

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Российский экономический университет им. Г.В. Плеханова»  
Московский приборостроительный техникум

## Курсовой проект

ПМ 11 Разработка, администрирование и защита баз данных

МДК 11.01 «Технология разработки и защиты баз данных»

Специальность 09.02.07 «Информационные системы и  
программирование»

МДК 11.01 ТРЗБД

Специальность 09.02.07 «Информационные системы и  
программирование»

Квалификация: Программист

Тема: «Разработка приложения для администрирование Linux машин»

## Пояснительная записка

Листов: 0

Руководитель

\_\_\_\_\_ / Г.Ю. Волкова

« \_\_\_\_ » \_\_\_\_\_ 2024 год

Исполнитель

\_\_\_\_\_ / И.В. Евский

« \_\_\_\_ » \_\_\_\_\_ 2024 год

2024

Введение

- 1.1. Тема: «Разработка приложения для администрирование Linux машин».
- 1.2. Состав курсового проекта:
  - 1.2.1. Задание КП
  - 1.2.2. Пояснительная записка
  - 1.2.3. Программа (исходные данные) на электронном носителе
  - 1.2.4. Презентация и инсталляционный пакет на git репозитории
- 1.3. Содержание пояснительной записки:
2. ВВЕДЕНИЕ
3. ОБЩАЯ ЧАСТЬ
  - 3.1. Цель разработки
  - 3.2. Средства разработки
4. СПЕЦИАЛЬНАЯ ЧАСТЬ
  - 4.1. Постановка задачи
    - 4.1.1. Входные данные предметной области
    - 4.1.2. Выходные данные предметной области
    - 4.1.3. Требования к проекту
  - 4.2. Внешняя спецификация
    - 4.2.1. Описание задачи
    - 4.2.2. Входные и выходные данные
    - 4.2.3. Методы
    - 4.2.4. Тесты
    - 4.2.5. Контроль целостности данных
  - 4.3. Проектирование
    - 4.3.1. Схема архитектуры приложения
    - 4.3.2. Логическая схема данных
    - 4.3.3. Физическая схема данных
    - 4.3.4. Структурная схема
    - 4.3.5. Функциональная схема
    - 4.3.6. Диаграмма классов
    - 4.3.7. Схема тестирования
    - 4.3.8. Схема пользовательского интерфейса
  - 4.4. Результат работы программы
5. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ
  - 5.1. Инструментальные средства
  - 5.2. Отладка программы
  - 5.3. Защитное программирование
  - 5.4. Характеристики программы
6. ЗАКЛЮЧЕНИЕ
7. СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ
8. ПРИЛОЖЕНИЕ А. Текст программы
9. ПРИЛОЖЕНИЕ Б. Сценарий и результаты тестовых испытаний
10. ПРИЛОЖЕНИЕ В. Руководство пользователя
11. ПРИЛОЖЕНИЕ Г. Скрипт базы данных

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение высшего образования  
«Российский экономический университет имени Г.В. Плеханова»  
**МОСКОВСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ТЕХНИКУМ**

«УТВЕРЖДАЮ»

Заместитель директора по учебной работе

\_\_\_\_\_ Д.А. Клопов

« \_\_\_\_ » \_\_\_\_\_ 2024 г.

## **ЗАДАНИЕ**

на выполнение курсового проекта (курсовой работы)

**Евскину Ивану Валерьевичу**

(фамилия, имя, отчество студента — полностью)

студенту группы П50-8-21 специальности 09.02.07 «Информационные системы и программирование» по МДК 11.01 «Технология разработки и защиты баз данных»

1. Презентация и инсталляционный пакет как tar.gz архив
2. Содержание пояснительной записки:

### **ВВЕДЕНИЕ**

#### **1. ОБЩАЯ ЧАСТЬ**

- 1.1. Цель разработки
- 1.2. Средства разработки

#### **2. СПЕЦИАЛЬНАЯ ЧАСТЬ**

##### **2.1. Постановка задачи**

- 2.1.1. Входные данные предметной области
- 2.1.2. Выходные данные предметной области
- 2.1.3. Требования к проекту

##### **2.2. Внешняя спецификация**

- 2.2.1. Описание задачи
- 2.2.2. Входные и выходные данные
- 2.2.3. Методы
- 2.2.4. Тесты
- 2.2.5. Контроль целостности данных

##### **2.3. Проектирование**

- 2.3.1. Схема архитектуры приложения
- 2.3.2. Логическая схема данных
- 2.3.3. Физическая схема данных
- 2.3.4. Структурная схема
- 2.3.5. Функциональная схема
- 2.3.6. Диаграмма классов
- 2.3.7. Схема тестирования
- 2.3.8. Схема пользовательского интерфейса

##### **2.4. Результат работы программы**

#### **3. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ**

- 3.1. Инструментальные средства
- 3.2. Отладка программы
- 3.3. Защитное программирование
- 3.4. Характеристики программы

### **СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ**

ПРИЛОЖЕНИЕ А. Текст программы

ПРИЛОЖЕНИЕ Б. Сценарий и результаты тестовых испытаний

ПРИЛОЖЕНИЕ В. Руководство пользователя

ПРИЛОЖЕНИЕ Г. Скрипт базы данных

## 3. Содержание задания по проекту (работе) — перечень вопросов, подлежащих разработке

	Разрабатываемый вопрос	Объем от всего задания, %	Срок выполнения
<b>A</b>	<b>Описательная часть проекта (введение, общее описание и т. д.)</b>	<b>5</b>	
1.	Введение	-	
2.	Цель разработки	-	
3.	Средства разработки	-	
<b>Б</b>	<b>Анализ задачи и её постановка</b>	<b>15</b>	
1.	Определение требований к программе	-	
2.	Спецификация программы (описание задачи, описание входных и выходных данных, методы)	-	
3.	Тесты, контроль целостности данных	-	
<b>В</b>	<b>Проектирование и реализация</b>	<b>55</b>	
1.	Схемы проекта (схема архитектуры, логическая схема данных, физическая схема данных, функциональная и структурная схемы, диаграмма классов, схема тестирования, схема пользовательского интерфейса)	-	
2.	Реализация в инструментальной среде	-	
<b>Г</b>	<b>Технологическая часть проекта</b>	<b>5</b>	
1.	Инструментальные средства разработки	-	
2.	Отладка программа	-	
3.	Защитное программирование	-	
4.	Характеристика программы	-	
<b>Д</b>	<b>Программная документация</b>	<b>10</b>	
1.	Приложение А. Текст программы	-	
2.	Приложение Б. Сценарий и результаты тестовых испытаний	-	
3.	Приложение В. Руководство пользователя	-	
4.	Приложение Г. Скрипт базы данных	-	
<b>Е</b>	<b>Экспериментальная часть проекта</b>	<b>10</b>	
1.	Программа на машинном носителе. Информация на носителе разбита на разделы: эксплуатационный пакет, тексты программы, документация.	-	

Руководитель курсового проекта (работы) Волкова Галина Юрьевна, преподаватель  
«06» сентября 2024 года \_\_\_\_\_ / Г.Ю. Волкова /

Дата выдачи курсового задания «06» сентября 2024 года  
Срок сдачи законченного проекта (работы) «1» ноября 2024 года

Задание принял к исполнению  
«06» сентября 2024 года \_\_\_\_\_ / Е.И. Валерьевич /

## **Презентация и инсталляционный пакет**

Программа предоставляется как tar.gz архив

## **ВВЕДЕНИЕ**

Наименование программы: ИС для назначения задач внутри организации.

Краткая характеристика области применения программы. Программа «ИС для назначения задач внутри организации» для назначения задач сотрудникам внутри организации.

## **1.ОБЩАЯ ЧАСТЬ**

### **1.1 Цель разработки**

Цель: Создание системы которая хранить данные о командах введённых пользователями в процессе администрирования систем на базе ядра linux.

### **1.2 Средства разработки**

Средства разработки: vim-редактор текста dbeather-программа для работы с СУБД  
firefox-браузер python3-интерпритатор

## **2. СПЕЦИАЛЬНАЯ ЧАСТЬ**

### **2.1 Постановка задачи**

#### **2.1.1 Входные данные предметной области**

Логин и пароль пользователя в системе, Порт ip пароль пользователь удалённой машины.

#### **2.1.2 Выходные данные предметной области**

Файл с логом действий пользователя.

#### **2.1.3 Требования к проекту**

Проект должен работать везде где соблюдаются системные требования.

### **2.2 Внешняя спецификация**

#### **2.2.1 Описание задачи**

Создать web приложение использующие стороннее приложение для администрирования систем на базе ядра linux

#### **2.2.2 Входные и выходные данные**

Логин и пароль пользователя в системе, Порт ip пароль пользователь удалённой машины.

#### **2.2.3 Методы**

CRUD Машины и методы подключения по ssh

#### **2.2.4 Тесты**

Ручное тестирование.

