МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

РАБОТА ПРОВЕРЕНА
Рецензент
генеральный директор
ООО «Мегарендер»
_____ В.В. Юрков
«___»_____ 2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____Г.И. Радченко
«___»_____ 2019 г.

# Разработка системы управления компьютерными классами на базе технологии Django

## ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Консультант,
к.ф.-м.н., доцент каф. СП
_____ П.С. Костенецкий
«___»_____2019 г.

Руководитель работы,
к.ф.-м.н., зав. каф. ЭВМ
_____ Г. И. Радченко
«___»_____2019 г.

Автор работы,
студент группы КЭ-222
_____А.И. Рекачинский
«___»_____2019 г.

Нормоконтролер,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___» _____2019 г.

Челябинск-2019

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____2019 г.

# ЗАДАНИЕ
**на выпускную квалификационную работу магистра**
студенту группы КЭ-222
Рекачинскому Александру Игоревичу
обучающемуся по направлению
09.04.01 «Информатика и вычислительная техника»

1. **Тема работы: «**Разработка системы управления компьютерными классами на базе технологии Django» утверждена приказом по университету от 25 апреля 2019 г. №899
2. **Срок сдачи студентом законченной работы:** 10 июня 2019 г.
3. **Исходные данные к работе:**
   - Mark, L. Learning Python / L. Mark. 2013. – 1594 p.
   - Django documentation. – https://docs.djangoproject.com/en/2.1/. Дата обращения: 01.02.2019
   - Роббинс, Д. HTML5, CSS3 и JavaScript. Исчерпывающее руководство / Д. М. Роббинс. М: Эксмо, 2014. – 528 с.
4. **Перечень подлежащих разработке вопросов:**
   1. Изучить существующие отечественные и зарубежные аналоги систем управления компьютерными классами.
   2. Выполнить проектирование программной системы.
   3. Выполнить реализацию программной системы.

4. Выполнить функциональное тестирование разработанной программной системы.
5. Выполнить внедрение разработанной программной системы в лаборатории «Суперкомпьютерного Моделирования» ЮУрГУ.

5. **Дата выдачи задания:** 1 декабря 2018 г.

Руководитель работы _____ *Г.И. Радченко*

Студент _____ *А.И. Рекачинский*

КАЛЕНДАРНЫЙ ПЛАН

| Этап | Срок сдачи | Подпись руководителя |
|---|---|---|
| Введение и обзор литературы | 01.03.2019 | |
| Проектирование системы | 01.04.2019 | |
| Реализация системы | 01.05.2019 | |
| Тестирование, отладка, эксперименты | 15.05.2019 | |
| Компоновка текста работы и сдача на нормоконтроль | 24.05.2019 | |
| Подготовка презентации и доклада | 30.05.2019 | |

Руководитель работы _____ *Г.И. Радченко*

Студент _____ *А.И. Рекачинский*

Аннотация

Рекачинский А. И. Разработка системы управления компьютерными классами на базе технологии Django. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2019, 78 с., 45 ил., библиогр. список – 22 наим.

Работа состоит из введения, 5 глав, заключения, библиографического списка и приложения.

В первом разделе, «Анализ предметной области», изложено описание основных существующих на рынке решений, предназначенных для администрирования компьютерных классов.

Во втором разделе, «Определение требований», определены функциональные и нефункциональные требования к программной системе.

В третьем разделе, «Проектирование», приведены UML диаграммы разрабатываемой системы.

В четвертом разделе, «Реализация», приведены основные алгоритмы работы системы и описан веб интерфейс пользователя.

В пятом разделе, «Развертывание и тестирование», приведены результаты тестирования, подтверждающие корректность работы программной системы и основные параметры развертывания на серверах.

# ОГЛАВЛЕНИЕ

# ВВЕДЕНИЕ

В настоящее время во всех образовательных учреждениях используются компьютеры. Число компьютеров в учреждениях может достигать нескольких сотен. Администрирование такого количества компьютеров является трудоемкой задачей. Большую часть проблем администрирования решают специально разработанные программы. Но готовые программные продукты не всегда могут предложить именно тот функционал, который требуется в каждом конкретном случае. В таких случаях имеет смысл разработки собственного программного решения.

Данная работа посвящена разработке системы управления компьютерными классами на базе технологии Django. Новая система имеет дополнительный функционал, отсутствующий в коммерческих решениях, который требуется в компьютерных классах лаборатории «Суперкомпьютерного Моделирования» ЮУрГУ [11], такой как:

- карта расположения ПК в аудиториях;
- режим “Тест класс”;
- выбор режима загрузки компьютеров, для переключения в режим «Тест-класс».

# 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

В данном разделе выполнен обзор аналогов разрабатываемой системы, а также используемой технологии Django.

## 1.1. ОБЗОР АНАЛОГОВ

В данном разделе сделан обзор наиболее известных систем для удаленного управления компьютерами и выполнен сравнительный анализ программ.

*Total Network Inventory* – программа для автоматической инвентаризации компьютеров в сети [4] (рисунок 1).

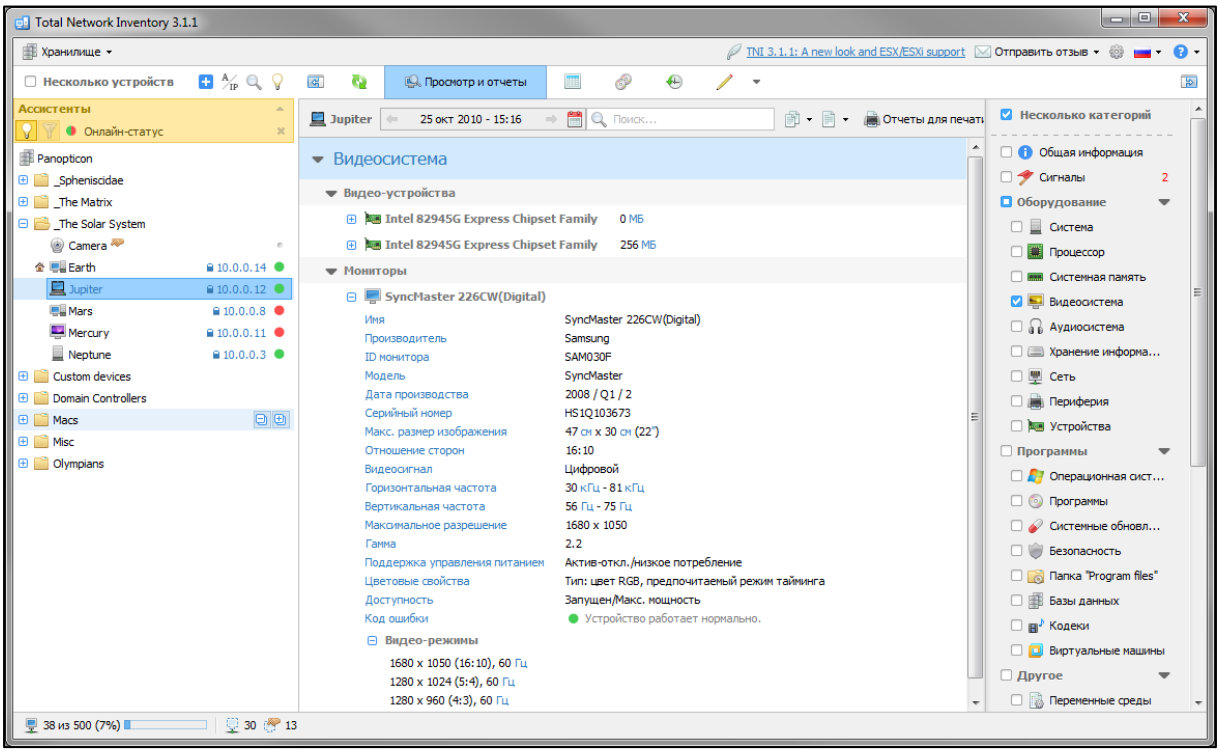


Рисунок 1 – Экранная форма Total Network Inventory

Не требует дополнительной установки клиента. Для работы нужно знать администраторский пароль, а также провести необходимые настройки в домене для разрешения управления компьютерами данной программой. Во время сканирования может собирать информацию о аппаратных характеристиках, подключенных устройствах, установленных программ и используемых лицензиях. Также в ней есть функционал, позволяющий гибко настраивать генерируемые отчеты по собираемым данным как для отдельных ПК, так и для

групп. Вся информация о ПК хранится в специально созданном файле размером в несколько сотен КБ, помимо всего, хранится и история изменения оборудования. Из главных рассматриваемых возможностей присутствуют функции для удаленного управления компьютерами, позволяющие выполнять включение, выключение и перезагрузку ПК.

*10-Страйк: Удаленный Доступ* – в ходе анализа было выявлено что данная система практически полностью имеет аналогичный функционал по сравнению с Total Network Inventory [5]. За тем исключением, что в этой программе можно удаленно заблокировать мышь, клавиатуру и просматривать рабочий стол удаленного ПК в режиме “Только просмотр”. Также есть возможность удаленного подключения через VPN и использование чатов как текстовых, так и голосовых с администрируемым ПК (рисунок 2).

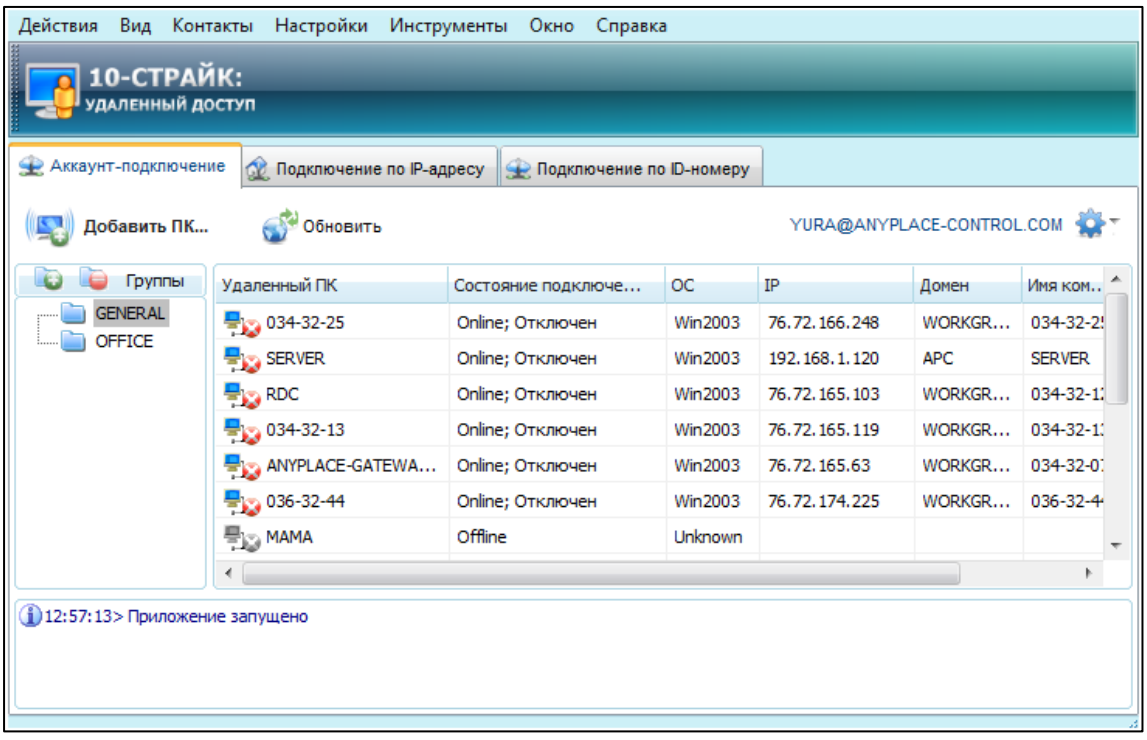


Рисунок 2 – Экранная форма 10-Страйк: Удаленный Доступ

*TeamViewer* – одна из популярных программ для удаленного доступа [6]. Для управления удаленным компьютером на нем нужно установить данное приложение, а затем зная ID и пароль подключиться. Она позволяет передавать файлы, смотреть демонстрацию экрана и управлять ПК. Используется простой понятный интерфейс. Не подходит для группового управления ком-

пьютерами из-за необходимости выполнять подключение к каждому из них для его выключения. Лицензия не позволяет бесплатно использовать данную программу в организациях (рисунок 3).

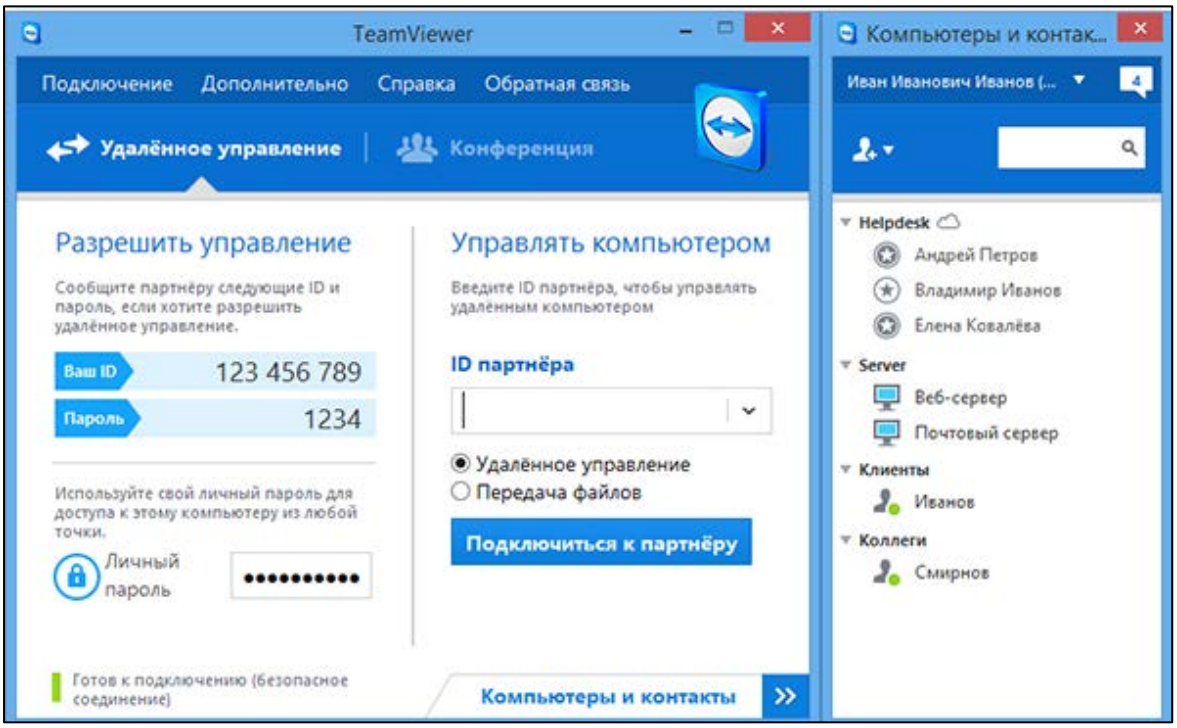


Рисунок 3 – Экранная форма TeamViwer

*LiteManager* – простая программа для удаленного управления компьютерами [7], которая состоит из двух основных компонентов (рисунок 4).

