Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

‹‹Уральский федеральный университет имени первого Президента России

Б.Н.Ельцина››

Кафедра ‹‹Интеллектуальных информационных технологий››

дисциплина ‹‹WEB программирование››

Курсовая работа

«Сайт игры ArmyCommander»

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_ Свинцов Д. В.

(подпись преподавателя)

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_ Зуева Г. А.

(подпись студентов)

Группа: ФО-321001

Екатеринбург

2015

# Введение

В этом семестре я работала над игрой ArmyCommander, имитирующей события Второй Мировой Войны. Необходимо было создать сайт, содержащий информацию об этой игре и об игроках. Поэтому разработка этого сайта и стала темой моей курсовой работы по Web программированию.

# Описание проекта

Сайт состоит из нескольких страниц:

Главная - содержит основную информацию об игре;

Гайд - содержит инструкцию по игре для пользователя;

Рейтинг - отображает топ игроков на каждой карте;

Карты - содержит файлы карт;

Добавление данных - для добавления сущностей "игрок" или "карта" в базу данных.

Также есть страница с формой авторизации.

Добавлять данные в базу данных может только авторизованный пользователь.

# Обоснование проекта

Сайт направлен на ознакомление с основной информацией об игре ArmyCommander и игровым процессом. Также на сайте может содержаться информация о результатах игроков и карты для игрового окна.

# Инструкция пользователя

Сайт состоит из пяти страниц, переход на каждую из которых производится при нажатии на соответствующий заголовок. Начальная страница - домашняя, содержит основную информацию об игре, её и видит пользователь, когда заходит на сайт.

Помимо пяти страниц, есть форма для входа на сайт в качестве администратора.

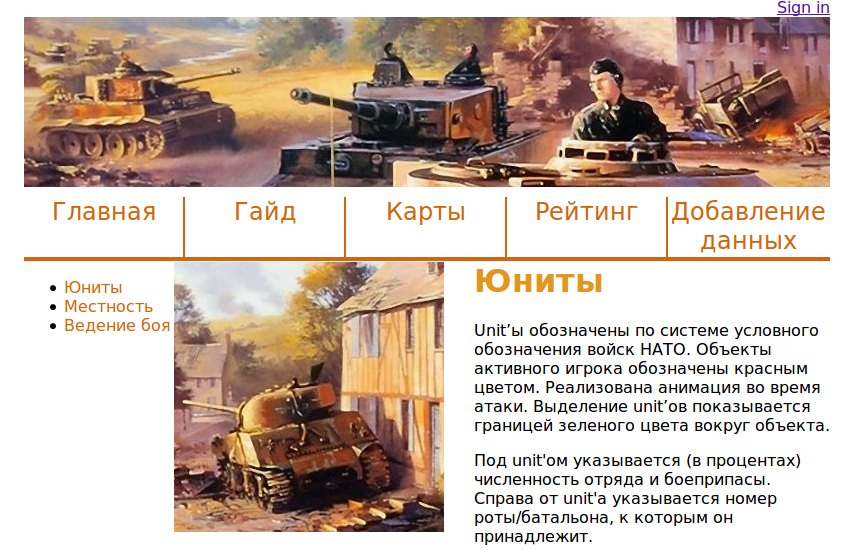


Рис 1. Вид сайта

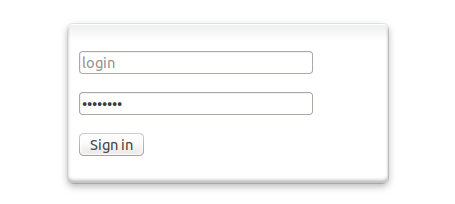


Рис 2. Форма авторизации

# Инструкция программиста

Установка

Для установки требуется версия Python 2.7, наличие установленных пакетов Jinja2 и PyramidFramework.

Открываем командную строку и переходим в корневую папку проекта. Затем прописываем команду:

python setup.py develop

Инициализация баз данных

Для инициализации баз данных прописываем в командной строке следующую команду:

initialize\_ArmyCommander\_db development.ini

Запуск проекта

Запуск проекта производится следующей командой:

pserve development.ini --reload

Авторизация  
Для того, чтобы зайти на сайт в качестве администратора, нужно перейти на страницу с формой авторизации, нажав на ссылку в правом верхнем углу. Также можно перейти на страницу "Добавление данных". Просматривать её могут только авторизованные пользователи, так что вместо страницы отобразится форма авторизации.