#### **2.2.5 Контроль целостности данных**

Резервное копирование - бекап бд для восстановления

Контроль доступа - доступ к бд и приложению по логину и паролю

Шифрование - шифрование данных по http и ssh протоколам

Хеширование - хеширование пароля пользователя



## 2.3 Проектирование

### 2.3.1 Схема архитектуры приложения

Схема архитектуры Рисунок 1

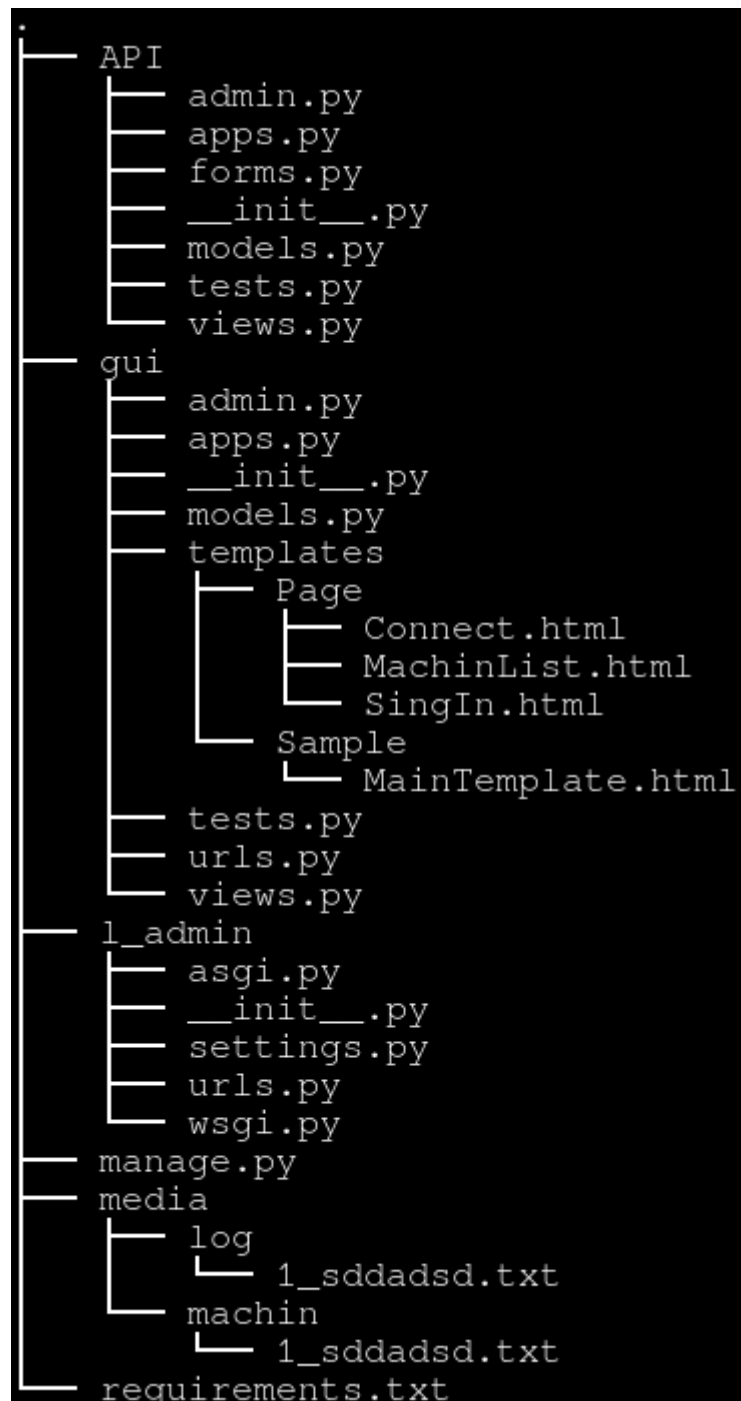


Рисунок 1-Структурная схема

### 2.3.2 Логическая схема данных

#### Логическая схема данных Рисунок 2

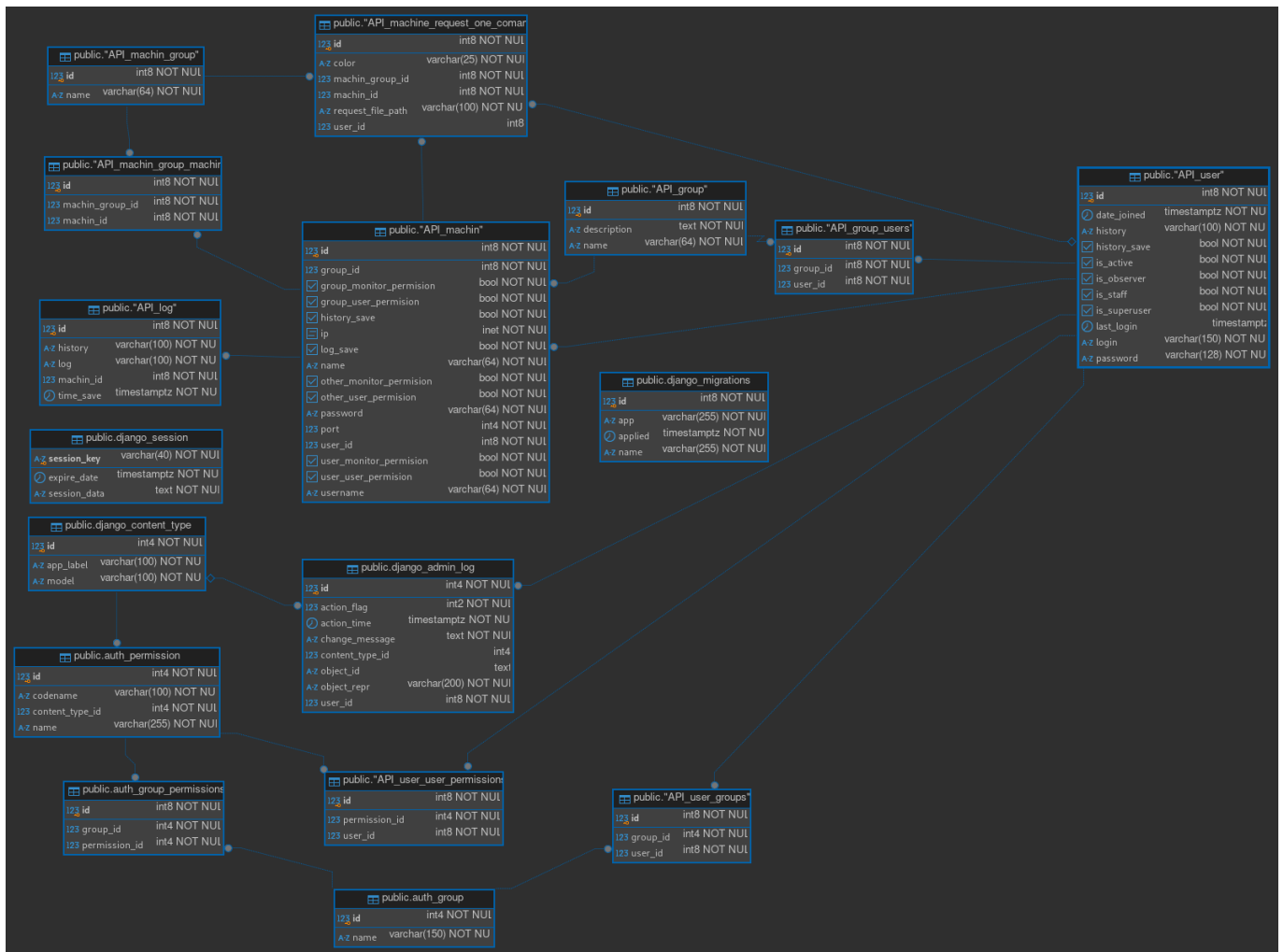


Рисунок 2-Логическая схема данных

### 2.3.3 Физическая схема данных

Физическая схема данных Рисунок 3

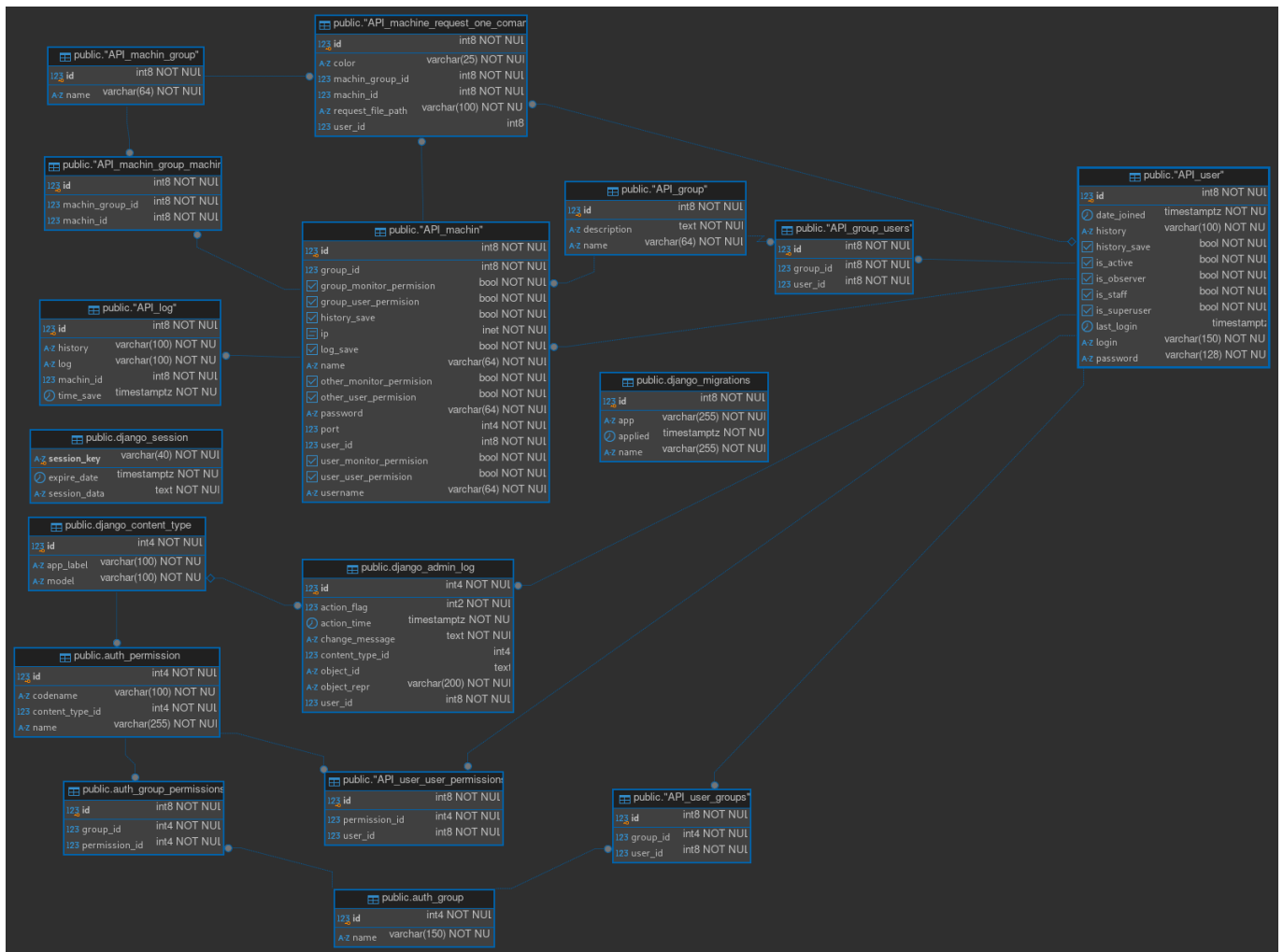


Рисунок 3-Физическая схема данных

### 2.3.4 Диаграмма классов

Диаграмма классов Рисунок 4

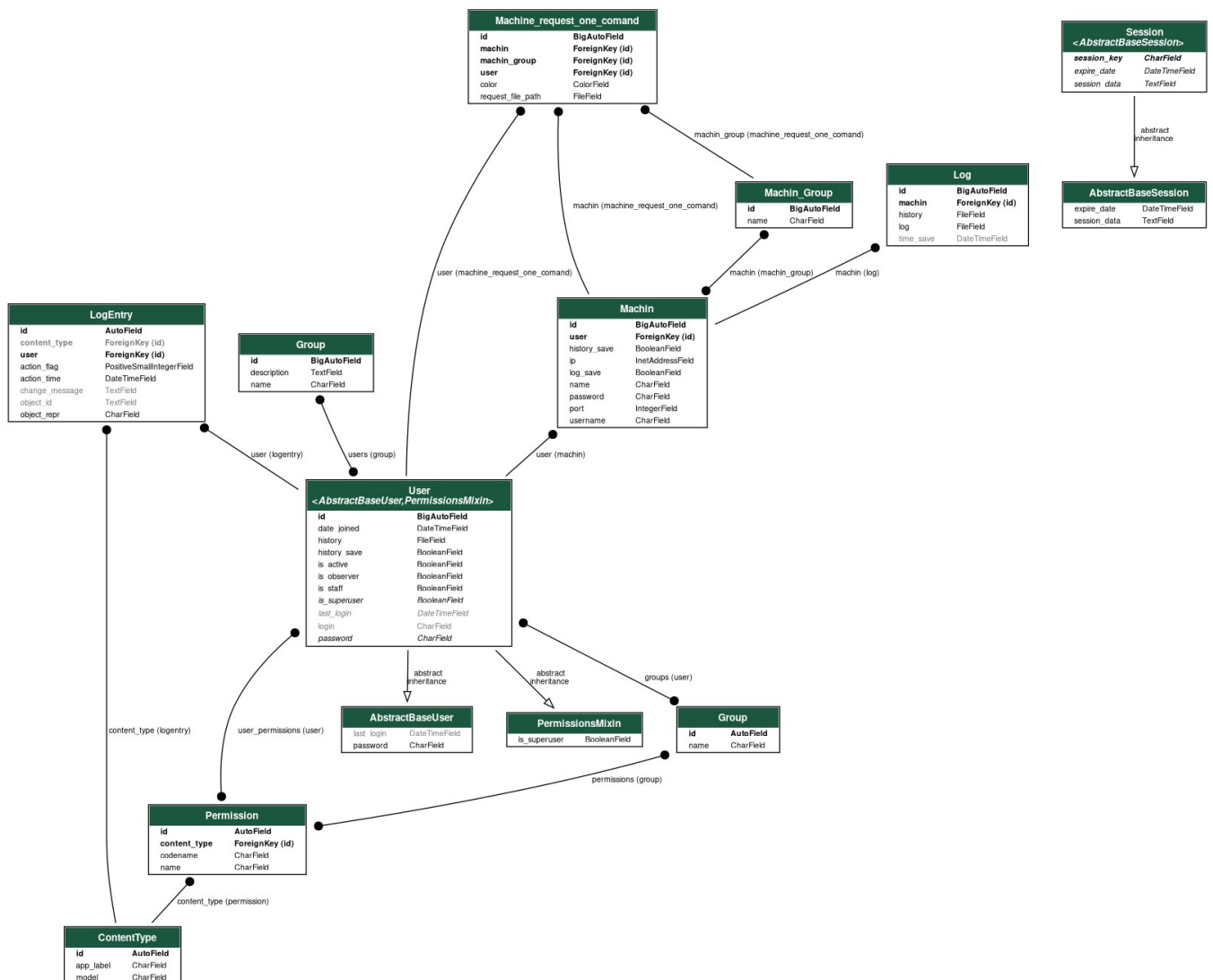


Рисунок 4-Диаграмма классов

### 2.3.5 Схема тестирования

Схема тестирования Рисунок 5

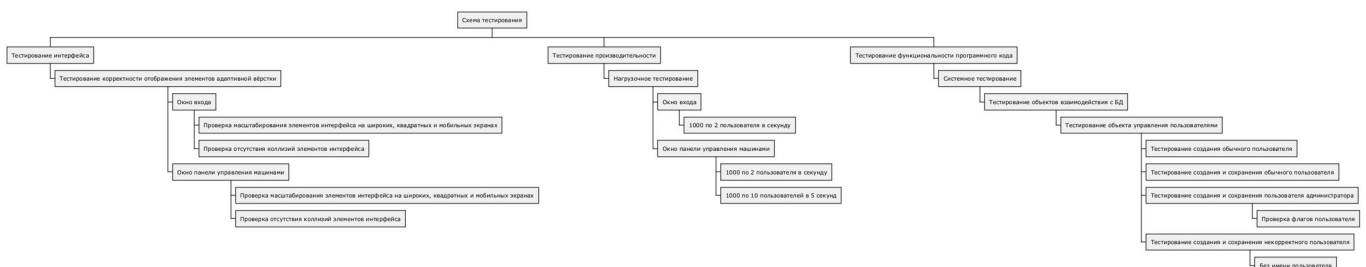


Рисунок 5-Схема тестирования

### 2.4 Результат работы программы

Программа запускается, пишет лог, подключается к бд и предоставляет web интерфейс на указанном при запуске порту для указанных ip адресов.

### **3.ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ**

#### **3.1 Инструментальные средства**

Инструментальные средства: vim-редактор текста python3-интерпритатор

#### **3.2 Отладка программы**

Отладочные средства:dbather-программа для работы с СУБД firefox-браузер

#### **3.3 Защитное программирование**

Пароли пользователя хешируются.