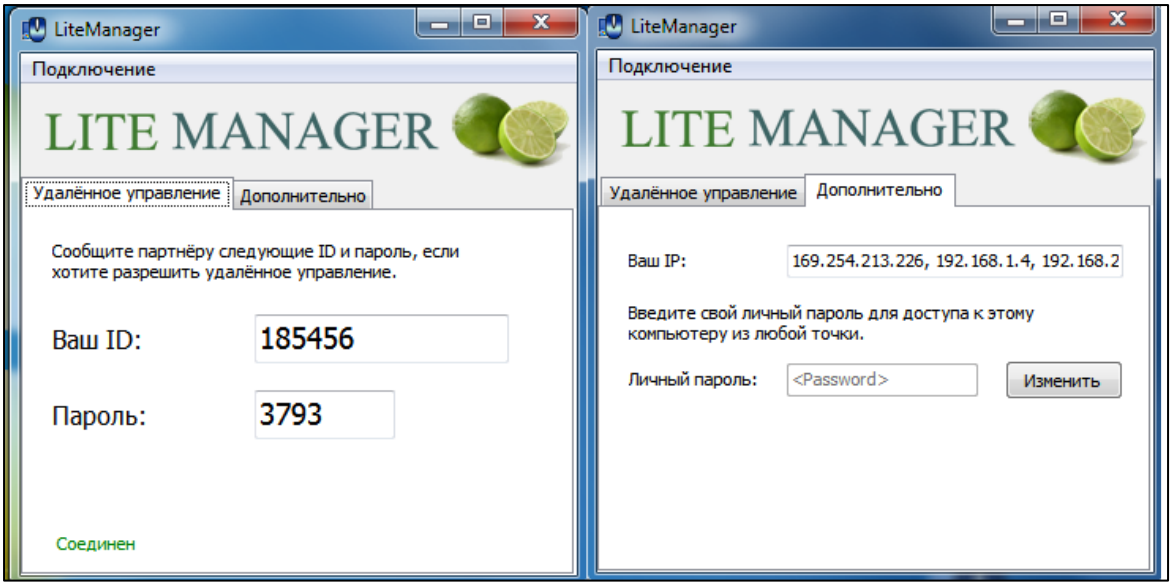


Рисунок 4 – Экранная форма LiteManager

Для удаленной работы требуется установка пользовательского модуля программы ROMServer, на удаленные компьютеры, после чего будет необходимо узнать ID и пароли от компьютеров. Также есть функционал для передачи файлов, управления реестром, запись экрана, удаленная установка ПО. Лицензия позволяет бесплатно использовать данную программу для 30 ПК.

*Ammy admin* – данная программа является аналогом выше рассмотренных программ таких как TeamViewer, LiteManager[8]. Отличается более скудным функционалом позволяющего выполнять только удаленное управление рабочим столом (рисунок 5).

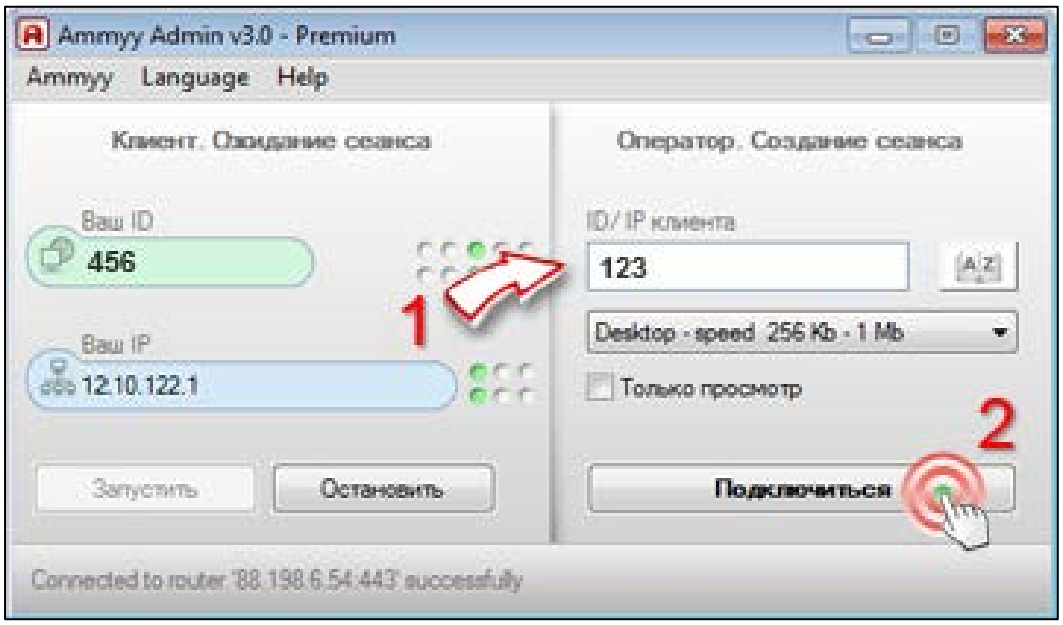


Рисунок 5 – Экранная форма Ammy admin

*RAdmin* повторяет часть функционала программы TeamViewer, среди которого есть удаленное управление рабочими столами и передача файлов [9]. Из отличий можно вынести удаленный доступ с машин, где установлена ОС Linux. Нет бесплатной версии, доступна только на 30 дней (см. рисунок 6).

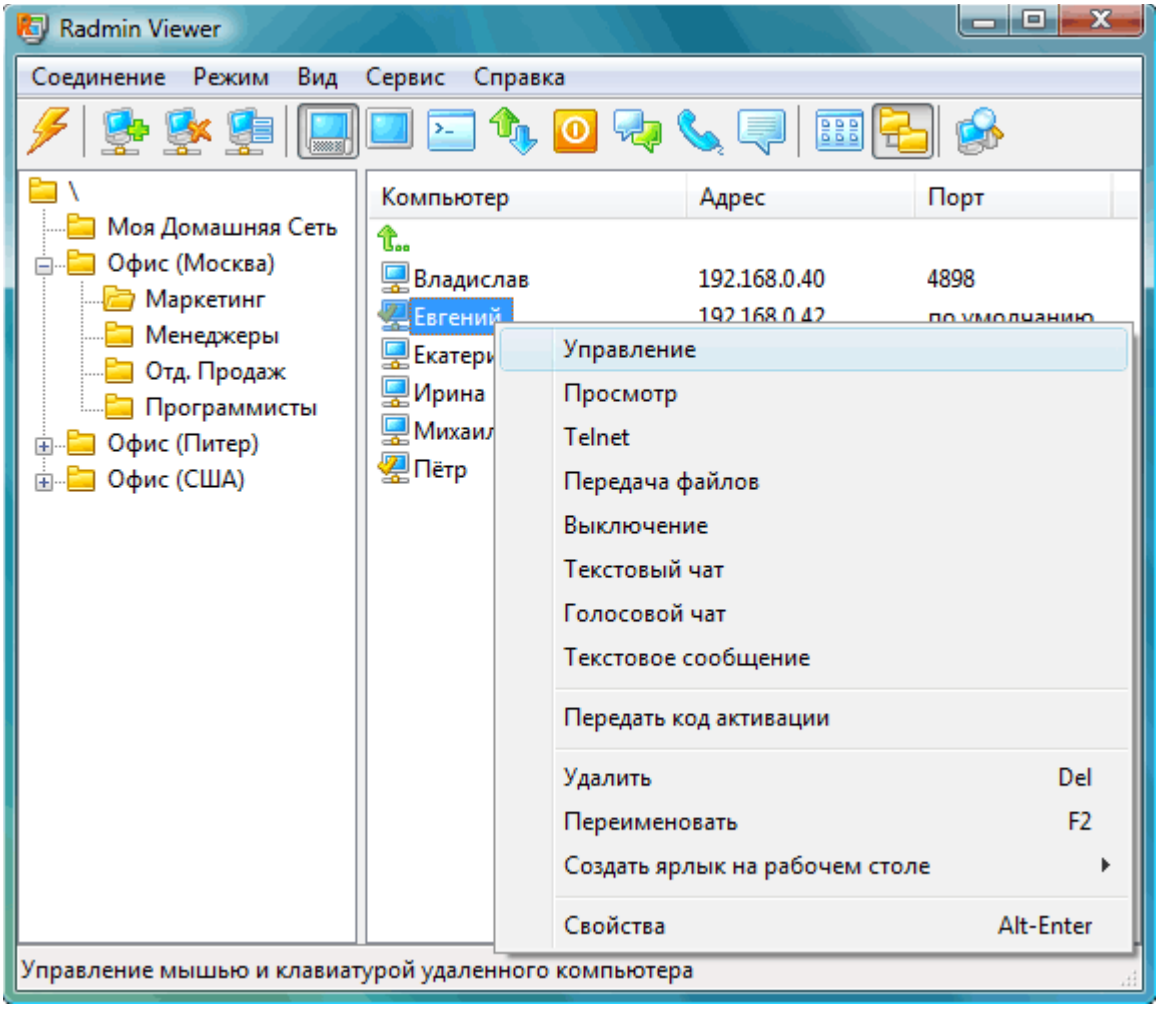


Рисунок 6 – Экранная форма Radmin Viewer

По результатам исследований возможностей, рассмотренных выше аналогов, была составлена таблице 1, позволяющая провести сравнительный анализ программ.

Таблица 1– Сравнение аналогов

| Необходимые требования | Total Network Inventory | 10-Страйк: Удаленный Доступ | TeamViewer | LiteManager | Ammy admin | RAdmin |
|---|---|---|---|---|---|---|
| Включение одного ПК | + | + | + | - | - | - |
| Включение групп ПК | + | - | - | - | - | - |
| Выключение одного ПК | + | + | + | + | + | + |
| Выключение групп ПК | + | - | - | - | - | - |
| Перезагрузка одного ПК | + | + | + | + | + | + |
| Перезагрузка групп ПК | + | - | - | - | - | - |

Продолжение таблицы 1

| | | | | | | |
|---|---|---|---|---|---|---|
| Отображение карты расположения ПК в аудиториях | - | - | - | - | - | - |
| Поддержка PXE-загрузки компьютеров по сети | - | - | - | - | - | - |
| Режим “Тест - класс” | - | - | - | - | - | - |
| Цена лицензии для 35 ПК (руб.) | 5 400 в год | 24 000 | 20 388 | 13 750 | 60 000 | 38 000 |

Исходя из обзора аналогов в таблице 1, видно, что аналоги хоть и позволяют выполнять часть необходимых функций, но их серьезными недостатками являются отсутствие опций:

- показ карты ПК в аудиториях;
- ограничение возможностей в управлении ПК для проведения тестирования студентов;
- нет управления режимом загрузки ПК (PXE сервер или жесткий диск).

## 1.2. АНАЛИЗ ТЕХНОЛОГИИ DJANGO

Фреймворки стали неотъемлемой частью веб-разработки, так как стандарты веб-приложений постоянно растут, как и сложность необходимой технологии. Совершенно неразумно изобретать велосипед для таких сложных технологий – поэтому использование фреймворков, одобренных тысячами разработчиков по всему миру, является правильным подходом для создания многофункциональных и интерактивных веб-приложений. У веб-приложения есть бэкэнд (на стороне сервера) и фронтенд (внешний интерфейс на стороне клиента). Так как основная часть данной работы заключается в разработке серверной части, поэтому были рассмотрены только бэкенд фрейморки.

*Django*

Django [10, 15, 17] – это веб-фреймворк MVT, который позволяет быстро разрабатывать безопасные и легко поддерживаемые веб-приложения. Созданный опытными разработчиками, Django берет на себя большую часть начальных работ веб-разработки, поэтому разработчики могут сосредоточиться на написании своего приложения без необходимости писать все базовые функции с нуля. Он бесплатный и с открытым исходным кодом, имеет активное сообщество, благодаря чему является одним из самых известных веб-фреймворков в мире, а также он имеет подробную документацию и множество вариантов бесплатной и платной поддержки. На нем разработаны такие популярные сайты как Instagram, YouTube, Google, NASA.

Django следует архитектуре MVT, которая обозначает Model-View-Template. MVT - это вариация знаменитой структуры MVC от Django, поэтому она совершенно аналогична тому, как работают другие фреймворки. Когда сервер Django получает запрос, маршрутизатор URL отображает запрос в соответствующее представление. Затем представление выбирает данные через модели, заполняет шаблон и отправляет его обратно пользователю. Почти при всех запросах от пользователя выполненных посредством веб-браузера, происходит следующий набор действий (см. рисунок 7):

1. Пользователь обращается к модулю маршрутизации URL.
2. Django перебирает URL шаблоны и при совпадении вызывает функцию в модуле представления.
3. Модуль представления в соответствии с вызванной функцией возвращает HTML страницу и при необходимости запрашивает данные из базы данных.

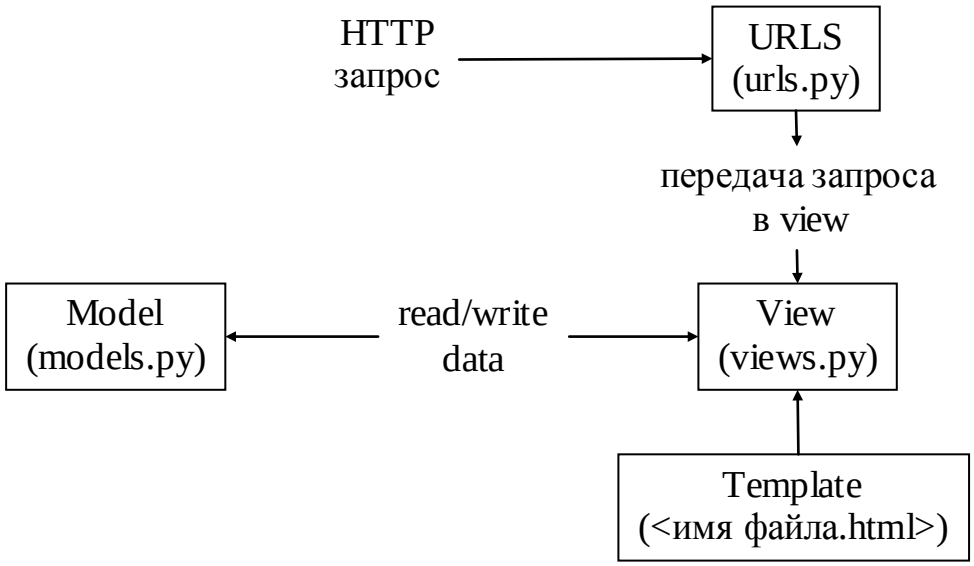


Рисунок 7 – Django Model View Template (MVT)

Модель Django использует высокий уровень Object-Relational Mapping (объектно-реляционное отображение), который упрощает работу с базой данных и данными, а также ускоряет процесс разработки. Без Object-Relational-Mapping разработчикам придется самим создавать таблицы и определять запросы или процедуры, которые иногда переводятся в изрядное количество SQL, которое может быть сложным и трудно отслеживаемым. Слой ORM позволяет записывать все определения таблиц в простом коде на языке Python, а также обеспечивает его преобразование в соответствующий язык запросов и облегчает операции CRUD (create, read, update, delete). В отличие от других технологий, все модели размещаются в одном файле, обычно, models.py, что позволяет создавать большие и простые проекты.

