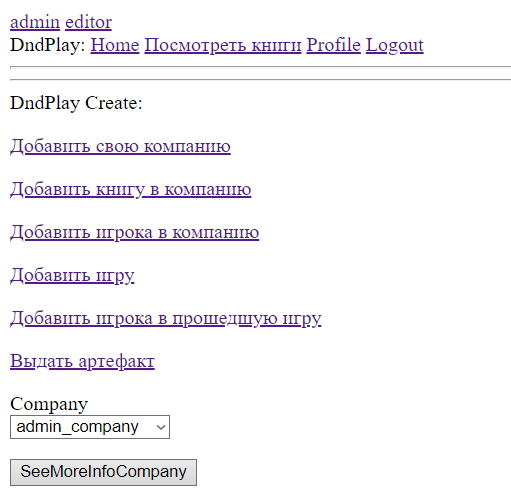


Рисунок 13 – Страница мастера

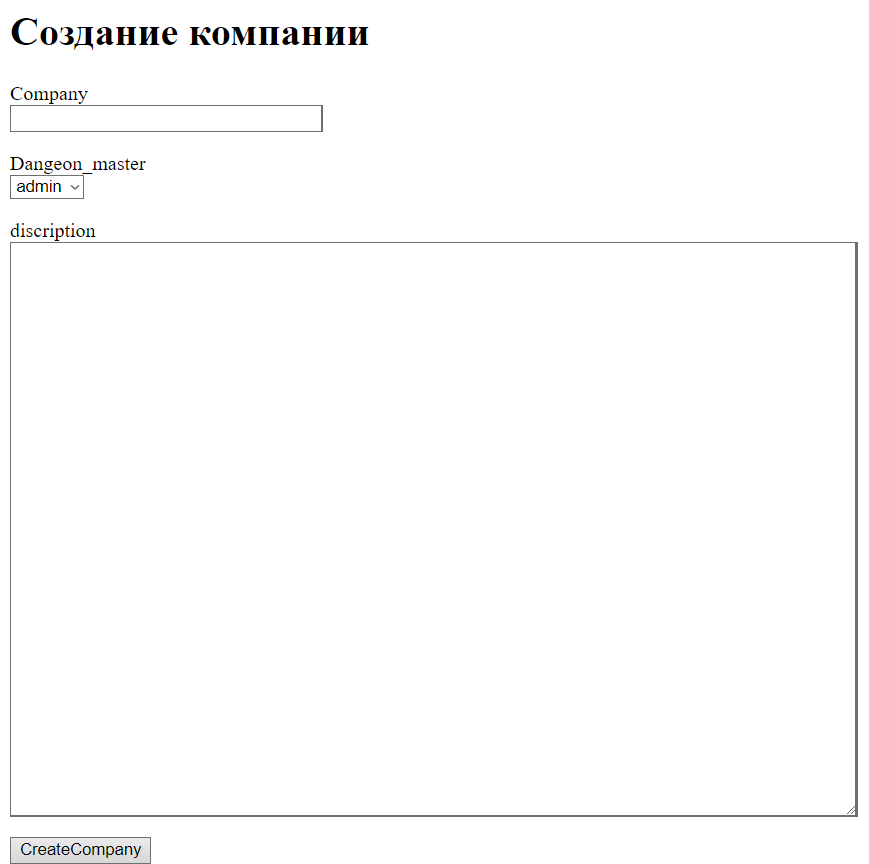


Рисунок 14 – Страница создания компании

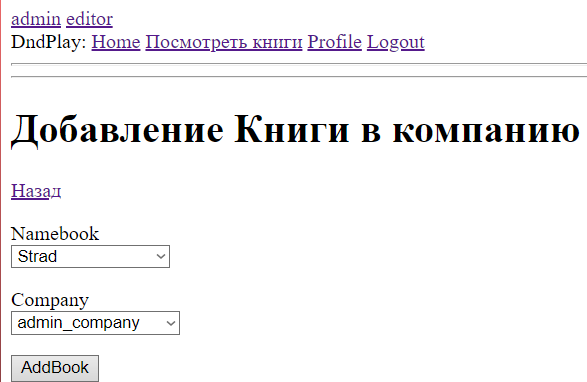


Рисунок 15 – Страница добавления книги в компанию

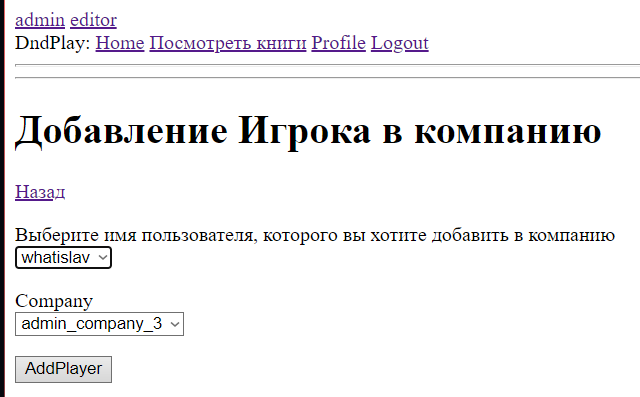


Рисунок 16 – Страница добавления игрока в компанию