Логин и пароль указаны в файле security.py.  
Если пользователь авторизован, логин указывается в левом верхнем углу страниц сайта.

Авторизовавшись, можно добавлять данные в базу данных на соответствующей странице сайта.

Описание ключевых функций, классов, блоков кода

В файл \_init\_.py были добавлены роуты для страниц сайта, а затем настройки авторизации и аутентификации, а также настройки sacrud для работы с базой данных.

from pyramid.config import Configurator

from sqlalchemy import engine\_from\_config

from pyramid.authentication import AuthTktAuthenticationPolicy

from pyramid.authorization import ACLAuthorizationPolicy

from .security import groupfinder

from .models import (

DBSession,

Base,

Player,

Score,

Map,

TopRating,

)

def main(global\_config, \*\*settings):

""" This function returns a Pyramid WSGI application.

"""

engine = engine\_from\_config(settings, 'sqlalchemy.')

DBSession.configure(bind=engine)

Base.metadata.bind = engine

authn\_policy = AuthTktAuthenticationPolicy(

'sosecret', callback=groupfinder, hashalg='sha512')

authz\_policy = ACLAuthorizationPolicy()

config = Configurator(settings=settings, root\_factory='armycommander.models.AccessGroups')

config.set\_authentication\_policy(authn\_policy)

config.set\_authorization\_policy(authz\_policy)

config.include('pyramid\_chameleon')

config.include('pyramid\_sqlalchemy')

config.include('pyramid\_sacrud',route\_prefix='admin')

config.include('pyramid\_jinja2')

config.add\_static\_view('static', 'static', cache\_max\_age=3600)

config.add\_route('home', '/home')

config.add\_route('guide', '/guide')

config.add\_route('map', '/map')

config.add\_route('rating', '/rating')

config.add\_route('editdatabase', '/workwithdatabase')

config.add\_route('login', '/login')

config.add\_route('logout', '/logout')

settings['pyramid\_sacrud.models'] = (('Project',[Player,Score,Map,TopRating]))

config.scan()

return config.make\_wsgi\_app()

В файле models.py описаны таблицы базы данных: Player, Score, Map, TopRating.

from sqlalchemy import (

Column,

Index,

Integer,

ForeignKey,

Text,

)

from sqlalchemy.ext.declarative import declarative\_base

from sqlalchemy.orm import (

scoped\_session,

sessionmaker,

relationship,

)

from pyramid.security import (

Allow,

Everyone,

)

from zope.sqlalchemy import ZopeTransactionExtension

DBSession = scoped\_session(sessionmaker(extension=ZopeTransactionExtension()))

Base = declarative\_base()

class Player(Base):

\_\_tablename\_\_ = 'player'

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

name = Column(Text, primary\_key=True)

def \_\_repr\_\_(self):

return self.name

class Map(Base):

\_\_tablename\_\_ = 'map'

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

name = Column(Text, primary\_key = True)

ratings = relationship("TopRating", backref = 'map', lazy='dynamic')

def \_\_repr\_\_(self):

return self.name

class Score(Base):

\_\_tablename\_\_ = 'score'

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

value = Column(Integer, primary\_key = True)

user\_id = Column(Integer, ForeignKey('player.id'))

#user = relationship("Player", backref = 'score')

map\_id = Column(Integer, ForeignKey('map.id'))

#map\_ = relationship("Map", backref='score')

def \_\_repr\_\_(self):

return self.value

class TopRating(Base):

\_\_tablename\_\_ = 'top\_rating'

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

map\_id = Column(Integer, ForeignKey('map.id'))

first\_user\_name = Column(Text, ForeignKey('player.name'))

second\_user\_name = Column(Text, ForeignKey('player.name'))

third\_user\_name = Column(Text, ForeignKey('player.name'))

def \_\_repr\_\_(self):

return self.first\_user\_name + ' ' + self.second\_user\_name + ' ' + self.third\_user\_name

class AccessGroups(object):

\_\_name\_\_ = None

\_\_acl\_\_ = [

(Allow, 'group:editors', 'edit'),

(Allow, Everyone, 'view'),

]

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

pass

Этот файл также содержит права для групп пользователей.

В файле security.py указаны группы пользователей.