ORM Django поддерживает несколько систем баз данных таких как PostgreSQL, MySQL и SQLite. SQLite удобен для тестирования и разработки,

потому что его можно использовать прямо из коробки без необходимости установки дополнительного программного обеспечения, а MySQL эффективен для продакшн версии за счет более высокой производительности.

*Безопасность*

Django помогает разработчикам избежать многих распространенных ошибок безопасности, предоставляя платформу, которая была разработана для того, чтобы «делать правильные вещи» для автоматической защиты сайта. Например, Django предоставляет безопасный способ управления учетными записями пользователей и их паролями, предотвращая часто используемые ошибки такие как непосредственное хранение паролей или помещение информации о сеансе в файлы cookie, где она уязвима, в Django вместо этого файлы cookie просто содержат ключ, а фактические данные хранятся в базе данных. вместо хэша пароля.

Хэш пароля - это шестнадцатеричное значение фиксированной длины, созданное с помощью криптографической хэш-функции. Django может проверить правильность введенного пароля, запустив его через хеш-функцию и сравнив выходные данные с сохраненным хеш-значением. Однако даже если хранимое хеш-значение скомпрометировано, злоумышленнику трудно определить исходный пароль.

Django по умолчанию обеспечивает защиту от многих уязвимостей, включая внедрение SQL-кода, межсайтовый скриптинг, подделку межсайтовых запросов и перехват кликов.

*Масштабируемость*

Django использует компонентную архитектуру микросервисов, каждый модуль архитектуры не зависит от других что позволяет заменить или изменить его при необходимости. Разделение между различными частями означает, что он может масштабироваться при необходимости увеличить производительность оборудования: серверы баз данных, серверы кэширования или серверы приложений. Некоторые из самых загруженных сайтов успешно мас-

штабировали Django, чтобы удовлетворить их требования, например, Instagram и Disqus.

*Обслуживаемость*

Код Django написан с использованием принципов и шаблонов проектирования, которые поощряют создание поддерживаемого и повторно используемого кода. В частности, он использует принцип «Не повторяйся», чтобы избежать ненужного дублирования, уменьшая объем кода. Django также способствует группированию связанных функциональных возможностей в многократно используемые приложения или модули, в соответствии с шаблоном Model View Controller (MVC).

*Портативный*

Django написан на высокоуровневом языке программирования Python, который работает на многих популярных платформах. Благодаря этому разработчики не привязаны к какой-либо конкретной серверной платформе что позволяет запускать разработанные приложения на многих версиях Linux, Windows и Mac OS X. Кроме того, Django хорошо поддерживается многими провайдерами веб-хостинга, которые часто предоставляют определенную инфраструктуру и документация для размещения сайтов Django.

# 2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

Для развертывания данной системы необходим следующий набор подсистем.

1. Сервер с операционной системой Linux для размещения PXE сервера в котором будет храниться образ Linux Proteus Kiosk и модуля для управления этим PXE сервером (pxe-bot).
2. Сервер с операционной системой Linux для размещения Django сервера и главного модуля управления (class-controller).
3. Веб сервер Nginx HTTP Server для работы с статическими файлами.
4. СУБД MySQL для хранения базы данных о компьютерах и пользователях.
5. Сервер с операционной системой Windows для размещения службы управления компьютерами от имени доменного администратора (windows-bot).

## 2.1. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

По итогам опроса системных администраторов аудиторий 110/3г и 112/3г, были определены функциональные требования. Разработанное веб-приложение должно позволять:

1. в реальном времени отображать состояние компьютеров, а именно включены они или нет;
2) показывать карты расположения ПК в аудиториях;
3) показывать IP адреса компьютеров;
4) выключать, включать все, перегружать ПК в конкретном классе;
5) выключать, включать, перегружать конкретный ПК;
6) выбирать способ загрузки компьютеров, для переключения в режим «Тест-класс».

## 2.2. НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

Система должна соответствовать следующим нефункциональным требования:

1) количество обслуживаемых ПК: не менее 35;
2) система должна позволять делать выбор типа загрузки компьютера:
    a. локальный диск (Windows для работы студентов),
    b. сервер PXE (образ Linux Porteus Kiosk для тестирования студентов).
3) Режим тестирования должен обладать следующими особенностями:

– при включенном режиме тестирования у студентов не должно быть доступа к различным программам операционной системы кроме браузера, через который будет выполняться тест на сайте. К запрещенным программам относятся те что позволяют делать скриншоты тестов и ответов к ним для дальнейшей передачи другим студентам через интернет или USB-флеш-накопители. Также есть группы студентов, использующих собственноручно написанные программы с включенными в них материалами для написания тестов, которые могут запускаться без установки и находиться в скрытном, ожидающем режиме пока его разработчик не проведет необходимые действия для показа ответов;
– у студентов не должно быть доступа к сайтам отличным от тех на которых проходят тестирования;
– не должно быть возможности использовать любые носители информации;
– не должно быть возможности использовать какие-либо сочетания клавиш для вызова специального меню администратора;
– из приложений должен быть доступен только браузер Google Chrome;

– браузер Google Chrome при загрузке операционной системы, должен автоматически открываться на весь экран;
– в закладках браузера должны быть доступны две ссылки, а именно на сайт edu.susu.ru и калькулятор.
– требуется исключить возможности, при которых студенты технических специальностей при обычной работе за компьютерами могут внести изменения в режим написания тестов для получения нечестных результатов тестирования;

Из приведенных выше требований было решено использовать облегченный образ системы Linux Porteus Kiosk, используемый в терминалах различных банков или магазинов, так него есть возможность гибкой настройки конечного состава сборки, который решает проблемы с 1 по 7 пункт.

Для решения проблемы номер 8, было решено хранить образ Linux Porteus Kiosk [18] на сервере PXE, который позволит загружать образ по локальной сети на компьютеры при необходимости проведения тестирования.

# 3. ПРОЕКТИРОВАНИЕ

## 3.1. ВАРИАНТЫ ИСПОЛЬЗОВАНИЯ

Для проектирования системы использовался универсальный язык моделирования UML 2.5 [1, 2]. Диаграмма вариантов использования для разрабатываемой системы изображена на рисунке 8.

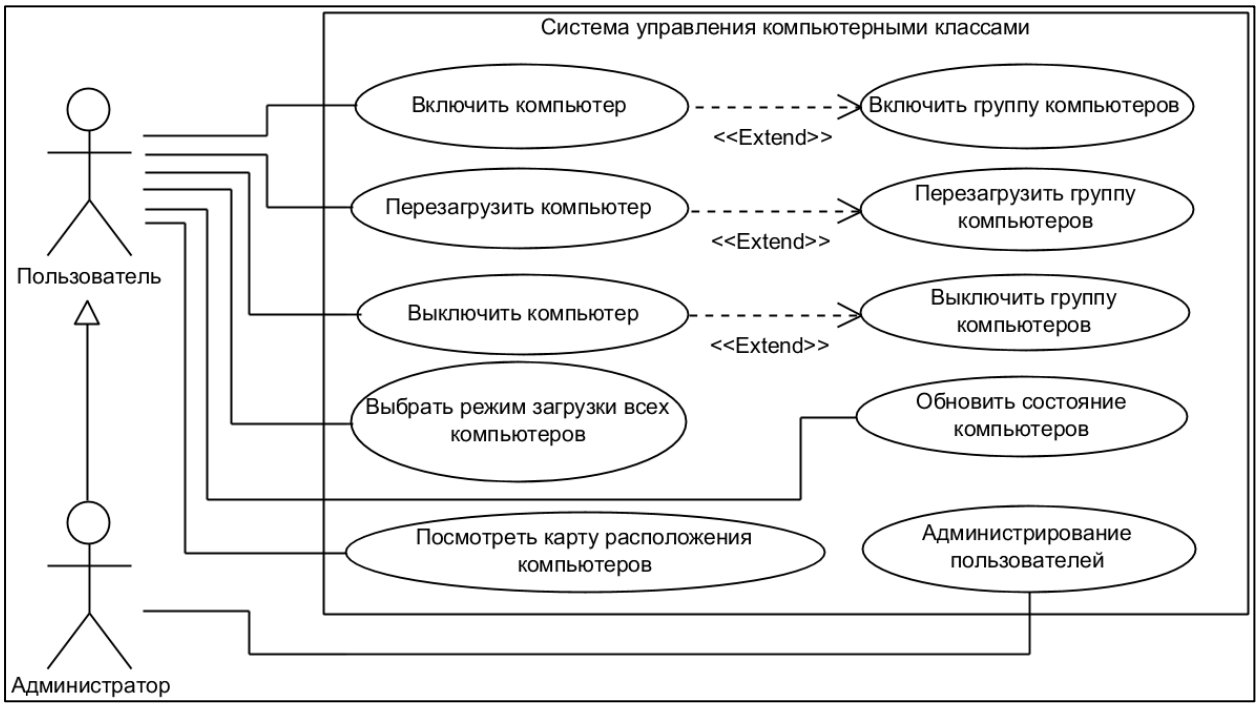


Рисунок 8 – Диаграмма вариантов использования

В системе имеется 2 вида действующих актеров: Пользователь и Администратор. Пользователь представляет собой человека, взаимодействующего с системой управления компьютерными классами.

Вариант использования «Включить компьютер». Вариант использования позволяет включить выбранный компьютер.

Основной поток событий состоит из четырех шагов:

1) пользователь в веб-интерфейсе нажимает на кнопку “Включить компьютер”;
2) сервер обрабатывает запрос;
3) на сервере формируется WOL пакет;
4) созданный пакет отправляется на выбранный ПК.

У варианта использования имеется одна точка расширения: Включить группу компьютеров.

Вариант использования «Включить группу компьютеров». Вариант использования включает группу компьютеров.

Основной поток событий состоит из четырех шагов:

1) пользователь в веб-интерфейсе нажимает на кнопку “Включить группу компьютеров”;
2) сервер обрабатывает запрос;
3) на сервере формируется WOL пакеты;
4) созданные пакеты отправляется на выбранную группу компьютеров.

Вариант использования «Перезагрузить компьютер». Вариант использования позволяет перезагрузить выбранный компьютер.

Основной поток событий состоит из четырех шагов:

1) пользователь в веб-интерфейсе нажимает на кнопку “Перезагрузить компьютер”;
2) сервер обрабатывает запрос;
3) с сервера на модуль windows-bot отправляется текстовая команда содержащая код действия и имя компьютера;
4) модуль windows-bot открывает командную строку и выполняет полученную команду.

У варианта использования имеется одна точка расширения: Перезагрузить группу компьютеров.

Вариант использования «Перезагрузить группу компьютеров». Вариант использования позволяет перезагрузить группу компьютеров.

Основной поток событий состоит из четырех шагов:

1) пользователь в веб-интерфейсе нажимает на кнопку “Перезагрузить группу компьютеров”;
2) сервер обрабатывает запрос;

3) с сервера на модуль windows-bot отправляется текстовая команда содержащая код действия и имена компьютеров;
4) модуль windows-bot открывает командную строку и выполняет полученную команду.

Вариант использования «Выключить компьютер». Вариант использования позволяет выключить выбранный компьютер.

Основной поток событий состоит из четырех шагов:

1) пользователь в веб-интерфейсе нажимает на кнопку “Выключить компьютер”;
2) сервер обрабатывает запрос;
3) с сервера на модуль windows-bot отправляется текстовая команда содержащая код действия и имя компьютера;
4) модуль windows-bot открывает командную строку и выполняет полученную команду.

У варианта использования имеется одна точка расширения: Выключить группу компьютеров.

Вариант использования «Выключить группу компьютеров». Вариант использования выключает группу компьютеров.

Основной поток событий состоит из четырех шагов:

1) пользователь в веб-интерфейсе нажимает на кнопку “Выключить группу компьютеров”;
2) сервер обрабатывает запрос;
3) с сервера на модуль windows-bot отправляется текстовая команда содержащая код действия и имена компьютеров;
4) модуль windows-bot открывает командную строку и выполняет полученную команду.

Вариант использования «Выбрать режим загрузки компьютеров». Вариант использования позволяет выбрать режим загрузки группы выбранных компьютеров.

Основной поток событий состоит из четырех шагов:

1) пользователь в веб-интерфейсе нажимает на кнопку “Выбрать режим загрузки компьютеров”;
2) сервер обрабатывает запрос;
3) с сервера на модуль pxe-bot отправляется команда на смену режима загрузки компьютера;
4) модуль pxe-bot изменяет режим загрузки компьютеров.

Вариант использования «Обновить состояние всех компьютеров». Вариант использования позволяет обновить информацию об компьютерах в локальной сети.

Основной поток событий состоит из пяти шагов:

1) пользователь в веб-интерфейсе нажимает на кнопку “Обновить состояние всех ПК”;
2) сервер обрабатывает запрос;
3) с модуля class-controller на модуль windows-bot отправляется команда на обновление информацию об компьютерах в локальной сети;
4) модуль windows-bot обновляет ARP таблицу и отправляет MAC и IP адреса модулю class-controller;
5) модуль class-controller обрабатывает полученную информацию.

Вариант использования «Посмотреть карту расположения компьютеров». Вариант использования позволяет пользователям увидеть схему расстановки компьютеров в аудитории.

Основной поток событий состоит из одного шага в котором пользователь входит на страницу аудитории.

Администратор представляет собой человека, взаимодействующего с системой управления компьютерными. Администратору доступны все варианты использования актера «Пользователь», а также имеется дополнительный вариант использования:

Вариант использования «Администрирование пользователей». Вариант использования позволяет выполнять добавление, удаление, редактирование учетных записей пользователей.

Основной поток событий состоит из двух шагов:

1) через веб-интерфейс создается учетная запись;
2) для созданной учетной записи назначаются права.

## 3.2. ДИАГРАММА КОМПОНЕНТОВ МОДУЛЕЙ СИСТЕМЫ

Диаграмма компонентов модулей системы, описывает дополнительные модули необходимые для выполнения функциональных требований (рисунок 9):

- модуль управления режимом загрузки компьютеров (pxe-bot);
- модуль управления компьютерами (windows-bot);
- главный модуль управления (class-controller).