#### **3.4 Характеристики программы**

Размер программы не более 900 Мб. Программа работает на устройствах запускающих интерпритатор питона и библиотеки описанные в requerement.txt.

Так-же программа не работает при отсутствии бд и отсутствии подключения к сети.

## **СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ**

**<https://docs.djangoproject.com/en/5.1/>**

**<https://www.postgresql.org/docs/>**

**<https://dbeaver.com/docs/dbeaver/>**

**<https://docs.python.org/3/>**

## **Расшифровка сокращений**

CRUD — создать, прочитать, обновить, удалить

ip — адрес в сети

бд — база данных

url-ссылка

## ПРИЛОЖЕНИЕ А. ТЕКСТ ПРОГРАММЫ

### АННОТАЦИЯ

#### 1. Настройки

Здесь описаны настройки проекта. В данном пункте предоставлен исходный код.

#### 2. Зависимости

Здесь описаны Зависимости проекта. В данном пункте предоставлен исходный код.

#### 3. Url

Здесь описаны URL проекта. В данном пункте предоставлен исходный код.

#### 4. Приложения

Приложения проекта

4.1 Приложение API. В данном пункте предоставлен исходный код.

Приложение API

4.1.3 Тесты. В данном пункте предоставлен исходный код.

Здесь описаны тесты

4.2 Приложение gui. В данном пункте предоставлен исходный код.

Приложение gui

##### 4.2.1 Url

Здесь описаны URL проекта. В данном пункте предоставлен исходный код.

##### 4.2.2 Представления

Здесь описаны Представления проекта. В данном пункте предоставлен исходный код.

##### 4.2.3 static

Здесь static файлы проекта. В данном пункте предоставлен исходный код.