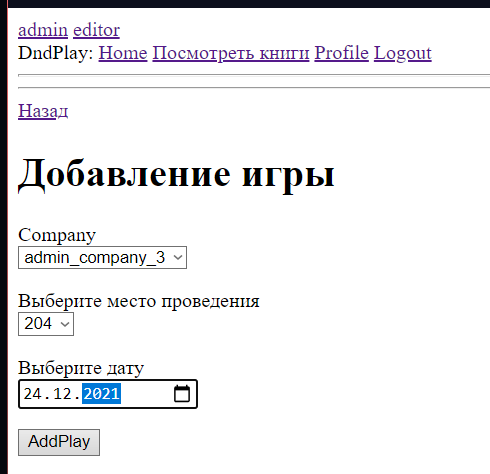


Рисунок 17 – Страница добавления игры

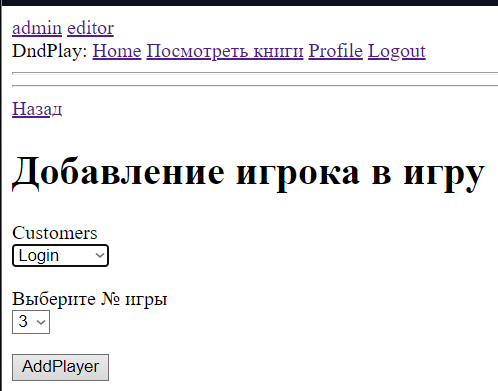


Рисунок 18 – Страница добавления игрока в игру

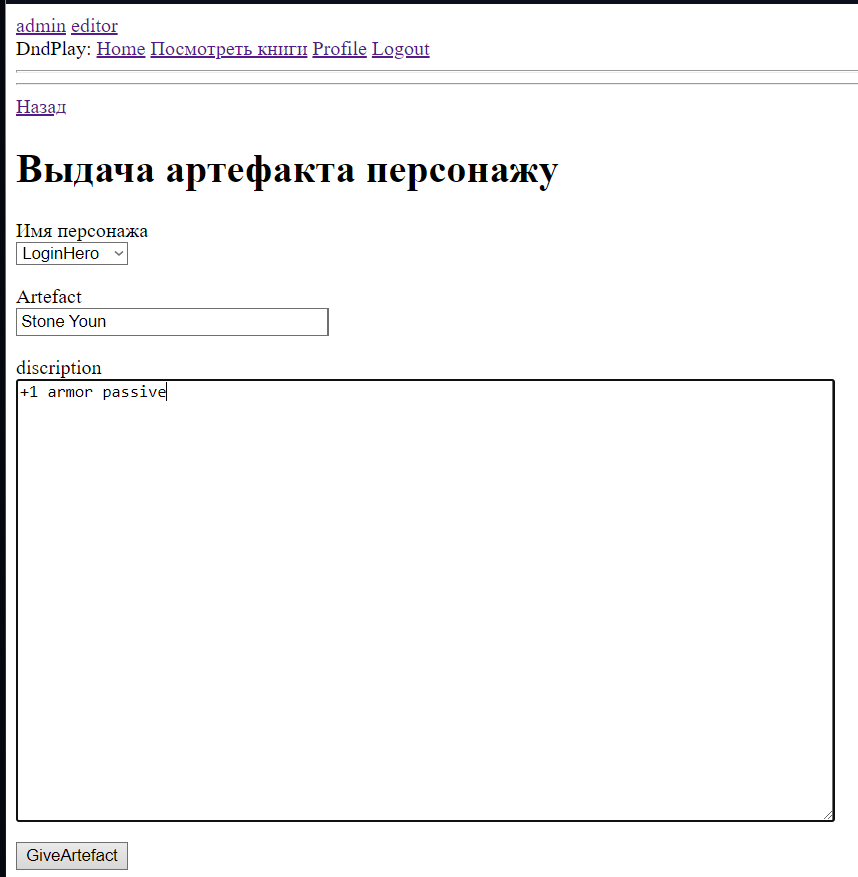


Рисунок 19 – Страница выдачи артефакта персонажу



Рисунок 20 – Страница выдачи артефакта персонажу