Главный модуль управления class-controller содержит в себе три компонента с помощью которых выполняются команды от пользователя. Компонент обновления состояний компьютеров включает в себя проверку доступности, проверку типа операционной системы и проверку IP адреса компьютеров. Компонент изменения режима загрузки компьютеров, позволяет компьютеру загружаться с образа скачанного с PXE сервера. Компонент выполнения действий с компьютерами, выполняет включение, выключение и перезагрузку компьютеров. Все вышеописанные компоненты для выполнения поставленных команд обращаются к модулям pxe-bot и windows-bot.

Модуль управления режимом загрузки компьютеров pxe-bot, необходим для работы PXE сервера, а также для выполнения выключения и перезагрузки компьютеров с запущенными Linux образами.

Модуль управления компьютерами windows-bot, предназначен для выполнения команд на включение, выключение или перезагрузку компьютеров

на базе операционной системы Windows. Также в него включены функции определения IP и типа операционной системы компьютеров в локальной сети.

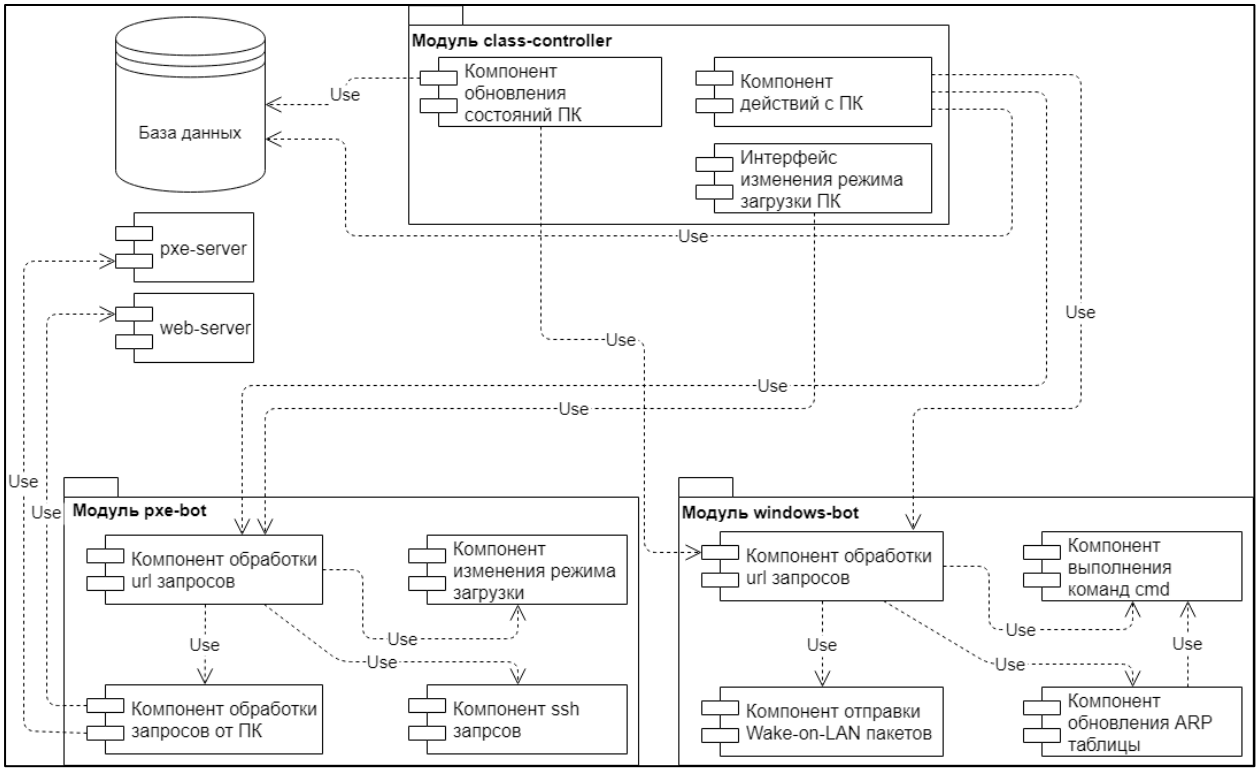


Рисунок 9 – Диаграмма компонентов модулей системы

## 3.3. ДИАГРАММЫ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Разработанные диаграммы описывают взаимодействие между внутренними модулями Django и дополнительно разработанными модулями.

### 3.3.1. Вход на Главную страницу

Диаграмма последовательностей «Вход на Главную страницу» описывает взаимодействие модулей Django при входе пользователя на главную страницу веб-приложения. Диаграмма последовательности для модулей Django изображена на рисунке 10.

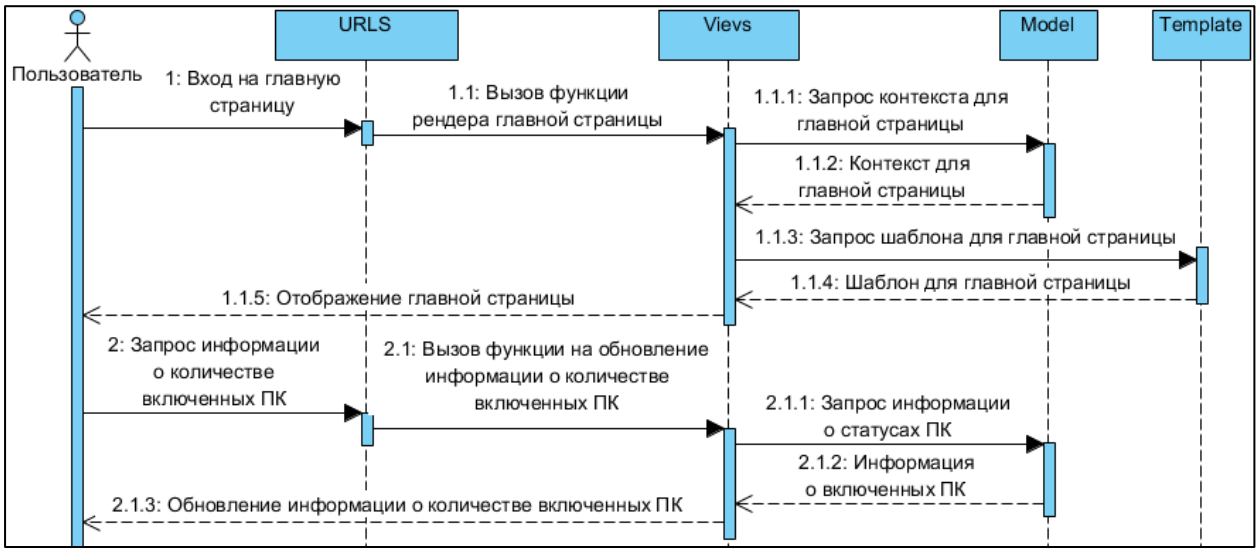


Рисунок 10 – Диаграмма последовательностей «Вход на Главную страницу»

### 3.3.2. Вход на страницу аудиторий

Диаграмма последовательностей «Вход на страницу аудиторий» описывает взаимодействие модулей Django при входе пользователя на страницы аудиторий веб-приложения. Диаграмма последовательности для модулей Django изображена на рисунке 11.

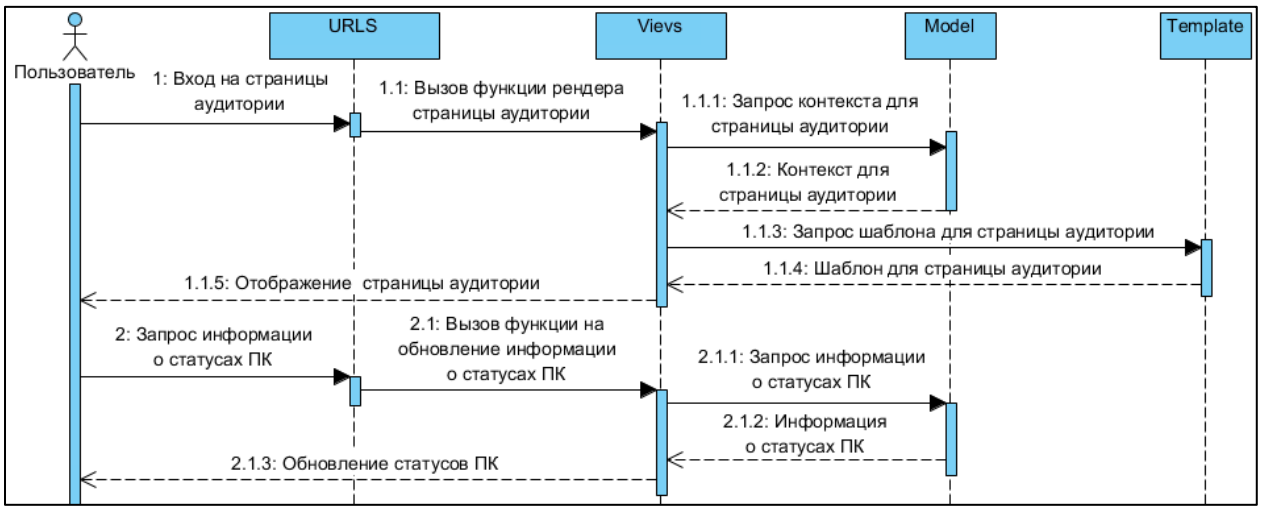


Рисунок 11 – Диаграмма последовательностей «Вход на страницу аудиторий»

### 3.3.3. Обновление статусов ПК

Диаграмма последовательностей «Обновление статусов ПК» описывает взаимодействие модулей Django при ручном обновлении статусов ПК. Диаграмма последовательности для модулей Django изображена на рисунке 12.

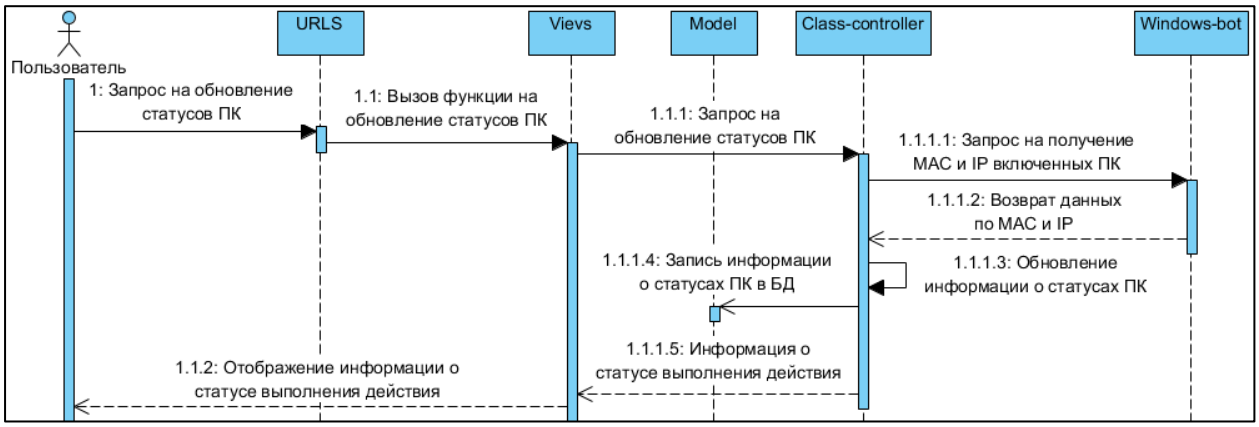


Рисунок 12 – Диаграмма последовательностей «Обновление статусов ПК»

На рисунке 13 изображена диаграмма последовательности для модуля class-controller.

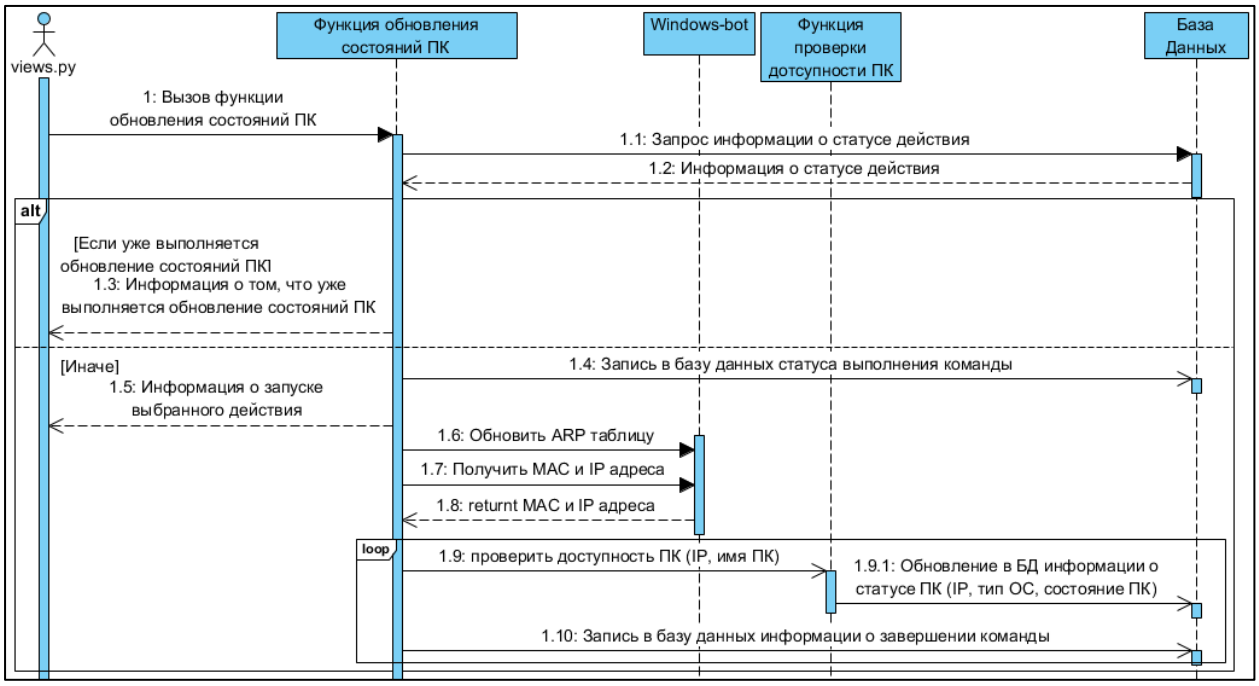


Рисунок 13 – Диаграмма последовательности для модуля class-controller

### 3.3.4. Включение, выключение, перезагрузка

Диаграмма последовательностей «Включение, выключение, перезагрузка» описывает взаимодействие модулей Django при выполнении включения, выключения, перезагрузки компьютеров. Диаграмма последовательности для модулей Django изображена на рисунке 14.