##### 4.2.4 templates

Здесь шаблоны файлы проекта. В данном пункте предоставлен исходный код.

##### 4.2.4.1 MainTemplate.html



Здесь основной шаблон проекта. В данном пункте предоставлен исходный код.

#### 4.2.5 Страницы

В данном разделе описаны страницы.

##### 4.2.5.1 Connect.html

Здесь страница Connect. В данном пункте предоставлен исходный код

##### 4.2.5.2 MachinList.html

Здесь страница MachinList. В данном пункте предоставлен исходный код

##### 4.2.5.3 SingIn.html

Здесь страница SingIn. В данном пункте предоставлен исходный код

##### 4.2.5.4 Status.html

Здесь шаблон статусов Status. В данном пункте предоставлен исходный код

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	4
1. Настройки.....	5
2. Зависимости.....	11
3. Url.....	12
4. Приложения.....	14
4.1 Приложение API.....	14
4.1.3 Тесты.....	20
4.2 Приложение gui.....	23
4.2.1 Url.....	23
4.2.2 Представления.....	24
4.2.3 Static.....	29
4.2.4 Templates.....	31
4.2.4.1 MainTemplate.html.....	31
4.2.5 Страницы.....	32
4.2.5.1 Connect.html.....	32
4.2.5.2 MachinList.html.....	33
4.2.5.3 SingIn.html.....	37
4.2.5.4 Status.html.....	38

## **ВВЕДЕНИЕ**

В данном разделе документации описан текст программы l\_admin

## 1.Настройки

l\_admin/l\_admin/settings.py

"""

Django settings for l\_admin project.

Generated by 'django-admin startproject' using Django 5.1.2.

For more information on this file, see

<https://docs.djangoproject.com/en/5.1/topics/settings/>

For the full list of settings and their values, see

<https://docs.djangoproject.com/en/5.1/ref/settings/>

"""

import os

from pathlib import Path

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

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

# Quick-start development settings - unsuitable for production

# See <https://docs.djangoproject.com/en/5.1/howto/deployment/checklist/>

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

SECRET\_KEY = 'django-insecure-#3()dy!ni\_q&h!y531o)(%!k1%\$&a12m-t\$lg88ki697\*ewnyu'

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

```
DEBUG = True
```

```
ALLOWED_HOSTS = ['*']
```

```
# Application definition
```

```
INSTALLED_APPS = [
```

```
    'django.contrib.admin',
```

```
    'django.contrib.auth',
```

```
    'django.contrib.contenttypes',
```

```
    'django.contrib.sessions',
```

```
    'django.contrib.messages',
```

```
    'django.contrib.staticfiles',
```

```
    'django_postgres_backup',
```

```
    'django_extensions',
```

```
    'colorfield',
```

```
    'netfields',
```

```
    'API',
```

```
    'gui'
```

```
]
```

```
AUTH_USER_MODEL = 'API.User'
```

```
MIDDLEWARE = [
```

```
    'django.middleware.security.SecurityMiddleware',
```

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

```
    'django.middleware.common.CommonMiddleware',
```

```
    'django.middleware.csrf.CsrfViewMiddleware',
```

```
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'l_admin.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

```
WSGI_APPLICATION = 'l_admin.wsgi.application'
```

```
# Database
```

# <https://docs.djangoproject.com/en/5.1/ref/settings/#databases>

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'mydb',
        'USER': 'myuser',
        'PASSWORD': '123',
        'HOST': 'localhost',
        'PORT': '',
    }
}
```

# Password validation

# <https://docs.djangoproject.com/en/5.1/ref/settings/#auth-password-validators>

```
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
```

```

    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

```

# Internationalization

# <https://docs.djangoproject.com/en/5.1/topics/i18n/>

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'Europe/Moscow'

USE\_I18N = True

USE\_TZ = True

MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')

MEDIA\_URL = '/media/'

# Static files (CSS, JavaScript, Images)

# <https://docs.djangoproject.com/en/5.1/howto/static-files/>

STATIC\_ROOT = BASE\_DIR / 'static'

STATIC\_URL = 'static/'



# Default primary key field type

# <https://docs.djangoproject.com/en/5.1/ref/settings/#default-auto-field>

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

## **2.Зависимости**

l\_admin/requirements.txt

asgiref==3.8.1

bcrypt==4.2.1

certifi==2024.8.30

cffi==1.17.1

charset-normalizer==3.4.0

cryptography==43.0.3

Django==4.2.16

django-appconf==1.0.6

django-colorfield==0.11.0

django-compressor==4.5.1

django-extensions==3.2.3

django-libsass==0.9

django-netfields==1.3.2

django-postgres-backup==1.0.0

django-rest-framework==3.15.2

idna==3.10

iniconfig==2.0.0

libsass==0.23.0

netaddr==1.3.0

packaging==24.2

paramiko==3.5.0

pillow==11.0.0

pluggy==1.5.0

psycpg2==2.9.10

psycopg2-binary==2.9.10

pycparser==2.22

PyNaCl==1.5.0

pytest==7.4.4

rcssmin==1.1.2

requests==2.32.3

rjsmin==1.2.2

six==1.16.0

sqlparse==0.5.1

urllib3==2.2.3

### 3.Url

l\_admin/l\_admin/urls.py

"""

URL configuration for l\_admin project.

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

<https://docs.djangoproject.com/en/5.1/topics/http/urls/>

Examples:

Function views

1. Add an import: from my\_app import views
2. Add a URL to urlpatterns: path("", views.home, name='home')

Class-based views

1. Add an import: from other\_app.views import Home
2. Add a URL to urlpatterns: path("", Home.as\_view(), name='home')

Including another URLconf

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

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

```
"""
```

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
from gui.urls import urlgui

urlpatterns = [
    path("", include(urlgui)),
    path('admin/', admin.site.urls),
] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)+
static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

## 4.Приложения

### 4.1 Приложение API

#### 4.1.1 Модели

```
l_admin/API/models.py

from django.db import models

from django.contrib.auth.models import AbstractBaseUser, BaseUserManager,
PermissionsMixin

from django.db import models

from django.core.validators import MaxValueValidator, MinValueValidator

from django.utils import timezone

from django.utils.translation import gettext_lazy as _

from colorfield.fields import ColorField

from datetime import timedelta, datetime

from netfields import InetAddressField, NetManager

from django import forms

class UserManager(BaseUserManager):

    def create_user(self, login, history_save=True, password=None,commit=True):

        if not login: raise ValueError(_('Users must have a login'))

        user = self.model(login=login,history_save=history_save)

        user.set_password(password)

        if commit: user.save(using=self._db)

        return user

    def create_superuser(self, login, password):

        user = self.create_user(password=password,login=login,commit=False)

        user.is_staff = True

        user.is_superuser = True
```

```
user.save(using=self._db)
```

```
return user
```

```
class User(AbstractBaseUser, PermissionsMixin):
```

```
    login = models.CharField(_('login'), max_length=150, blank=True, unique=True)
```

```
    is_active = models.BooleanField(_('active'), default=True, help_text=_('Designates  
whether this user should be treated as active. Unselect this instead of deleting accounts.'),)
```

```
    is_staff = models.BooleanField(_('staff status'), default=False, help_text=_('Designates  
whether the user can log into this admin site.'),)
```

```
    is_observer = models.BooleanField(default=False)
```

```
    history_save = models.BooleanField(default=True)
```

```
    history = models.FileField()
```

```
    date_joined = models.DateTimeField(_('date joined'), default=timezone.now)
```

```
    objects = UserManager()
```

```
    USERNAME_FIELD = 'login'
```

```
    def get_full_name(self):
```

```
        return self.login
```

```
    def __str__(self):
```

```
        return self.login
```

```
class Group(models.Model):
```

```
    users = models.ManyToManyField(User)
```

```
    name = models.CharField(max_length=64, unique=True)
```

```
    description = models.TextField(max_length=512)
```

```
class Machin(models.Model):
```

```
    name = models.CharField(max_length=64)
```

```
    username = models.CharField(max_length=64)
```

```
    password = models.CharField(max_length=64)
```

```
    user=models.ForeignKey(User, on_delete=models.PROTECT)
```

```

group=models.ForeignKey(Group, on_delete=models.PROTECT)
history_save = models.BooleanField(default=True)
log_save = models.BooleanField(default=True)
ip = InetAddressField()

port=models.IntegerField(validators=[MaxValueValidator(65535),MinValueValidator(1)])
objects = NetManager()
user_monitor_permission=models.BooleanField(default=True)
user_user_permission=models.BooleanField(default=True)
group_monitor_permission=models.BooleanField(default=True)
group_user_permission=models.BooleanField(default=True)
other_monitor_permission=models.BooleanField(default=True)
other_user_permission=models.BooleanField(default=True)
def form(self):
    return MachinForm(instance=self)
def getip(self):
    ip=str(self.ip.ip)
    return ip
class MachinForm(forms.ModelForm):
    password = forms.CharField(widget=forms.PasswordInput(), required = False)
    group=None
    try:
        group = forms.Select(choices=Group.objects.all().values_list('id', 'name'))
    except:
        group=None
    class Meta:
        model = Machin