## Возможности Админа:

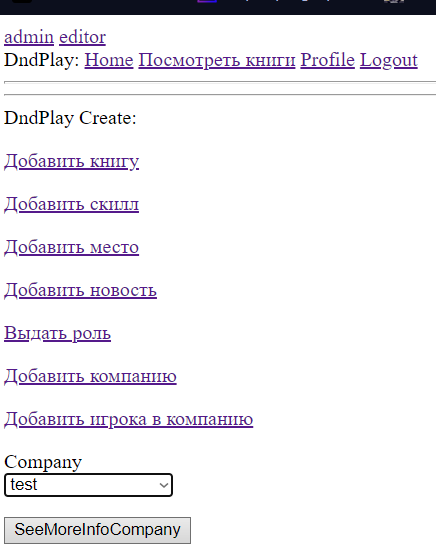


Рисунок 21 – Админ панель

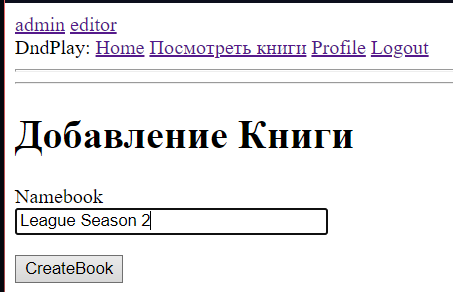


Рисунок 22 – Страница добавления книги

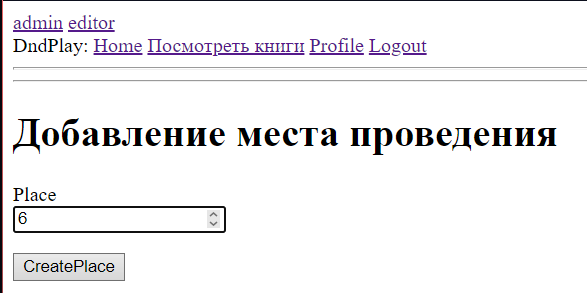


Рисунок 23 – Страница добавления места проведения

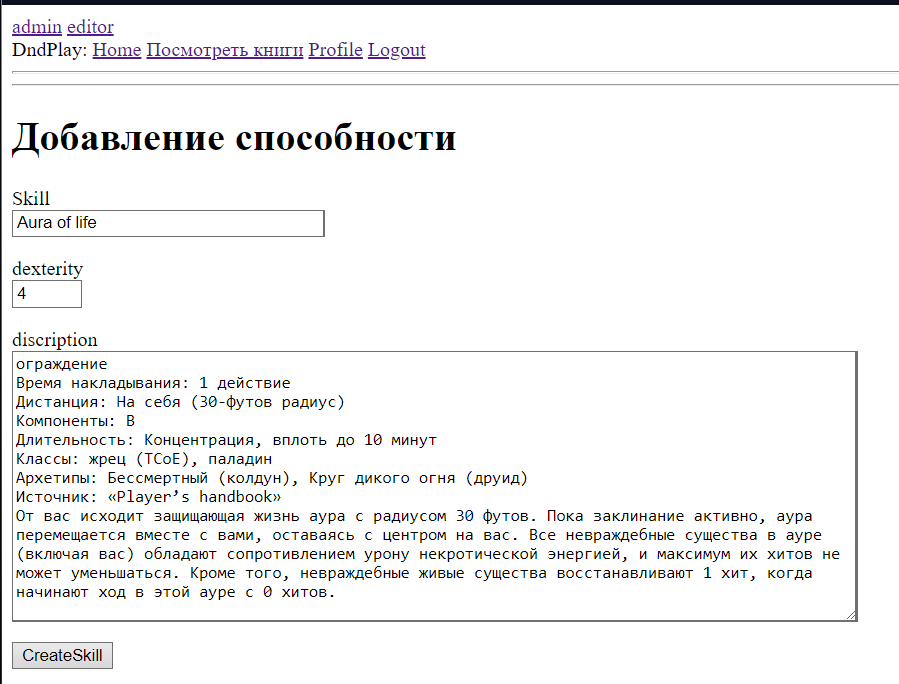


Рисунок 24 – Страница добавления способности(скилла)