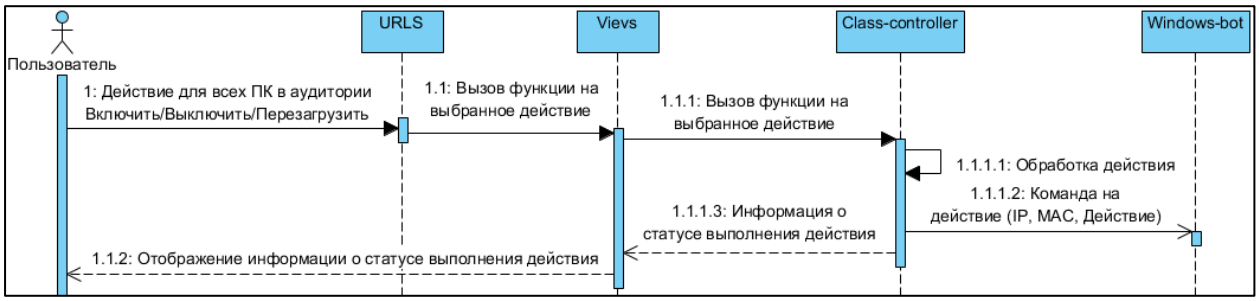


Рисунок 14 –Диаграмма последовательностей «Включение, выключение, перезагрузка»

На рисунке 15 изображена диаграмма последовательности для модуля class-controller.

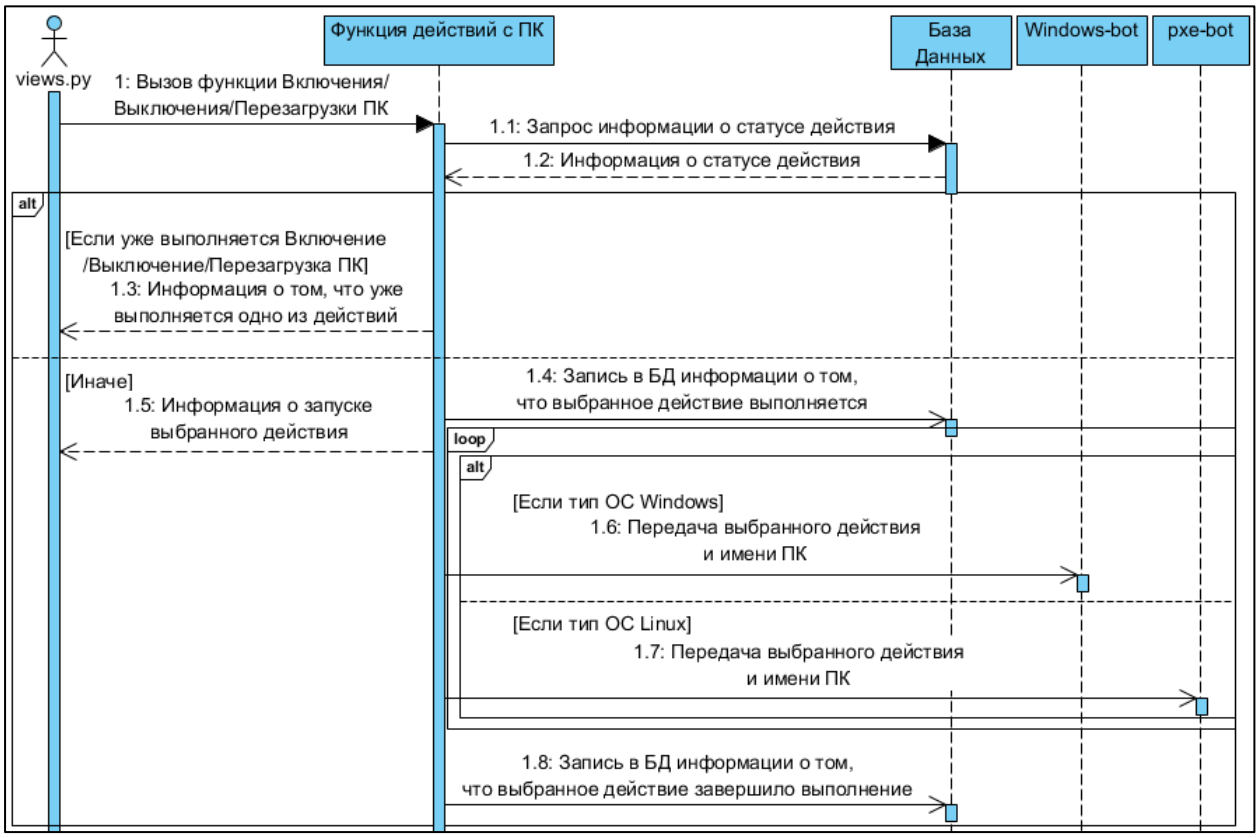


Рисунок 15 – Диаграмма последовательности для модуля class-controller

### 3.3.5. Выбор режима загрузки

Диаграмма последовательностей «Выбор режима загрузки» описывает взаимодействие модулей Django при смене режима загрузки компьютера.

Диаграмма последовательности для модулей Django изображена на рисунке 16.

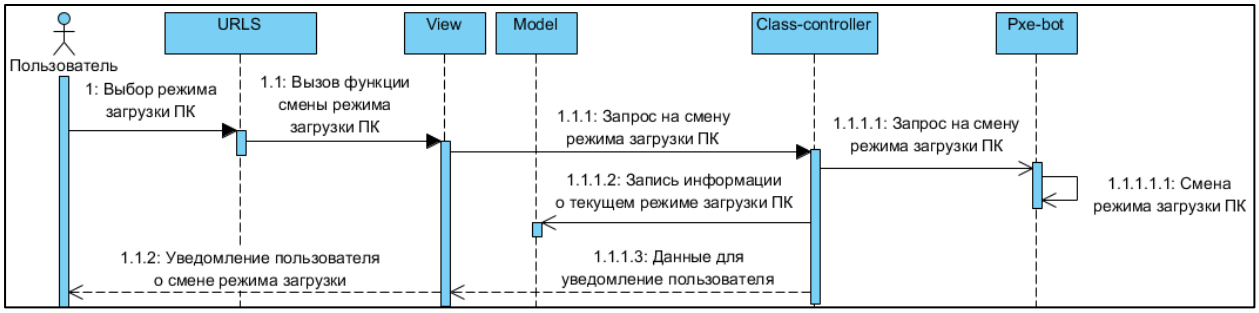


Рисунок 16 – Диаграмма последовательностей «Выбор режима загрузки»

### 3.3.6. Модуль pxe-bot

Диаграмма последовательности для модуля pxe-bot изображена на рисунке 17.

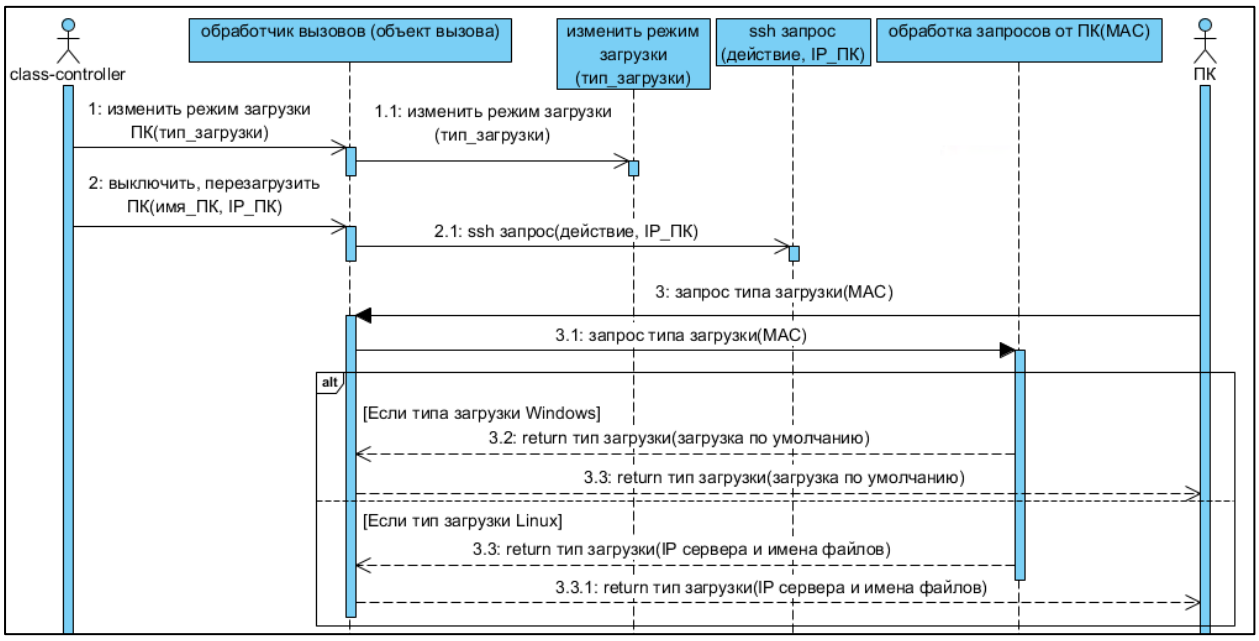


Рисунок 17– Модуль управления режимом загрузки (pxe-bot)

### 3.3.7. Модуль windows-bot

Диаграмма последовательности для модуля windows-bot изображена на рисунке 18.

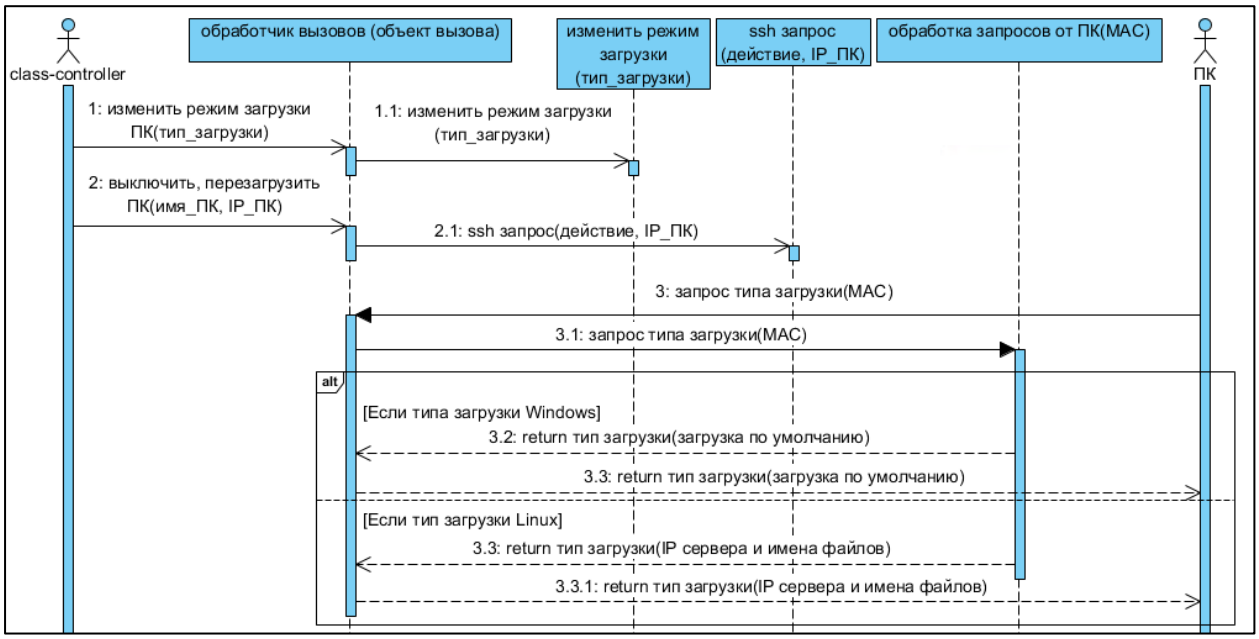


Рисунок 18 – Модуль управления компьютерами (windows-bot)

## 3.4. ДИАГРАММА БАЗЫ ДАННЫХ

На рисунке 19 представлена схема базы данных, составленная на основе разработанных диаграмм для системы.

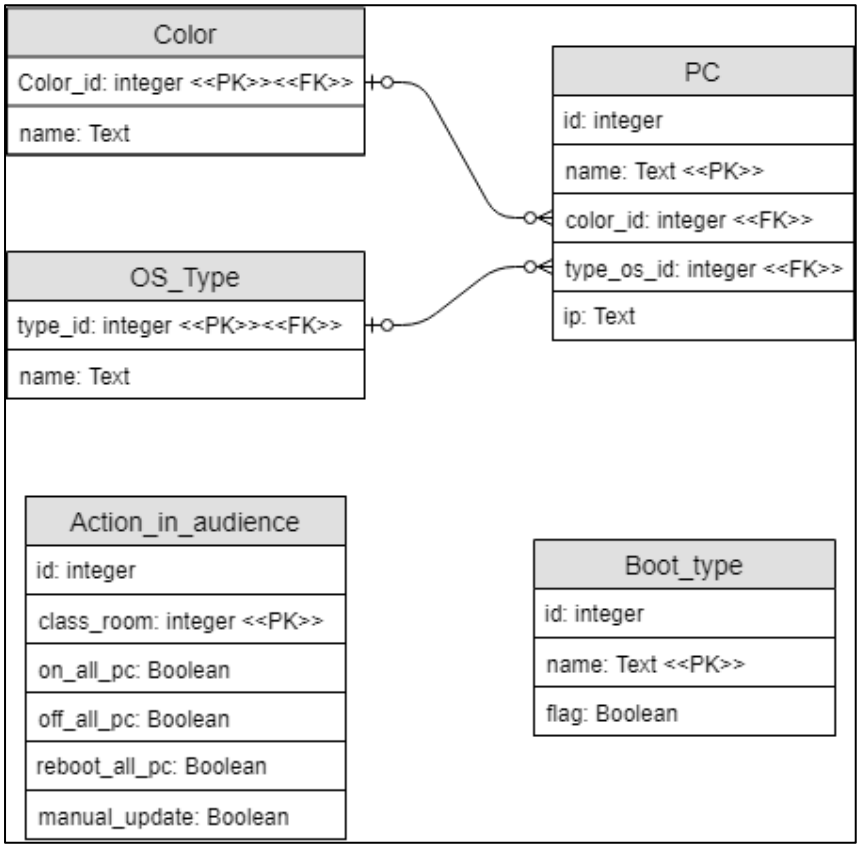


Рисунок 19 – Схема базы данных

Таблица *PC* содержит информацию о компьютерах. Каждый кортеж состоит из следующих атрибутов:

– *id* содержит уникальный идентификатор компьютера;

– *name* содержит название компьютера и является первичным ключом;

– *color_id* содержит уникальный идентификатор цвета, который означает включен ли компьютер или нет, и является внешним ключом на один из кортежей в таблице *Color;*

– *type_os_id* содержит уникальный идентификатор типа операционной системы, которая используется в данный момент, и является внешним ключом на один из кортежей в таблице *OS_type;*

– *ip* содержит IP адрес компьютера.