```

```
fields = ('__all__')
```

```
class Log(models.Model):
```

```
    machin=models.ForeignKey(Machin, on_delete=models.CASCADE)
```

```
    time_save=models.DateTimeField(auto_now_add=True)
```

```
    log=models.FileField(editable=False,auto_created = True)
```

```
    history=models.FileField(editable=False,auto_created = True)
```

```
class Machin_Group(models.Model):
```

```
    machin = models.ManyToManyField(Machin,related_name='machin_group')
```

```
    name = models.CharField(max_length=64, unique=True)
```

```
class Machine_request_one_comand(models.Model):
```

```
    machin_group=models.ForeignKey(Machin_Group, on_delete=models.CASCADE)
```

```
    machin=models.ForeignKey(Machin, on_delete=models.CASCADE)
```

```
    user=models.ForeignKey(User, on_delete=models.SET_NULL, null=True)
```

```
    color = ColorField()
```

```
    request_file_path=models.FileField(editable=False)
```



### 4.1.2 Админ панель

```
l_admin/API/admin.py

from django import forms
from django.contrib import admin
from django.contrib.auth.admin import UserAdmin as BaseUserAdmin
from django.contrib.auth.forms import ReadOnlyPasswordHashField
from django.contrib.auth.models import Group
admin.site.unregister(Group)
from .models import *

class AddUserForm(forms.ModelForm):
    password1 = forms.CharField(label='Password', widget=forms.PasswordInput)
    password2 = forms.CharField(label='Confirm password', widget=forms.PasswordInput)
    class Meta:
        model = User
        fields = ('login',)
    def clean_password2(self):
        # Check that the two password entries match
        password1 = self.cleaned_data.get("password1")
        password2 = self.cleaned_data.get("password2")
        if password1 and password2 and password1 != password2: raise
forms.ValidationError("Passwords do not match")
        return password2
    def save(self, commit=True):
        # Save the provided password in hashed format
        user = super().save(commit=False)
        user.set_password(self.cleaned_data["password1"])
        if commit: user.save()
```

```

    return user

class UpdateUserForm(forms.ModelForm):
    password = ReadOnlyPasswordHashField()

    class Meta:
        model = User
        fields = ('password',
'login', 'is_active', 'is_staff', 'is_superuser', 'is_observer', 'history_save', 'history', 'date_joined',)

    def clean_password(self):
        return self.initial["password"]

class UserAdmin(BaseUserAdmin):
    form = UpdateUserForm
    add_form = AddUserForm
    list_display = ('login', 'is_staff')
    list_filter = ('is_staff',)
    fieldsets = ((None, {'fields': ('login', 'password')}), ('Permissions', {'fields': ('is_active',
'is_staff', 'is_superuser', 'is_observer', 'history_save', 'history', 'date_joined')})),)
    add_fieldsets = ((None, {'classes': ('wide',), 'fields': ('login', 'password1', 'password2')})),)
    search_fields = ('login',)
    ordering = ('login',)
    filter_horizontal = ()

admin.site.register(User, UserAdmin)
admin.site.register(Group)
admin.site.register(Log)

class MachinAdmin(admin.ModelAdmin):
    form = MachinForm

admin.site.register(Machin, MachinAdmin )
admin.site.register(Machin_Group)

```

```
admin.site.register(Machine_request_one_comand)
```

### 4.1.3 Тесты

```
l_admin/API/tests.py
from django.test import TestCase

from django.core.exceptions import ObjectDoesNotExist

from django.contrib.auth.hashers import make_password

from .models import User

class TC_UserManager(TestCase):
    def setUp(self):
        self.userManager = User.objects

    def test_CreateUser_regular_login_myuser_password_123(self):
        expected_login = "myuser"
        expected_password = "123"
        expected_is_staff = False
        expected_is_superuser = False

        self.userManager.create_user(login=expected_login, password=expected_password)
        user = self.userManager.get(login=expected_login)
        salt = user.password.split('$')[2]
        actual_login = user.login
        actual_password = user.password
```

```

actual_is_staff = user.is_staff
actual_is_superuser = user.is_superuser

self.assertEqual(expected_login, actual_login)
self.assertEqual(make_password(expected_password, salt=salt), actual_password)
self.assertEqual(expected_is_staff, actual_is_staff)
self.assertEqual(expected_is_superuser, actual_is_superuser)

def test_CreateUser_must_have_a_login(self):
    expected_password = "123"
    expected_exception_message = "Users must have a login"

    with self.assertRaises(ValueError) as catch:
        self.userManager.create_user(login=None, password=expected_password)

    actual_exception_message = str(catch.exception)
    self.assertEqual(expected_exception_message, actual_exception_message)

def test_CreateUser_when_commit_is_false_is_not_committed(self):
    expected_exception_message = "User matching query does not exist."

    login = "myuser"
    password = "123"
    self.userManager.create_user(login=login, password=password, commit=False)
    with self.assertRaises(ObjectDoesNotExist) as catch:
        self.userManager.get(login=login)
    actual_exception_message = str(catch.exception)

```

```
self.assertEqual(expected_exception_message, actual_exception_message)
```

```
def test_CreateSuperuser_superuser_login_mysuperuser_password_123(self):
```

```
    expected_login = "mysuperuser"
```

```
    expected_password = "123"
```

```
    expected_is_staff = True
```

```
    expected_is_superuser = True
```

```
    self.userManager.create_superuser(login=expected_login,
password=expected_password)
```

```
    superuser = self.userManager.get(login=expected_login)
```

```
    salt = superuser.password.split('$')[2]
```

```
    actual_login = superuser.login
```

```
    actual_password = superuser.password
```

```
    actual_is_staff = superuser.is_staff
```

```
    actual_is_superuser = superuser.is_superuser
```

```
    self.assertEqual(expected_login, actual_login)
```

```
    self.assertEqual(make_password(expected_password, salt=salt), actual_password)
```

```
    self.assertEqual(expected_is_staff, actual_is_staff)
```

```
    self.assertEqual(expected_is_superuser, actual_is_superuser)
```

```
def test_CreateSuperuser_must_have_a_login(self):
```

```
    expected_password = "123"
```

```
    expected_exception_message = "Users must have a login"
```

with self.assertRaises(ValueError) as catch:

self.userManager.create\_superuser(login=None, password=expected\_password)

actual\_exception\_message = str(catch.exception)

self.assertEqual(expected\_exception\_message, actual\_exception\_message)

class TC\_User(TestCase):

pass

## 4.2 Приложение gui

### 4.2.1 Url

l\_admin/gui/urls.py

from django.urls import path

from . import views

urlgui=[

path("", views.SingIn, name='url-singin'),

path('logout/', views.Logout, name='url-logout'),

path('list', views.List, name='url-list'),

path('list/<int:id>', views.List, name='url-list'),

path('connect/ssh/<int:id>', views.connect\_ssh, name='url-connect-ssh'),

]

### 4.2.2 Представления

```
l_admin/gui/views.py
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login, logout
from django.http import HttpResponseRedirect, Http404, HttpResponse
from API import models
import paramiko
import time
from datetime import datetime
# Create your views here.
def SingIn(request):
    if request.method == 'GET':
        context = ""
        return render(request, 'Page/SingIn.html', {'context': context})

    elif request.method == 'POST':
        username = request.POST.get('username', "")
        password = request.POST.get('password', "")

        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return HttpResponseRedirect('/list')
        else:
            context = {'error': 'Wrong credintials'} # to display error?
            return render(request, 'Page/SingIn.html', {'context': context})
```

```
def List(request,id=None):
```

```
    machins=models.Machin.objects.filter(user=request.user,user_monitor_permission=True)|
    models.Machin.objects.filter(other_monitor_permission=True)|
    models.Machin.objects.filter(group__in=
    models.Group.objects.filter(users=request.user),group_monitor_permission=True)
```

```
    machins=machins.order_by('id')
```

```
    if not request.user.is_authenticated: raise Http404
```

```
    if request.method == 'GET':
```

```
        return List_render(request,machins)
```

```
    elif request.method == 'POST':
```

```
        if 'id' in request.POST and id==0:
```

```
models.Machin.objects.get(id=request.POST['id']).delete()
```

```
        elif 'group_filter' in request.POST:
```

```
machins=models.Machin.objects.all().filter(machin_group__id=request.POST['group_filter'])
```

```
    return List_render(request,machins)
```

```
elif id!=None and id!=0:
```

```
    machin=models.Machin.objects.get(id=request.POST['id'])
```

```
    machin.name=request.POST['name']
```

```
    machin.group=models.Group.objects.get(id=request.POST['group'])
```

```
    machin.ip=request.POST['ip']
```

```
    machin.port=request.POST['port']
```

```
    machin.username=request.POST['username']
```

```
    if request.POST['password'] != '': machin.password=request.POST['password']
```

```
    machin.save()
```

```
elif id==0:
```

```
    machin=models.Machin()
```



```

    machin.name=request.POST['name']
    machin.group=models.Group.objects.get(id=request.POST['group'])
    machin.ip=request.POST['ip']
    machin.port=request.POST['port']
    machin.username=request.POST['username']
    machin.password=request.POST['password']
    machin.user=request.user
    machin.history_save=bool(request.POST['history_save'])
    machin.history_save=bool(request.POST['history_save'])
    machin.save()

    return List_render(request,machins)
def List_render(request,machins):
    return render(request, 'Page/MachinList.html', {
        "machins": machins,
        "form":models.MachinForm(),
        "machin_groups":models.Machin_Group.objects.all()
    })
def connect_ssh(request,id):
    if not request.user.is_authenticated: raise Http404
    cmdout=""
    usr=request.user
    if request.method == 'POST':
        try:
            cmdout=ssh(id,usr,request.POST['cmdin'])
            print(cmdout)
        except:
            cmdout="ERROR connect"