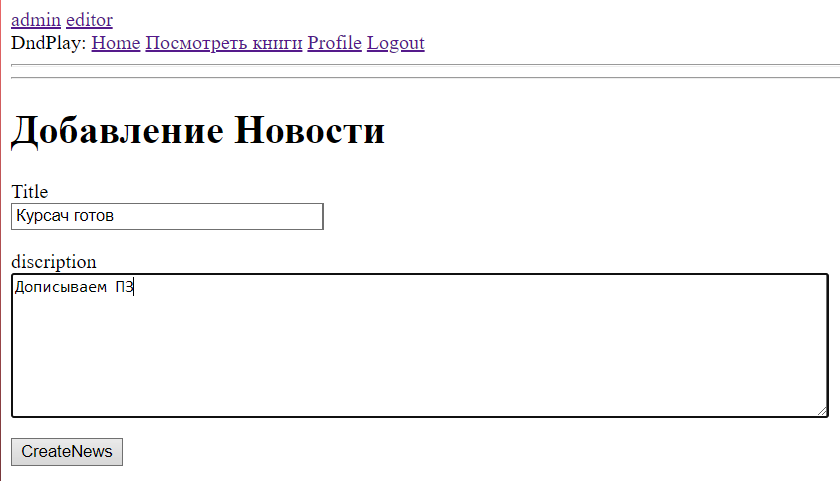


Рисунок 25 – Страница добавления новости

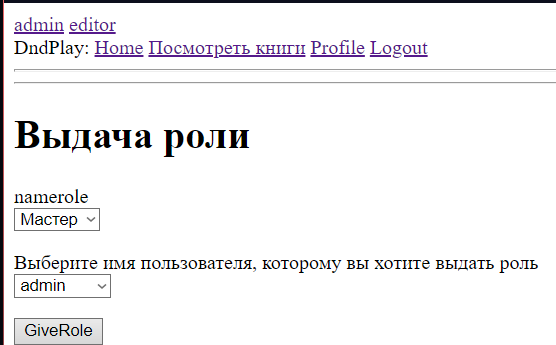


Рисунок 26 – Страница выдачи роли

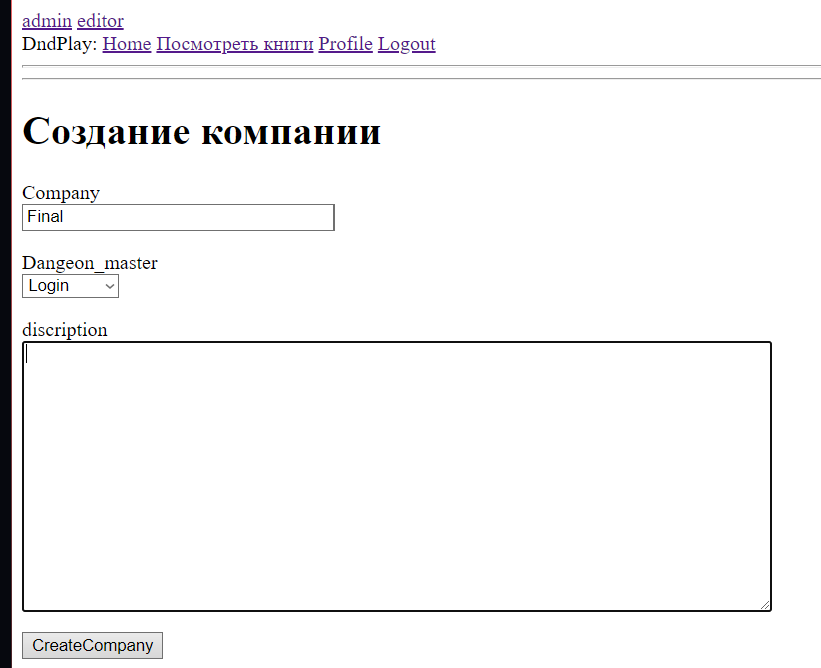