Таблица Action_in_audience содержит информацию о текущих действиях с всеми компьютерами в аудиториях. Каждый кортеж состоит из следующих атрибутов:

– *id* содержит уникальный идентификатор аудитории;

– *class_room* содержит номер аудитории и является первичным ключом;

– *on_all_pc* содержит логическую переменную указывающую на текущий статус команды;

– *off_all_pc* содержит логическую переменную указывающую на текущий статус команды;

– *reboot_all_pc* содержит логическую переменную указывающую на текущий статус команды;

– *manual_update* содержит логическую переменную указывающую на текущий статус команды.

Таблица *boot_type* содержит в себе информацию, указывающий на текущий тип загрузки компьютеров. Каждый кортеж состоит из следующих атрибутов:

– *id* содержит уникальный идентификатор режима загрузки;

– *flag* содержит логическую переменную указывающую на текущий статус режима загрузки;

– *name* содержит название типа загрузки.

Таблица *color* содержит в себе информацию о цвете указывающего на текущий статус компьютера. Каждый кортеж состоит из следующих атрибутов:

– *id* содержит уникальный идентификатор цвета и является внешним ключом на один из кортежей в таблице *PC*. Также, является частью составного первичного ключа;

– *name* содержит в себе название цвета.

Таблица *OS_type* содержит в себе информацию о типе операционной системы на запущенном компьютере. Каждый кортеж состоит из следующих атрибутов:

– *id* содержит уникальный идентификатор цвета и является внешним ключом на один из кортежей в таблице *PC*. Также, является частью составного первичного ключа;

– *name* содержит в себе название операционной системы.

# 4. РЕАЛИЗАЦИЯ

В данном разделе приведены основные алгоритмы, использующиеся в разработанном веб приложении. Основной код разработанного веб-приложения приведен на листинге А.1 приложения А.

## 4.1. ОСНОВНЫЕ АЛГОРИТМЫ

### 4.1.1. Алгоритм включения, выключения, перезагрузки ПК

*Текстовое описание алгоритма*

Данный алгоритм описывает логику работы программы при включении, выключении или перезагрузки всех ПК в выбранной аудитории.

ШАГ 1. Пользователь в web-браузере нажимает на кнопку включения или выключения, или перезагрузки всех ПК.

ШАГ 2. Web-браузер отправляет get запрос на сервер который содержит информацию об аудитории и выбранном действии. Пример таких запросов представлен на рисунке 20.

```
GET class-controller/110/change_status_pc/on/true
GET class-controller/110/change_status_pc/off/true
GET class-controller/110/change_status_pc/reboot/true
```

Рисунок 20 – GET запрос

Где *class-controller* – это название приложения, *110* – номер аудитории, *change_status_pc* – функция, отвечающая за выполнение действий с компьютерами, *on, off, reboot* – команды на включение, выключение и перезагрузку всех компьютеров в 110 аудитории.

ШАГ 3. URL маршрутизатор при получении HTTP запроса от пользователя выполняет проверку на совпадение с URL-шаблоном. При первом совпадении происходит вызов функции в модуле представления, отвечающей за выполнение действий с компьютерами. Пример URL шаблонов в формате python представлен на рисунке 21.

```
urlpatterns = [
    path('accounts/', include('django.contrib.auth.urls')),

    path('', views.home, name='home'),
    path('home/', views.home, name='home'),
    path('110/', views.aud_110, name='110'),
    path('112/', views.aud_112, name='112'),

    path('home/home_context_110/', views.home_context_110),
    path('home/home_context_112/', views.home_context_112),

    re_path(r'(?P<classroom>[\d]{3})/get_context/(?P<name_pc>([\w]{2}[\d]{5})|([\D]{2}))', views.get_context),
    re_path(r'(?P<classroom>[\d]{3})/change_status_pc/(?P<action>[\w]{2,6})/(?P<pcname>[\w]{2}[\d]{5})', views.change_status_pc),
    re_path(r'(?P<classroom>[\d]{3})/change_status_pc/(?P<action>[\w]{2,6})/(?P<in_aud_all>[\w]{4})', views.change_status_pc),
    re_path(r'home/change_status_pc/(?P<action>[\w]{2,6})/(?P<all_aud>[\w]{4})', views.change_status_pc),

    path('change_boot_load/', views.change_boot_load),
    path('manual_update_states/', views.manual_update_states),
]
```

Рисунок 21 – URL шаблоны приложения class-controller

ШАГ 4. Вызванная функция проверяет тип запроса. Далее из get запроса определяется выбранное действие с компьютерами. После чего происходит вызов соответствующей выбранному действию функции в главном модуле управления. Пример функции определения действия представлен в виде псевдокода на рисунке 22.

```
def изменить_статус_компьютера(запрос):
    если метод GET:
        если запрос содержит команду на действие:
            вызвать функцию, выполняющую действия в модуле class-controller
```

Рисунок 22 – Псевдокод функции определения действия

ШАГ 5. В главном модуле управления (class-controller) происходит проверка выполнения данного действия другим пользователем. Если действие уже выполняется, то пользователь будет проинформирован уведомлением об этой ситуации (рисунок 23), а продолжение алгоритма включения, выключения, перезагрузки всех ПК остановится. Иначе выполняется ШАГ 6.

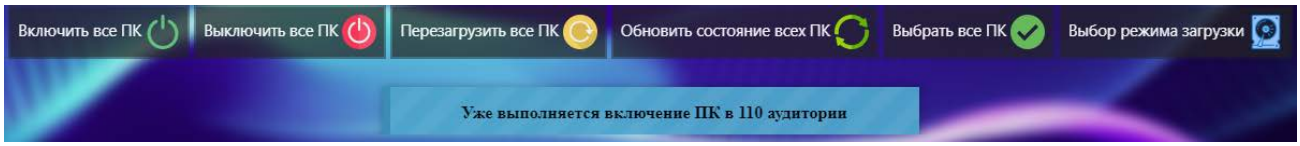


Рисунок 23 – Уведомление пользователя

ШАГ 6. Вызывается функция, выполняющая действия со всеми компьютерами, в параметрах которой указывается выбранное пользователем действие и номер аудитории. Она выполняет проверку компьютеров в выбранной аудитории, а именно доступность каждого из них и тип используемой операционной системы. На основании проверки функция подготавливает

итоговый список компьютеров, с которыми можно провести выбранное действие. После чего в многопоточном режиме вызывается функция, выполняющая действия с одним компьютером, в параметрах которой указывается имя компьютера и выбранное действие.

ШАГ 7. Вызванная функция вызывает модуль управления компьютерами (windows-bot) в параметрах которой указывается имя компьютера и выбранное действие.

ШАГ 8. Модуль управления компьютерами (windows-bot) на вход получает информацию об выбранном действии, компьютере и его IP адресе, после чего от имени администратора домена, модуль выполняет присланное ему действие.

#### 4.1.2. Алгоритм смены режима загрузки ПК

*Текстовое описание алгоритма*

Данный алгоритм повторяет первые 2 шага представленные в алгоритме раздела 4.2.1.

ШАГ 3. URL маршрутизатор вызывает функцию в модуле views, которая отвечает за смену режима загрузки компьютеров.

ШАГ 4. Вызванная функция проверяет тип запроса. Далее из get запроса определяется выбранный режим загрузки компьютеров.

ШАГ 5. Вызывается главный модуль управления компьютерами (class-controller) из которого происходит вызов модуля управления режимом загрузки компьютеров (pxe-bot), а также происходит запись в базу данных о смене режима загрузки.

#### 4.1.3. Алгоритм обновления состояний ПК

*Текстовое описание алгоритма*

Данный алгоритм повторяет первые 2 шага представленные в алгоритме раздела 4.2.1.

ШАГ 3. URL маршрутизатор вызывает функцию в модуле views, которая отвечает за ручное обновление статусов компьютеров.

ШАГ 4. Вызванная функция проверяет тип запроса. Далее из get запроса определяется команда на обновление состояний компьютеров.

ШАГ 5. Вызывается главный модуль управления (class-controller) который запускает четыре основных функций:

1. ping_thread_arp() – данная функция вызывает модуль управления компьютерами (windows-bot) и выполняет пинг диапазона IP адресов для обновления arp таблицы.
2. get_ip_t() – функция обновляет IP адреса содержащиеся в базе данных, которые затем используются для показа пользователям в веб интерфейсе и определения доступности компьютера.
3. get_macs() – функция запрашивает у модуля управления компьютерами (windows-bot) информацию об MAC адресах и относящихся к ним IP адресах.
4. load_color_state() – функции обновляют состояния компьютеров содержащиеся в базе данных, которые затем используются для показа пользователям в веб интерфейсе.

## 4.2. КОНФИГУРАЦИОННЫЕ ФАЙЛЫ

В разработанной системе имеется два типа конфигурационных файлов:

- cfg_<номер аудитории>_location.cfg – файл с информацией о расположении компьютеров в аудитории, их именами и необходимым контекстом для таблицы в веб-интерфейсе (см. рисунок 24).
- classes.cfg – файл с информацией о MAC адресах компьютеров в аудиториях (см. рисунок 25).

```
{"namepc": '21', "aud": '110', "number": '21', "namepct": 'PC11021'},
{"namepc": '02', "aud": '110', "number": '22', "namepct": 'PC11002'},
{"namepc": '--', "aud": '110', "number": '23', "namepct": '--'},
{"namepc": '--', "aud": '110', "number": '24', "namepct": '--'},
```


Рисунок 24 – Расположения ПК в аудитории


```
"PC11013": {"mac": "E0.69.95.D7.4F.C4"},
"PC11007": {"mac": "E0.69.95.D7.4F.89"},
"PC11004": {"mac": "E0.69.95.D7.4F.AE"},
"PC11011": {"mac": "E0.69.95.D7.4F.CE"},
"PC11001": {"mac": "E0.69.95.D7.4F.8E"},
```


Рисунок 25 – MAC адреса компьютеров

## 4.3. WEB ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ

Рендер web страниц в Django происходит при помощи системы шаблонов, которая указывает структуру выходного HTML [14, 20] документа. Правильно разработанные шаблоны позволяют уменьшить количество написанной похожей или одинаковой гипертекстовой разметки, а также улучшают читабельность документа. В данном проекте использовался шаблонизатор под названием Jinja2 - это популярная библиотека Python [13, 16, 19]. С помощью данной библиотеки были разработаны шаблоны для трех web страниц, а именно “Главная”, “Аудитория 110” и “Аудитория 112”. Все разработанные шаблоны располагаются в папке “templates” (рисунок 26).

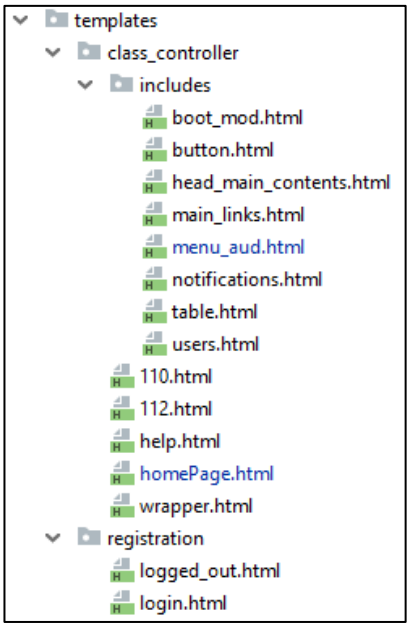


Рисунок 26 – HTML документы разработанной системы

Это общая папка для всех HTML документов в проекте. Она обладает следующей структурой:

- class_controller – название разрабатываемого приложения. Данное повторение названия необходимо чтобы избежать конфликтов с другими приложениями в данном проекте;
- includes – html документы которые используются для внедрения в другие шаблоны;
- registration – html документы для страницы идентификации.

wrapper.html – содержит общую для всех страниц разметку, а именно:

- мета заголовки (см. рисунок 28);
- ссылки на статические файлы (см. рисунок 29);
- блок навигации по сайту (рисунок 27);
- блок входа для пользователей (см. рисунок 30).

Разметка документа *wrapper* в формате html представлена на рисунке 27.

```
{% load static %}
<!DOCTYPE html>
<html lang="ru">
<head>
    {% include "lsm/includes/head_main_contents.html"%}
    {% include "lsm/includes/main_links.html"%}
</head>
<body>
<div class="main_menu_d_l">
        <div class="main_logo">
            <img class="logo_img" src="{% static "Images/logo.png" %}" alt="My image">
        </div>
        <div id="menu_main">
                <ul>
                    <li><a href="{% url "lsm:home" %}">Главная</a></li>
                    <li><a href="{% url 'lsm:110' %}">Аудитория 110</a></li>
                    <li><a href="{% url 'lsm:112' %}">Аудитория 112</a></li>
                </ul>
        </div>
        {% include "lsm/includes/users.html"%}
</div>

{% block content %}

{% endblock %}

</body>
</html>
```

Рисунок 27 – Разметка документа wrapper.html

```
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>LSM</title>
```

Рисунок 28 – Мета заголовки (head_main_contents.html)

```
{% load static %}
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/normalize/5.0.0/normalize.min.css">
<link rel='stylesheet prefetch' href='https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css'>
<script type="text/javascript" src="{% static 'js/jquery-3.3.1.js' %}"> </script>
<link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}" type="text/css">
<link rel="stylesheet" href="{% static 'css/main.css' %}" type="text/css">
<script type="text/javascript" src="{% static 'js/main.js' %}" defer> </script>
<link rel="shortcut icon" type="image/png" href="{% static "Images/favicon.ico" %}">
```

Рисунок 29 – Ссылки на статические файлы (main_links.html)

```
<div style="float: right; width: 8%; height: 80px; position: relative; top: 1em;">
    <div style="position: relative; float: left; display: inline-block;">
        <li style="list-style-type: none;"><a href="/admin/">Панель администратора</a> </li>
    </div>
    <div style="position: relative; float: left; display: inline-block;">
        {% if user.is_authenticated %}
          <li style="list-style-type: none;"><button id="role" name="{{ role }}" style="display: none;"></button>User: {{
          user.get_username }}</li>
          <li style="list-style-type: none;"><a href="{% url 'lsm:logout'%}?next={{request.path}}">Выйти</a></li>
        {% else %}
          <li style="list-style-type: none;"><a href="{% url 'lsm:login'%}?next={{request.path}}">Войти</a></li>
        {% endif %}
    </div>
</div>
```

Рисунок 30 – Блок входа для пользователей (users.html)

Файл boot_mod.html (рисунок 31) содержит блок, отображающий текущий режим загрузки компьютеров, а также функцию для его смены.

```
{% load static %}
<div class="menu_2 menu_2_blue boot_mode_d3 menu_button_32x32_div">
        <div class="boot_mode_d1">
            <button class="menu_2_addition ">Выбор режима загрузки
                <img class="menu_button_32x32_img" src="{% static 'Images/os_type.png' %}">
            </button>
        </div>
        <div class="boot_mode_d2" >
            <div id="win7" class="div_b_L_3_on_off boot_mode_d4" style="background: {{ boot_mod_win7 }}">
                <button class="button_action boot_mode_button" onclick="update_boot_mod(this)" value="" name="windows7" type="submit"
                title="">Win7</button>
            </div>
            <div id="kiosk" class="div_b_L_3_on_off boot_mode_d4" style="background: {{ boot_mod_kiosk }}">
                <button class="button_action boot_mode_button" onclick="update_boot_mod(this)" value="" name="kiosk" type="submit"
                title="">Linux</button>
            </div>
        </div>
</div>
```