```

```

elif request.method == 'GET':
    try:
        cmdout=ssh(id,usr,"echo \"l_admin connect\"")
    except:
        cmdout="ERROR connect"
    return render(request, 'Page/Connect.html',{'cmdout':cmdout})
#no render
def ssh(id,usr,cmd):
    machin=models.Machin.objects.get(id=id)
    ssh_ = paramiko.SSHClient()
    ssh_.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh_.connect(str(machin.ip.ip), port=machin.port,
username=machin.username,password=machin.password, timeout=3)
    (stdin, stdout, stderr) = ssh_.exec_command(cmd)
    cmdout=stdout.read().decode("utf-8")
    ssh_.close()
    #log
    fph=f'media/machin/{machin.id}_{machin.name}.txt'
    fpl=f'media/log/{machin.id}_{machin.name}.txt'
    try:
        log=models.Log.objects.get(machin=machin)
    except:
        log=models.Log()
        log.machin=machin
        log.history.name=fph
        log.log.name=fpl
        log.save()

```

```

if machin.history_save:
    f = open(fph, "a")
    f.write "[" + datetime.now().strftime("%a, %d %b %Y %H:%M:%S +0000") + "]"
    "+cmd+"\n")
    f.close()

if machin.log_save:
    f = open(fpl, "a")
    f.write(f"[{datetime.now().strftime('%a, %d %b %Y %H:%M:%S +0000')}]"connect
    {usr} to {str(machin.ip.ip)}:{machin.port}\n")
    f.close()

print(log)

return cmdout

def Logout(request):
    logout(request)
    return redirect('url-singin')

#StatusCustom

def handler400(request,exception):
    response = render(request, "Page/Status.html", {"status": 400})
    response.status_code = 400
    return response

def handler403(request,exception):
    response = render(request, "Page/Status.html", {"status": 403})
    response.status_code = 403
    return response

def handler404(request,exception):
    response = render(request, "Page/Status.html", {"status": 404})
    response.status_code = 404

```

```

return response
def handler500(request):
    response = render(request, "Page/Status.html", {"status": 500})
    response.status_code = 500
    return response

```

### 4.2.3 Static

```

l_admin/gui/static/style.css
body{
    margin-bottom:5vh;
}
footer {
    position: fixed;
    left: 0;
    bottom: 0;
    width: 100%;
    background-color: #000D2D;
}
footer a,footer a:link,footer a:active,footer a:visited,footer a:hover {
    color: white;
    font-size: 2vh;
}
button{
    color: white;
    width: 100px;
    height: 40px;
    background: #0C00C0;

```

```
border-radius: 5px;
text-align: center;
}
input,
input:-webkit-autofill,
input:-webkit-autofill:hover,
input:-webkit-autofill:focus,
textarea:-webkit-autofill,
textarea:-webkit-autofill:hover,
textarea:-webkit-autofill:focus,
select:-webkit-autofill,
select:-webkit-autofill:hover,
select:-webkit-autofill:focus {
    -webkit-text-fill-color: black;
    -webkit-box-shadow: 0 0 0px 1000px white inset;
    border-radius: 0.4vh;
}
.center{
    width: 100%;
    display: grid;
    place-items: center;
}
.box{
    box-shadow: 0 0 0.8vh 0.1vh black;
    border-radius: 0.8vh;
}
```

## 4.2.4 Templates

### 4.2.4.1 MainTemplate.html

l\_admin/gui/templates/Sample/MainTemplate.html

```
{% load static %}
<!DOCTYPE html>
<html>
  <head>
    <link href="{% static 'style.css' %}" rel="stylesheet"/>
    <link rel="icon" href="{% static "logo.png" %}" {% static
"images/logo.png" %}/>
    <meta content="width=device-width, initial-scale=1" name="viewport" />
    <title>
      {% block title%}
      {% endblock title%}
    </title>
  </head>
  <body>
    <header>
    </header>
    <main>
      {% block main%}
      {% endblock main%}
    </main>
    <footer style=" text-align: left;">
      <a href="{%url 'admin:index' %}">Admin</a>
      <a href="{%url 'url-list' %}">List</a>
```

```
<a href="{%url 'url-logout' %}"> Logout</a>
```

```
</footer>
```

```
</body>
```

```
</html>
```

## 4.2.5 Страницы

### 4.2.5.1 Connect.html

```
l_admin/gui/templates/Page/Connect.html
```

```
{% extends 'Sample/MainTemplate.html' %}
```

```
{% block title%
```

```
    Подключение
```

```
{% endblock title%
```

```
{% block main%
```

```
<style>
```

```
    button,input{
```

```
        height: 4vh;
```

```
        font-size: 2vh;
```

```
        width: 5%;
```

```
    }
```

```
    div{
```

```
        margin: 0.8vh;
```

```
        text-align: center;
```

```
    }
```

```
    body {
```

```
        height: 100%;
```

```
        overflow-y: hidden;
```

```
    }
```

```

</style>
<div class="center" style="min-height: 100vh;">
    <div class="box" style="place-items: start; display: flex; flex-direction: column;
justify-content: flex-start; width: 80%; height:80%; ">    <div class="box" style="padding:
5px;margin: 35px; place-items: start; display: flex; flex-direction: column; justify-content:
flex-start; width: 80%; height:80%; text-align: left; overflow-y: scroll;">
        {{cmdout | linebreaksbr}}
    </div>
    <form action = "" method = "POST" style="width: 80%; text-align: left; padding:
35px; padding-top: 0px;">
        {% csrf_token %}
        <input id="cmdin" style="width: 80%;" type="text" name="cmdin" />
        <button type="submit">enter</button>
    </div>
</div>
{% endblock main%}

```

#### 4.2.5.2 MachinList.html

```

l_admin/gui/templates/Page/MachinList.html
{% extends 'Sample/MainTemplate.html' %}
{% block title%}
    Войти
{% endblock title%}
{% block main%}
<style>
    button,input{
        height: 4vh;
        width: 10vh;

```



```

        font-size: 2vh;
    }
    div{
        margin: 0.8vh;
        text-align: center;
    }
    input{
        width:19vh;
    }
    .tf input{
        height: 2vh;
        width: 2vh;
        font-size: 2vh;
    }
</style>
<div class="center" style="min-height: 90vh;">
    <div class="box" style="place-items: start; display: flex; flex-direction: row;
justify-content: flex-start; width: 80%; height:80%; ">
        <table>
        <tr>
            <th>Имя</th>
            <th>Группа</th>
            <th>Ip</th>
            <th>Port</th>
            <th>Имя</th>
            <th>Пароль</th>
            <th>Обновить</th>

```

```

<th>Удалить</th>

<th>Подключится</th>

</tr>

{%for machin in machins%}

<tr>

<form method="post" onsubmit="return confirm('Вы уверены?');">
    {% csrf_token %}
    <td>{{ machin.form.name }}</td>
    <td>{{ machin.form.group }}</td>
    <td>{{ machin.form.ip }}</td>
    <td>{{ machin.form.port }}</td>
    <td>{{ machin.form.username }}</td>
    <td>{{ machin.form.password }}</td>
    <input type="hidden" value="{{ machin.id }}" name="id">
    <td><button formaction="{% url 'url-list' machin.id %}"
type="submit">Обновить</button></td>
    <td><button formaction="{% url 'url-list' 0 %}"
type="submit">Удалить</button></td>

</form>

<td>

<a href="{% url 'url-connect-ssh' machin.id
%}">Подключится</a>

</td>

</tr>

{%endfor%}

<tr>

<th>Имя</th>

<th>Группа</th>

```

```

<th>Ip</th>
<th>Port</th>
<th>Имя</th>
<th>Пароль</th>
<th>Лог</th>
<th>История</th>
<th>Добавить</th>
</tr>
<tr>
<form method="post" onsubmit="return confirm('Вы уверены?');">
    {% csrf_token %}
    <td>{{ form.name }}</td>
<td>{{ form.group }}</td>
    <td>{{ form.ip }}</td>
    <td>{{ form.port }}</td>
    <td>{{ form.username }}</td>
    <td>{{ form.password }}</td>
    <td class="tf">{{ form.log_save }}</td>
    <td class="tf">{{ form.history_save }}</td>
    <td><button formaction={% url 'url-list' 0%}
type="submit">Добавить</button></td>
</form>
</tr>
<table>
<div style="display: flex; flex-direction: column; width: 10vh">
    <form method="post">
        {% csrf_token %}

```

```

<select name="group_filter" id="group_filter" multiple>
    {%for machin_group in machin_groups%}
        <option
value='{{machin_group.id}}'>{{machin_group.name}}</option>
    {%endfor%}
</select>
    <button formaction={% url 'url-list'%}
type="submit">Фильтр</button>
</form>
</div>
</div>
{% endblock main%}

```

### 4.2.5.3 SingIn.html

```

l_admin/gui/templates/Page/SingIn.html
{% extends 'Sample/MainTemplate.html' %}
{% block title%}
    Войти
{% endblock title%}
{% block main%}
<style>
    button,input{
        height: 4vh;
        width: 30vh;
        font-size: 2vh;
    }