Рисунок 27 – Страница создание компании

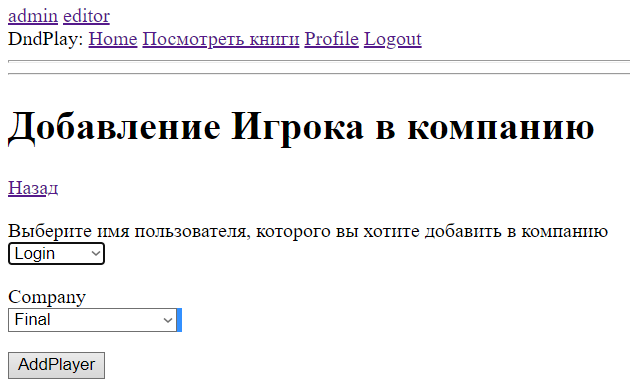


Рисунок 28 – Страница добавления игрока в компанию

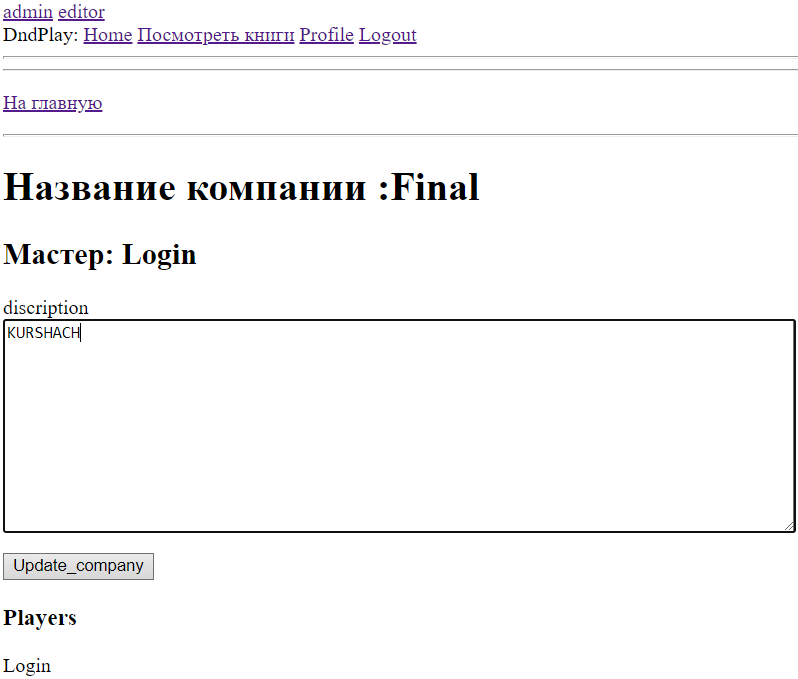


Рисунок 29 – Страница информации о компании и изменение описания