Рисунок 31 – Состав файла boot_mod.html

Файл button.html (см. рисунок 32) содержит разметку для ячеек таблиц аудиторий 110 и 112 содержащих функции управления компьютерами.

```
{% load static %}
{% for content in contents %}
    <td class="border_cell_pc">
        <div id="div_include_{{ content.number }}" class="div_b_L_1">
            <div class="div_b_L_2">
                <div class="div_b_L_3_on_off" >
                    <button class="button_action" onclick="localFunction(this); hover_pc_cell('div_include_{{
                    content.number }}', 'div_main_{{ content.number }}')" value="{{ content.namepct }}"
                    name="on" type="submit">ВКЛ</button>
                </div>
                <div class="div_b_L_3_on_off" style="width: 50%;">
                    <button class="button_action" onclick="localFunction(this); hover_pc_cell('div_include_{{
                    content.number }}', 'div_main_{{ content.number }}')" value="{{ content.namepct }}"
                    name="reboot" type="submit">Перезагрузка</button>
                </div>
                <div class="div_b_L_3_on_off">
                    <button class="button_action" onclick="localFunction(this); hover_pc_cell('div_include_{{
                    content.number }}', 'div_main_{{ content.number }}')" value="{{ content.namepct }}"
                    name="off" type="submit">ВЫКЛ</button>
                </div>
            </div>
            <div class="div_b_L_3_on_off div_b_L_3_on_off_bottom">
                <button class="button_action" style="font-size:100%;" value="" name="" type="submit"
                title="">   </button>
            </div>
        </div>
        <div id="div_main_{{ content.number }}" class="button_cell_pc" >
            <div class="div_m_L_2_chekbox">
                {% if content.namepct != '--' %}
                    <input id="chek_{{ content.number }}" onclick="getChekbox(this)" name="{{ content.namepct
                    }}" value="1" type="checkbox" class="checkbox_pc" >
                {% endif %}
            </div>
            <div class="div_m_L_2">
                <button id="button_pc_color_{{ content.number }}" class="border_cell_pc_t"
                onclick="hover_pc_cell('div_main_{{ content.number }}', 'div_include_{{ content.number }}',
                this)" value="{{ content.namepct }}"  name="button_m_{{ content.number }}" type="submit"
                title="Показать опции" >{{ content.namepc}}</button>
            </div>
            <div class="div_os_type_1">
                    <div id="os_type_{{ content.number }}" pc_name="{{ content.namepct }}">
                        {% if content.namepct != '--' %}
                        <div id="div_os_type_win" class="div_os_type_2" style="display: none;">
                            <img class="menu_button_32x32_img" src="{% static 'Images/windows.png' %}">
                        </div>
                        <div id="div_os_type_linux" class="div_os_type_2" style="display: none;">
                            <img class="menu_button_32x32_img" src="{% static 'Images/linux.png' %}">
                        </div>
                        {% endif %}
                    </div>
            </div>
            {% if role == 'admin' or role == 'superadmin' %}
                <div id="div_ip_{{ content.number }}" class="div_ip" pc_name="{{ content.namepct }}"></div>
            {% endif %}
        </div>
    </td>
{% endfor %}
```

Рисунок 32 – Состав файла button.html

Файл menu_aud.html (см. рисунок 33) содержит разметку для меню управления всеми компьютерами в выбранной аудитории.

```
{% load static %}
<div class="menu_2_div_1">
    <div class="menu_2 menu_2_blue menu_button_32x32_div">
        <button id="on_b" class="menu_2_addition" onclick="{{ onclick_function }}" name="{{ name_button_on
        }}"     type="submit" value="{{ name_aud }}">Включить все ПК</button>
        <img class="menu_button_32x32_img" src="{% static 'Images/power-on.png' %}" >
    </div>
    <div class="menu_2 menu_2_blue menu_button_32x32_div">
        <button id="off_b" class="menu_2_addition" onclick="{{ onclick_function }}" name="{{ name_button_off
        }}"    type="submit" value="{{ name_aud }}">Выключить все ПК</button>
        <img class="menu_button_32x32_img" src="{% static 'Images/power-off.png' %}" >
    </div>
    <div class="menu_2 menu_2_blue menu_button_32x32_div">
        <button id="reboot_b" class="menu_2_addition" onclick="{{ onclick_function }}" name="{{
        name_button_reboot }}" type="submit" value="{{ name_aud }}">Перезагрузить все ПК</button>
        <img class="menu_button_32x32_img" src="{% static 'Images/reboot.png' %}" >
    </div>
    {% if role == 'admin' or role == 'superadmin' %}
        <div class="menu_2 menu_2_blue menu_button_32x32_div">
            <button id="manual_update" class="menu_2_addition" onclick="manual_update(this);"
            name="manual_update" type="submit"> Обновить состояние всех ПК</button>
            <img class="menu_button_32x32_img" src="{% static 'Images/refresh.png' %}" >
        </div>
    {% endif %}
    {% if name_aud != 'home' %}
        <div id="checkbox_all_cheked" class="menu_2 menu_2_blue menu_button_32x32_div" style="display:
        inline-block">
            <button class="menu_2_addition" onclick="checkbox_all_cheked();" type="submit"> Выбрать все ПК
                <img class="menu_button_32x32_img" src="{% static 'Images/checked.png' %}" >
            </button>
        </div>
        <div id="checkbox_all_uncheked_off" class="menu_2 menu_2_blue menu_button_32x32_div" style="display:
        none">
            <button class="menu_2_addition" onclick="checkbox_all_uncheked();" type="submit"> Снять выбор со
            всех ПК
                <img class="menu_button_32x32_img" src="{% static 'Images/unchecked.png' %}" >
            </button>
        </div>
    {% endif %}
    {% include "lsm/includes/boot_mod.html"%}
    {% include "lsm/includes/notifications.html"%}
</div>
```

Рисунок 33 – Состав файла menu_aud.html

Файл notifications.html (рисунок 34) содержит разметку для уведомлений пользователей о статусе выполнения выбранных команд.

```
{% load static %}
<div id="success" class="message" style="display: none;">
    <a id="close" title="Закрыть"  href="#"
    onClick="document.getElementById('success').setAttribute('style','display: none;');">&times;</a>
</div>
<div id="error" class="message" style="display: none;">
    <a id="close" title="Закрыть"  href="#"
    onClick="document.getElementById('error').setAttribute('style','display: none;');">&times;</a>
</div>
<div id="warning" class="message" style="display: none;">
    <a id="close" title="Закрыть"  href="#"
    onClick="document.getElementById('warning').setAttribute('style','display: none;');">&times;</a>
</div>
<div id="info" class="message" style="display: none;">
    <a id="close" title="Закрыть"  href="#"
    onClick="document.getElementById('info').setAttribute('style','display: none;');">&times;</a>
</div>
<div id="ajax_load" style="position: relative; margin: auto; width: auto; height: auto; display : none;">
   <img  style="width: 40px; height: 40px;" src="{% static 'Images/ajax-loader.gif' %}" ><div style="font-size:
   15px; color: white;"> Выполнение команды! Ожидайте...</div>
</div>
```

Рисунок 34 – Состав файла notifications.html

Также для правильного отображения и функционала документа используются ccs стили и js скрипты (см. рисунок 35).

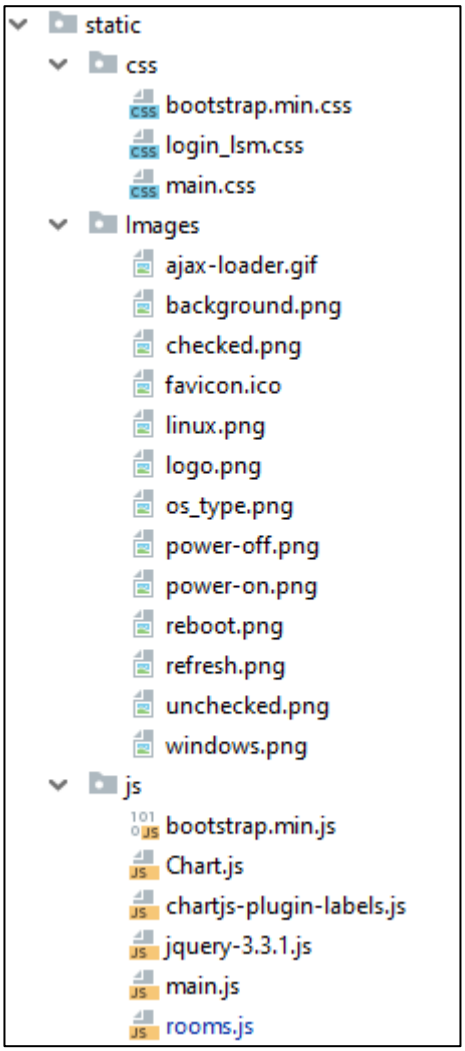


Рисунок 35 – Статичные файлы веб страниц

Папка *css* содержит следующие элементы.

*bootstrap.min.css* – включает в себя HTML и CSS-шаблоны оформления для веб-форм, меток, кнопок, блоков навигации и прочих компонентов веб-интерфейса.

*login_lsm.css* – включает в себя стили для оформления страницы входа пользователя.

*main.css* – включает в себя все основные стили, используемые для оформления страниц.

Папка *images* включает в себя картинки фона, логотипа и других значков.

Папка *js* содержит следующие элементы:

– bootstrap.min.js – набор готовых плагинов;

– Chart.js – инструмент, предназначенный для создания графиков и диаграмм;
– chartjs-plugin-labels.js – плагин для Chart.js позволяющий отображать информацию внутри круговых диаграмм;
– jquery-3.3.1.js – библиотека JavaScript, используемая для упрощенного взаимодействия между JavaScript и HTML;
– main.js – включает в себя скрипты для работы уведомлений и некоторого общего функционала;
– rooms.js – включает в себя скрипты для отправления запросов серверу содержащих команды для управления компьютерами.

Интерфейс главной страницы показан на рисунке 36. Данная страница включает в себя следующие файлы – homePage.html, wrapper.html, notifications.html, Chart.js, chartjs-plugin-labels.js, jquery-3.3.1.js, bootstrap.min.css, main.css, main.js.

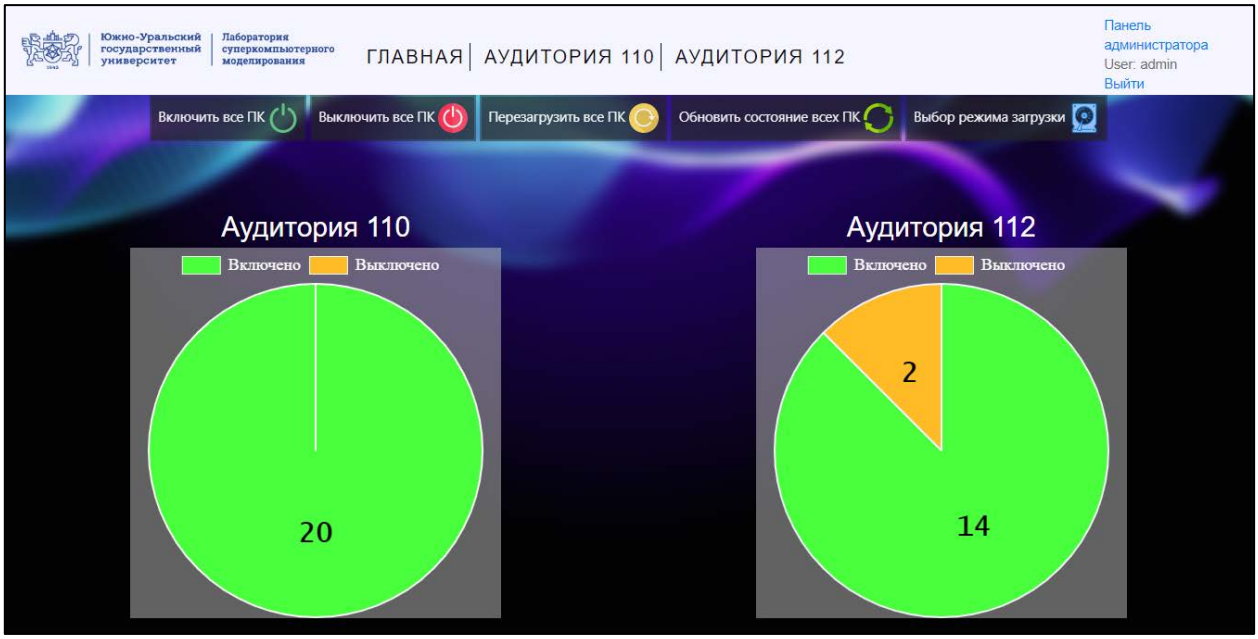


Рисунок 36 – Интерфейс главной страницы

Интерфейс страницы для 110 аудитории показан на рисунке 37. Данная страница включает в себя следующие файлы – 110.html, wrapper.html, notifications.html, rooms.js, jquery-3.3.1.js, bootstrap.min.css, main.css, main.js. Страница 112 аудитории включает в себя точно такой же набор файлов.

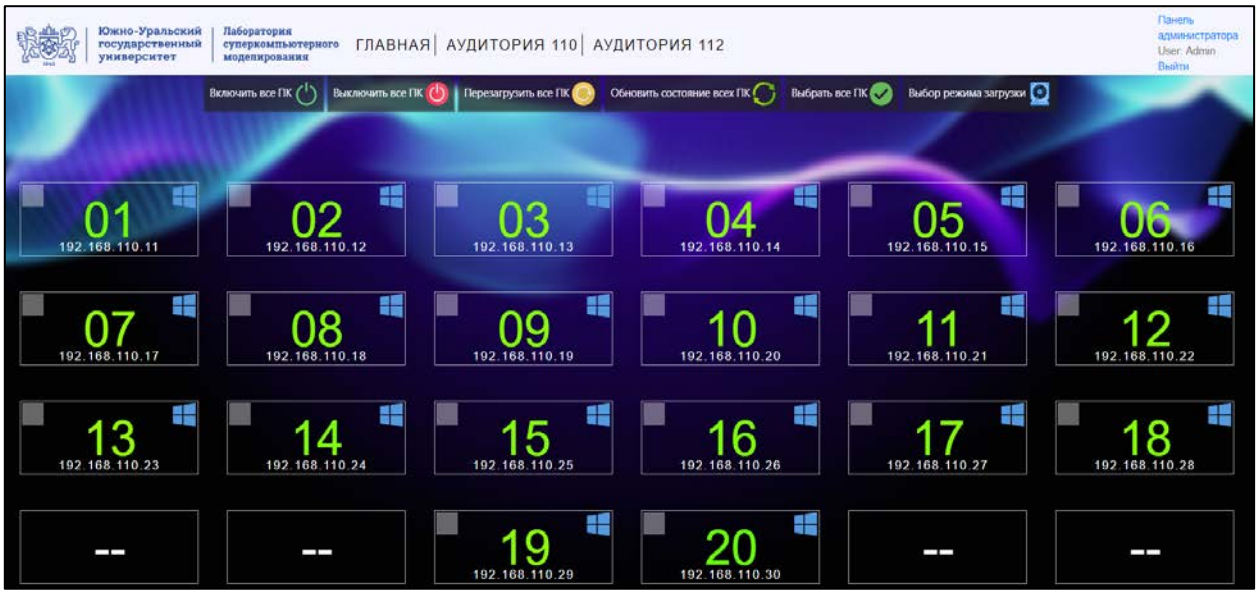


Рисунок 37 – Интерфейс страницы для 110 аудитории

Интерфейс страниц (рисунок 38) для аудиторий состоит из четырех главных элементов, а именно: главного меню, меню управления всеми компьютерами, меню пользователя и меню управления одним компьютером.