```

```

div{
    margin: 0.8vh;
    text-align: center;
}
</style>
<form method="post" class="center" style="min-height: 90vh">
    {% csrf_token %}
    <div class="box" style="display: flex; flex-direction: column; justify-content:
space-around;">
        <div>
            <input type="text" name="username" placeholder="Логин" required>
        </div>
        <div>
            <input type="password" name="password" placeholder="Пароль" required>
        </div>
        <div>
            <button type="submit">Войти</button>
        </div>
    </div>
</form>
{% endblock main%}

```

#### 4.2.5.4 Status.html

```

l_admin/gui/templates/Page/Status.html
{% extends 'Sample/MainTemplate.html' %}
{% block title%}
    {{status}}

```

```
{% endblock title%}
```

```
{% block main%}
```

```
<div class="center" style="min-height: 60vh">
```

```
    <div>
```

```
        <h1 style="text-align: justify; font-size: 800%">{{status}}</h1>
```

```
        <h2 style="text-align: justify; font-size: 400%">Ooops!</h2>
```

```
    </div>
```

```
</div>
```

```
{% endblock main%}
```

# ПРИЛОЖЕНИЕ Б. СЦЕНАРИЙ И РЕЗУЛЬТАТЫ ТЕСТОВЫХ ИСПЫТАНИЙ

## АННОТАЦИЯ

### 1. Сценарий тестов

В данном разделе представлен сценарий тестов

### 2. Тесты jmeter

В данном разделе представлены тесты jmeter

### 3. Ручное тестирование

В данном разделе представлено ручное тестирование

### 4. Unit тесты

В данном разделе представлены unit тесты

#### 4.1 Результат Unit тестов

В данном разделе представлен результат unit тестов

#### 4.2 Unit тесты исходный код

В данном разделе представлен исходный код unit тестов

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1. Сценарий тестов.....	4
2. Тесты jmeter.....	5
3. Ручное тестирование.....	6
4. Unit тесты.....	6
4.1 Результат Unit тестов.....	6
4.2 Unit тесты исходный код.....	6



## **ВВЕДЕНИЕ**

В данном разделе документации описаны тесты программы l\_admin

## 1.Сценарий тестов

Рисунок 1-схема тестирования

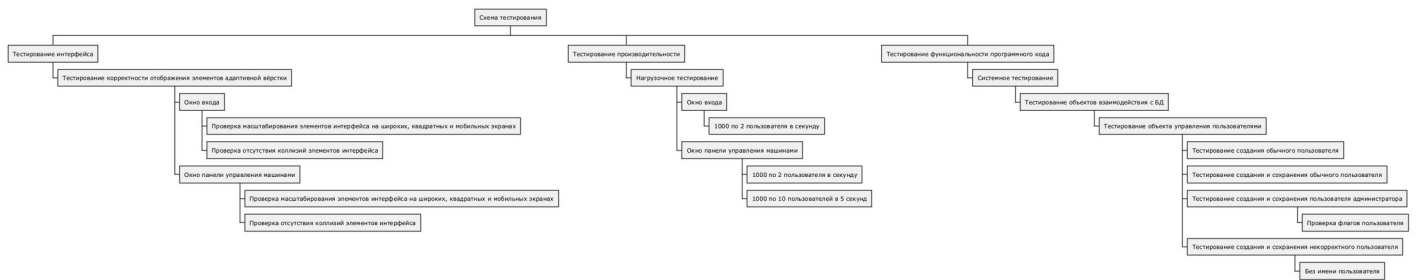


Рисунок 1-схема тестирования

## **2.Тесты jmeter**

Тесты jmeter находятся l\_admin/test/result/index.html

### 3.Ручное тестирование

№	Предпосылки	Приоритетность	Шаги тестирования	Данные тестирования	Ожидаемый результат	Фактический результат
1	Проверка отображения элементов окна входа	С	Открытие приложения по url /, изменение размера окна	Не предусмотрены	Интерфейс отображается корректно	Интерфейс отображается корректно
2	Проверка отображения элементов окна панели управления	С	Открытие приложения по url /login, изменение размера окна	Не предусмотрены	Данные успешно изменены и отображаются корректно	Данные успешно изменены и отображаются корректно

### 4.Unit тесты

#### 4.1 Результат Unit тестов

```
(env) [we@wePC l_admin]$ python manage.py test API
Found 5 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....
-----
Ran 5 tests in 0.807s

OK
Destroying test database for alias 'default'...
(env) [we@wePC l_admin]$ python manage.py test
Found 5 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....
-----
Ran 5 tests in 0.821s

OK
Destroying test database for alias 'default'...
```

## 4.2 Unit тесты исходный код

```
l_admin/API/tests.py
from django.test import TestCase

from django.core.exceptions import ObjectDoesNotExist

from django.contrib.auth.hashers import make_password

from .models import User

class TC_UserManager(TestCase):
    def setUp(self):
        self.userManager = User.objects

    def test_CreateUser_regular_login_myuser_password_123(self):
        expected_login = "myuser"
        expected_password = "123"
        expected_is_staff = False
        expected_is_superuser = False

        self.userManager.create_user(login=expected_login, password=expected_password)
        user = self.userManager.get(login=expected_login)
        salt = user.password.split('$')[2]
        actual_login = user.login
        actual_password = user.password
        actual_is_staff = user.is_staff
```

```
actual_is_superuser = user.is_superuser
```

```
self.assertEqual(expected_login, actual_login)
```

```
self.assertEqual(make_password(expected_password, salt=salt), actual_password)
```

```
self.assertEqual(expected_is_staff, actual_is_staff)
```

```
self.assertEqual(expected_is_superuser, actual_is_superuser)
```

```
def test_CreateUser_must_have_a_login(self):
```

```
    expected_password = "123"
```

```
    expected_exception_message = "Users must have a login"
```

```
    with self.assertRaises(ValueError) as catch:
```

```
        self.userManager.create_user(login=None, password=expected_password)
```

```
    actual_exception_message = str(catch.exception)
```

```
    self.assertEqual(expected_exception_message, actual_exception_message)
```

```
def test_CreateUser_when_commit_is_false_is_not_committed(self):
```

```
    expected_exception_message = "User matching query does not exist."
```

```
    login = "myuser"
```

```
    password = "123"
```

```
    self.userManager.create_user(login=login, password=password, commit=False)
```

```
    with self.assertRaises(ObjectDoesNotExist) as catch:
```

```
        self.userManager.get(login=login)
```

```
    actual_exception_message = str(catch.exception)
```

```
self.assertEqual(expected_exception_message, actual_exception_message)
```

```
def test_CreateSuperuser_superuser_login_mysuperuser_password_123(self):
```

```
    expected_login = "mysuperuser"
```

```
    expected_password = "123"
```

```
    expected_is_staff = True
```

```
    expected_is_superuser = True
```

```
    self.userManager.create_superuser(login=expected_login,  
password=expected_password)
```

```
    superuser = self.userManager.get(login=expected_login)
```

```
    salt = superuser.password.split('$')[2]
```

```
    actual_login = superuser.login
```

```
    actual_password = superuser.password
```

```
    actual_is_staff = superuser.is_staff
```

```
    actual_is_superuser = superuser.is_superuser
```

```
    self.assertEqual(expected_login, actual_login)
```

```
    self.assertEqual(make_password(expected_password, salt=salt), actual_password)
```

```
    self.assertEqual(expected_is_staff, actual_is_staff)
```

```
    self.assertEqual(expected_is_superuser, actual_is_superuser)
```

```
def test_CreateSuperuser_must_have_a_login(self):
```

```
    expected_password = "123"
```

```
    expected_exception_message = "Users must have a login"
```

```
    with self.assertRaises(ValueError) as catch:
```

```
self.userManager.create_superuser(login=None, password=expected_password)
```

```
actual_exception_message = str(catch.exception)
```

```
self.assertEqual(expected_exception_message, actual_exception_message)
```

```
class TC_User(TestCase):
```

```
    pass
```



## ПРИЛОЖЕНИЕ В. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

### **АННОТАЦИЯ**

#### 1. Инструкция по установке

В данном разделе описаны инструкции по установке

#### 2. Инструкция по эксплуатации

В данном разделе описаны инструкции по эксплуатации

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ.....	3
1. Инструкция по установке.....	4
2. Инструкция по эксплуатации.....	5

## **ВВЕДЕНИЕ**

В данном разделе документации описано использование программы l\_admin

## 1. Инструкция по установке

```
python -m venv <path-env>
```

```
source <path-env>/bin/activate
```

Активация виртуального окружения

```
pip install -r requirements.txt
```

Установка пакетов

```
python manage.py makemigrations
```

```
python manage.py migrate --run-syncdb
```

Сделать миграции в файле l\_admin/settings.py настроить бд

```
python manage.py createsuperuser
```

Создание суперпользователя

```
python manage.py runserver [::]:80
```

Запуск сервера на 80 порту разрешить подключение со всех ip

«Данная программа поддерживает ipv6»

```
python manage.py postgres_backup
```

Создание бекапа бд

```
python manage.py postgres_restore --clean --if-exists
```

Восстановление бекапа бд

## **2.Инструкция по эксплуатации**

При помощи веббраузера подключится к web интерфейсу программы после авторизоваться добавить машину далее подключится к ней и отправлять команды.