USERS = {'admin':'password',

'admino':'00000000'}

GROUPS = {'admino':['group:editors']}

def groupfinder(userid, request):

if userid in USERS:

return GROUPS.get(userid, [])

Файл views.py содержит представления страниц сайта и страницы с формой авторизации.

from pyramid.response import Response

from sqlalchemy.exc import DBAPIError

from .models import (

DBSession,

Player,

Score,

Map,

TopRating,

AccessGroups,

)

from pyramid.httpexceptions import (

HTTPFound,

HTTPNotFound,

)

from pyramid.view import (

view\_config,

forbidden\_view\_config,

)

from pyramid.security import (

remember,

forget,

)

from .security import USERS

@view\_config(route\_name='home', renderer='templates/home.jinja2')

def home(request):

if request.authenticated\_userid == "admino":

return {'username': request.authenticated\_userid, 'project': 'ArmyCommander'}

return {'project': 'ArmyCommander'}

@view\_config(route\_name='map', renderer='templates/map.jinja2')

def maps(request):

if request.authenticated\_userid == "admino":

return {'username': request.authenticated\_userid, 'project': 'ArmyCommander'}

return {'project': 'ArmyCommander'}

@view\_config(route\_name='guide', renderer='templates/guide.jinja2')

def guide(request):

return {'username': request.authenticated\_userid, 'project': 'ArmyCommander'}

@view\_config(route\_name='rating', renderer='templates/rating.jinja2')

def rating(request):

try:

maps = DBSession.query(Map).all()

top\_db = DBSession.query(TopRating).all()

except DBAPIError:

return Response(conn\_err\_msg, content\_type='text/plain', status\_int=500)

if request.authenticated\_userid == "admino":

return {'username': request.authenticated\_userid, 'maps': maps, 'TopRating': top\_db, 'project': 'ArmyCommander'}

return {'maps': maps, 'TopRating': top\_db, 'project': 'ArmyCommander'}

@view\_config(route\_name='editdatabase', context='.models.AccessGroups',

renderer='templates/workwithdatabase.jinja2', permission='edit')

def edit\_db(request):

try:

players = DBSession.query(Player).all()

maps = DBSession.query(Map).all()

except DBAPIError:

return Response(conn\_err\_msg, content\_type='text/plain', status\_int=500)

user = request.authenticated\_userid

from urlparse import parse\_qs

if request.POST:

if "player\_button" in request.params:

max\_id = len(players)

param = request.params["player\_name"]

DBSession.add(Player(id=max\_id,name=param))

players = DBSession.query(Player).all()

return {'allmaps': maps, 'allplayers': players, 'username': user, 'information': 'player was added', 'project': 'ArmyCommander'}

if "map\_button" in request.params:

max\_id = len(maps)

param = request.params["map\_name"]

DBSession.add(Map(id=max\_id,name=param))

maps = DBSession.query(Map).all()

return {'allmaps': maps, 'allplayers': players, 'username': user, 'information': 'map was added', 'project': 'ArmyCommander'}

return {'allmaps': maps, 'allplayers': players, 'username': user, 'project': 'ArmyCommander'}

conn\_err\_msg = """\

Pyramid is having a problem using your SQL database. The problem

might be caused by one of the following things:

1. You may need to run the "initialize\_ArmyCommander\_db" script

to initialize your database tables. Check your virtual

environment's "bin" directory for this script and try to run it.

2. Your database server may not be running. Check that the

database server referred to by the "sqlalchemy.url" setting in

your "development.ini" file is running.

After you fix the problem, please restart the Pyramid application to

try it again.

"""

@view\_config(route\_name='login', renderer='templates/loginform.jinja2')

@forbidden\_view\_config(renderer='templates/loginform.jinja2')

def login(request):

if 'submitted' in request.params:

login = request.params['login']

password = request.params['password']

if USERS.get(login) == password:

headers = remember(request, login)

return HTTPFound(location = 'home', headers = headers)

else:

return {'message': "Incorrect login or password", 'project': 'ArmyCommander'}

return {'project': 'ArmyCommander'}

@view\_config(route\_name='logout')

def logout(request):

headers = forget(request)

return HTTPFound(location = request.referrer,

headers = headers)

# Выводы

Во время разработки сайта в проекте были использованы следующие технологии:

* Фреймворк Pyramid
* Шаблонизатор Jinja2
* ОРМ SQLAlchemy
* SQLite

В процессе работы я научилась создавать сайты при помощи фреймворка Pyramid с использованием шаблонов, а также подключать к проекту базу данных SQLite и работать с ней. Кроме этого я научилась создавать группы пользователей и изучила способы авторизации пользователей.