## ПРИЛОЖЕНИЕ Г. ТЕКСТ ПРОГРАММЫ

### **АННОТАЦИЯ**

1. Views.py

Скрипт взятия данных

2. Models.py

Скрип создания таблиц

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1. Views.py.....	4
2. Models.py.....	10

## **ВВЕДЕНИЕ**

В данном разделе документации описан скрипт базы данных программы l\_admin.

Поскольку postgresql имеет нативную интеграцию с python в данном приложении будут представлен код отвечающий за работу с бд.



**1.Views.py**

```
l_admin/gui/views.py
```

```
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login, logout
from django.http import HttpResponseRedirect, Http404, HttpResponse
from API import models
import paramiko
import time
from datetime import datetime
# Create your views here.
def SingIn(request):
    if request.method == 'GET':
        context = ""
        return render(request, 'Page/SingIn.html', {'context': context})

    elif request.method == 'POST':
        username = request.POST.get('username', "")
        password = request.POST.get('password', "")

        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return HttpResponseRedirect('/list')
        else:
            context = {'error': 'Wrong credintials'} # to display error?
            return render(request, 'Page/SingIn.html', {'context': context})
```

```
def List(request,id=None):
```

```
    machins=models.Machin.objects.filter(user=request.user,user_monitor_permission=True)|
    models.Machin.objects.filter(other_monitor_permission=True)|
    models.Machin.objects.filter(group__in=
    models.Group.objects.filter(users=request.user),group_monitor_permission=True)
```

```
    machins=machins.order_by('id')
```

```
    if not request.user.is_authenticated: raise Http404
```

```
    if request.method == 'GET':
```

```
        return List_render(request,machins)
```

```
    elif request.method == 'POST':
```

```
        if 'id' in request.POST and id==0:
```

```
models.Machin.objects.get(id=request.POST['id']).delete()
```

```
        elif 'group_filter' in request.POST:
```

```
machins=models.Machin.objects.all().filter(machin_group__id=request.POST['group_filter'])
```

```
    return List_render(request,machins)
```

```
elif id!=None and id!=0:
```

```
    machin=models.Machin.objects.get(id=request.POST['id'])
```

```
    machin.name=request.POST['name']
```

```
    machin.group=models.Group.objects.get(id=request.POST['group'])
```

```
    machin.ip=request.POST['ip']
```

```
    machin.port=request.POST['port']
```

```
    machin.username=request.POST['username']
```

```
    if request.POST['password'] != '': machin.password=request.POST['password']
```

```
    machin.save()
```

```
elif id==0:
```

```
    machin=models.Machin()
```

```

    machin.name=request.POST['name']
    machin.group=models.Group.objects.get(id=request.POST['group'])
    machin.ip=request.POST['ip']
    machin.port=request.POST['port']
    machin.username=request.POST['username']
    machin.password=request.POST['password']
    machin.user=request.user
    machin.history_save=bool(request.POST['history_save'])
    machin.history_save=bool(request.POST['history_save'])
    machin.save()

    return List_render(request,machins)
def List_render(request,machins):
    return render(request, 'Page/MachinList.html', {
        "machins": machins,
        "form":models.MachinForm(),
        "machin_groups":models.Machin_Group.objects.all()
    })
def connect_ssh(request,id):
    if not request.user.is_authenticated: raise Http404
    cmdout=""
    usr=request.user
    if request.method == 'POST':
        try:
            cmdout=ssh(id,usr,request.POST['cmdin'])
            print(cmdout)
        except:
            cmdout="ERROR connect"

```

```

elif request.method == 'GET':
    try:
        cmdout=ssh(id,usr,"echo \"l_admin connect\"")
    except:
        cmdout="ERROR connect"
    return render(request, 'Page/Connect.html',{'cmdout':cmdout})
#no render
def ssh(id,usr,cmd):
    machin=models.Machin.objects.get(id=id)
    ssh_ = paramiko.SSHClient()
    ssh_.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh_.connect(str(machin.ip.ip), port=machin.port,
username=machin.username,password=machin.password, timeout=3)
    (stdin, stdout, stderr) = ssh_.exec_command(cmd)
    cmdout=stdout.read().decode("utf-8")
    ssh_.close()
    #log
    fph=f'media/machin/{machin.id}_{machin.name}.txt'
    fpl=f'media/log/{machin.id}_{machin.name}.txt'
    try:
        log=models.Log.objects.get(machin=machin)
    except:
        log=models.Log()
        log.machin=machin
        log.history.name=fph
        log.log.name=fpl
        log.save()

```

```

if machin.history_save:
    f = open(fph, "a")
    f.write "[" + datetime.now().strftime("%a, %d %b %Y %H:%M:%S +0000") + "]"
    "+cmd+"\n")
    f.close()

if machin.log_save:
    f = open(fpl, "a")
    f.write(f"[{datetime.now().strftime('%a, %d %b %Y %H:%M:%S +0000')}]connect
{usr} to {str(machin.ip.ip)}:{machin.port}\n")
    f.close()

print(log)

return cmdout

def Logout(request):
    logout(request)
    return redirect('url-singin')

#StatusCustom

def handler400(request,exception):
    response = render(request, "Page/Status.html", {"status": 400})
    response.status_code = 400
    return response

def handler403(request,exception):
    response = render(request, "Page/Status.html", {"status": 403})
    response.status_code = 403
    return response

def handler404(request,exception):
    response = render(request, "Page/Status.html", {"status": 404})
    response.status_code = 404

```

```
    return response

def handler500(request):
    response = render(request, "Page/Status.html", {"status": 500})
    response.status_code = 500
    return response
```

**2.Models.py**

l\_admin/API/models.py — таблицы в бд

```

from django.db import models

from django.contrib.auth.models import AbstractBaseUser, BaseUserManager,
PermissionsMixin

from django.db import models

from django.core.validators import MaxValueValidator, MinValueValidator

from django.utils import timezone

from django.utils.translation import gettext_lazy as _

from colorfield.fields import ColorField

from datetime import timedelta, datetime

from netfields import InetAddressField, NetManager

from django import forms

class UserManager(BaseUserManager):

    def create_user(self, login, history_save=True, password=None, commit=True):

        if not login: raise ValueError(_('Users must have a login'))

        user = self.model(login=login, history_save=history_save)

        user.set_password(password)

        if commit: user.save(using=self._db)

        return user

    def create_superuser(self, login, password):

        user = self.create_user(password=password, login=login, commit=False)

        user.is_staff = True

        user.is_superuser = True

        user.save(using=self._db)

        return user

class User(AbstractBaseUser, PermissionsMixin):

```

```

login = models.CharField(_('login'), max_length=150, blank=True, unique=True)

is_active = models.BooleanField(_('active'), default=True, help_text=_('Designates
whether this user should be treated as active. Unselect this instead of deleting accounts.'),)

is_staff = models.BooleanField(_('staff status'), default=False, help_text=_('Designates
whether the user can log into this admin site.'),)

is_observer = models.BooleanField(default=False)

history_save = models.BooleanField(default=True)

history = models.FileField()

date_joined = models.DateTimeField(_('date joined'), default=timezone.now)

objects = UserManager()

USERNAME_FIELD = 'login'

def get_full_name(self):
    return self.login

def __str__(self):
    return self.login

class Group(models.Model):
    users = models.ManyToManyField(User)
    name = models.CharField(max_length=64, unique=True)
    description = models.TextField(max_length=512)

class Machin(models.Model):
    name = models.CharField(max_length=64)
    username = models.CharField(max_length=64)
    password = models.CharField(max_length=64)
    user = models.ForeignKey(User, on_delete=models.PROTECT)
    group = models.ForeignKey(Group, on_delete=models.PROTECT)
    history_save = models.BooleanField(default=True)
    log_save = models.BooleanField(default=True)

```



```

ip = InetAddressField()

port=models.IntegerField(validators=[MaxValueValidator(65535),MinValueValidator(1)])

objects = NetManager()

user_monitor_permission=models.BooleanField(default=True)
user_user_permission=models.BooleanField(default=True)
group_monitor_permission=models.BooleanField(default=True)
group_user_permission=models.BooleanField(default=True)
other_monitor_permission=models.BooleanField(default=True)
other_user_permission=models.BooleanField(default=True)

def form(self):
    return MachinForm(instance=self)

def getip(self):
    ip=str(self.ip.ip)
    return ip

class MachinForm(forms.ModelForm):
    password = forms.CharField(widget=forms.PasswordInput(), required = False)
    group=None
    try:
        group = forms.Select(choices=Group.objects.all().values_list('id', 'name'))
    except:
        group=None
    class Meta:
        model = Machin
        fields = ('__all__')

class Log(models.Model):
    machin =models.ForeignKey(Machin, on_delete=models.CASCADE)

```

```
time_save=models.DateTimeField(auto_now_add=True)
```

```
log =models.FileField(editable=False,auto_created = True)
```

```
history =models.FileField(editable=False,auto_created = True)
```

```
class Machin_Group(models.Model):
```

```
    machin = models.ManyToManyField(Machin,related_name='machin_group')
```

```
    name = models.CharField(max_length=64, unique=True)
```

```
class Machine_request_one_comand(models.Model):
```

```
    machin_group=models.ForeignKey(Machin_Group, on_delete=models.CASCADE)
```

```
    machin=models.ForeignKey(Machin, on_delete=models.CASCADE)
```

```
    user=models.ForeignKey(User, on_delete=models.SET_NULL, null=True)
```

```
    color = ColorField()
```

```
    request_file_path=models.FileField(editable=False)
```