Главное меню позволяет перемещаться между тремя страницами сайта:

– “Главная”;
– “Аудитория 110”;
– “Аудитория 112”.

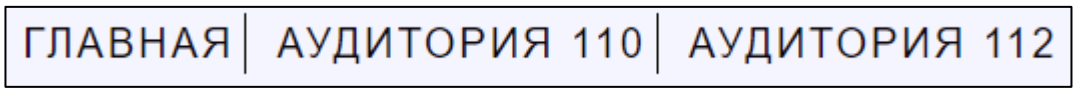


Рисунок 38 – Главное меню сайта

Меню управления всеми компьютерами (см. рисунок 39) позволяет выполнить одно из пяти основных действий:

– включить все компьютеры;
– выключить все компьютеры;
– перезагрузить все компьютеры;
– выбрать режим загрузки компьютера;
– выполнить ручное обновление состояний компьютеров.

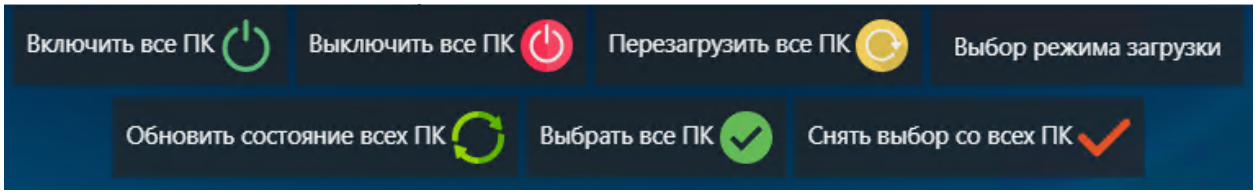


Рисунок 39 – Меню управления всеми компьютерами в аудитории

Данное меню отличается по функционалу для главной страницы так как позволяет выполнять действия для всех компьютеров двух аудиторий. Также меню может переключаться в групповой режим, который позволяет выполнять действия только для выбранных компьютеров (рисунок 40).

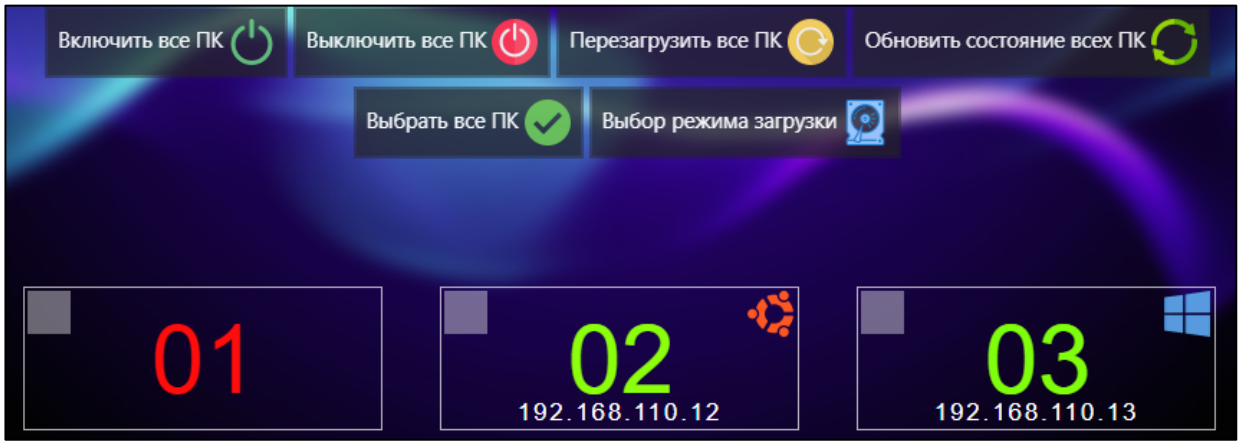


Рисунок 40 – Групповой режим работы меню управления компьютерами

Выбор компьютеров происходит при помощи, галочек, которые можно активировать в ячейках компьютеров (рисунок 41).

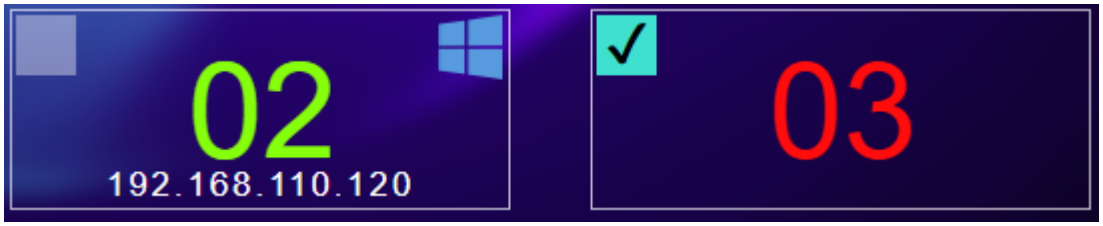


Рисунок 41 – Ячейка компьютера

Ячейка компьютера имеет четыре основных элемента:

- галочка для работы в групповом режиме;
- номер компьютера;
- IP адрес компьютера;
- тип операционной системы.

Номер компьютера имеет два состояния (см. рисунок 41). Красный цвет означает что компьютер выключен, зеленый, что включен. IP также имеет два состояния. У выключенного компьютера он отсутствует, поэтому в ячейке на месте блока с IP будет пусто, у включённого будет написан соответствующий IP адрес. Также в каждой ячейке справа находится значок показывающий тип запущенной операционной системы, Windows или Linux.

Каждая ячейка компьютера имеет внутренний функционал, который появляется если нажать на нее (рисунок 42). Свободное место в ячейках, в будущем позволит нарастить дополнительные функции.

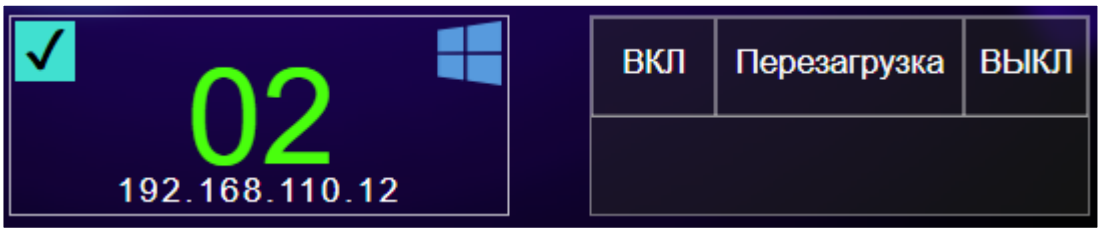


Рисунок 42 – Внутренний функционал ячейки компьютера

Кнопка ручного обновления состояний компьютеров (рисунок 43) позволяет обновить информацию о статусах ПК их IP адресе и типе ОС.

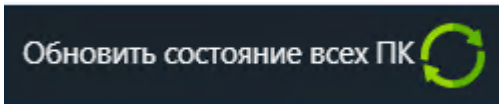


Рисунок 43 – Ручное обновление состояний компьютеров

# 5. РАЗВЕРТЫВАНИЕ И ТЕСТИРОВАНИЕ

Развертывание и тестирование разработанной системы проводилось на двух серверах. Для модуля windows-bot была развернута виртуальная машина в гипервизоре Hyper-V. В качестве операционной системы была выбрана Windows 10 LTSC x64 сборка номер 1809. Данная ОС отличается более урезанным функционалом, который не требуется для работы модуля windows-bot, что в свою очередь позволит уменьшить количество используемых аппаратных ресурсов.

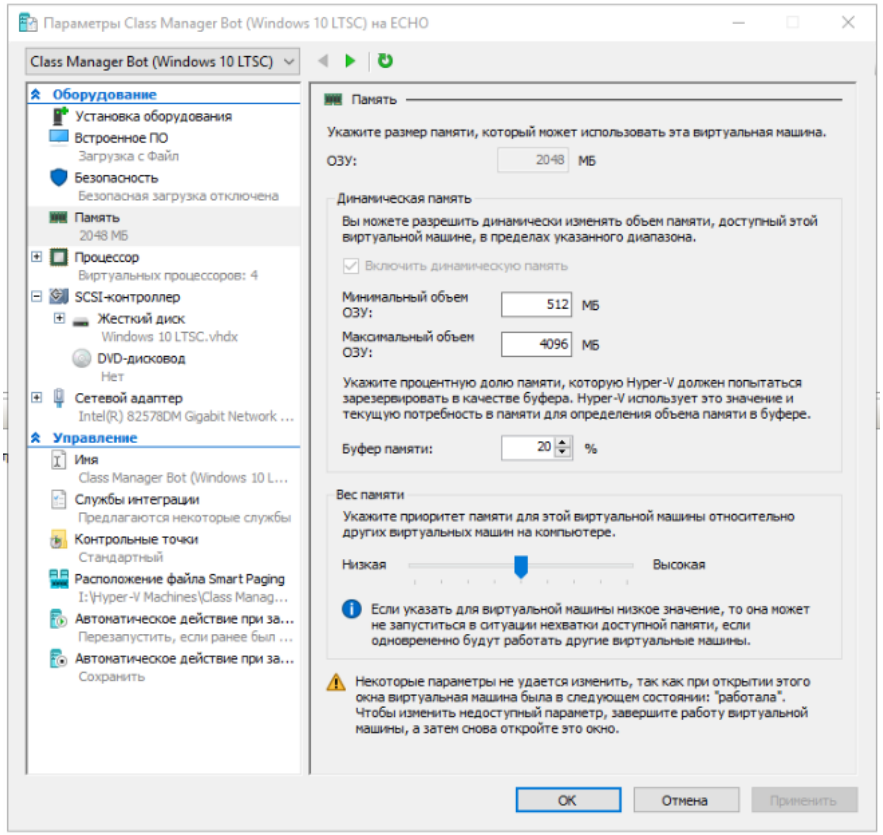


Рисунок 44 – Настройки виртуальной машины для модуля windows-bot

Для виртуальной машины были выбраны следующие настройки оборудования (рисунок 44):

- минимальный объём оперативной памяти 512 мегабайт;
- максимальный объем оперативной памяти 4096 мегабайт;
- размер жесткого диска 100 гигабайт;

– виртуальных процессоров 4.

Память HDD и RAM выделяется динамически, что также позволяет меньше затрачивать ресурсы сервера. После того как была установлена операционная система, была произведена установка Python версии 3.7 с двумя необходимыми для работы модуля библиотеками cherrypy и wakeonlan. Запуск модуля производился через командную строку Windows (рисунок 45).

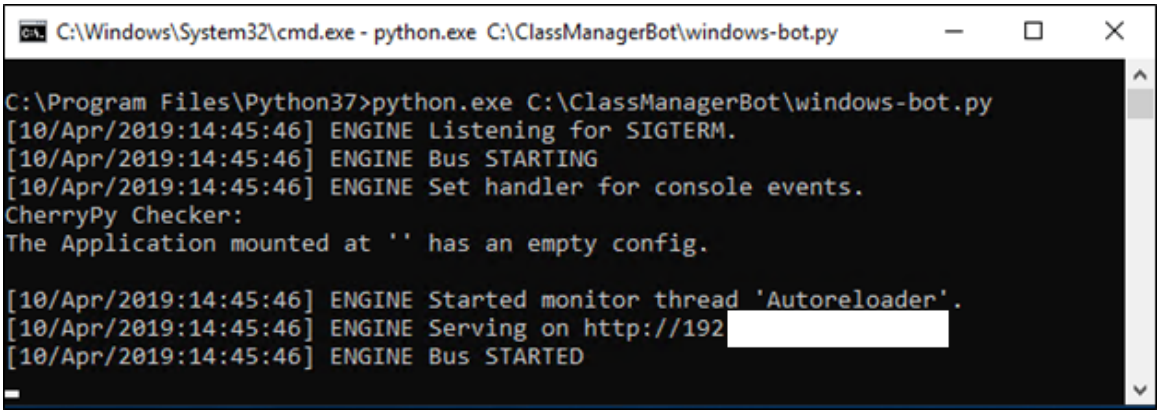


Рисунок 45 – Скриншот запущенного модуля windows-bot

Для модулей Django и pxe-bot был развернут сервер в гипервизоре Hyper-V на операционной системе Ubuntu Server x64 [21]. Настройки этой виртуальной машины повторяют настройки виртуальной машины для модуля windows-bot, за тем лишь исключением что виртуальных процессоров было выбрано 8.

Тестирование приложения производились следующими методами:

– функциональное тестирование;
– тестирование интерфейса пользователя.

Функциональное тестирование – это тестирование программного обеспечения в целях проверки реализуемости функциональных требований, то есть способности программного обеспечения в определенных условиях решать задачи, нужные пользователям [3]. Результаты функционального тестирование совпали с ожидаемыми. Примеры результатов тестирования приведены в таблице 2.

Таблица 2 – Результаты функционального тестирования.

| № | Название теста | Шаги | Ожидаемый результат | Тест пройден? |
|---|---|---|---|---|
| 1 | Проверка авторизации пользователя. | Неавторизованный пользователь входит на любые страницы веб-приложения. | Веб-приложение перенаправляет пользователя на страницу авторизации. | Да |
| 2 | Успешная авторизация. | Неавторизованный пользователь вводит корректный логин и пароль от своей учетной записи. | Веб-приложение перенаправляет пользователя на главную страницу сайта. | Да |
| 3 | Вход на страницы сайта. | Авторизованный пользователь входит на каждую страницу веб-приложения. | Пользователь получает доступ к запрашиваемой странице. | Да |
| 4 | Проверка работоспособности диаграмм на главной странице. | 1. Пользователь входит на главную страницу.<br>2. Пользователь запоминает количество включенных и выключенных компьютеров в аудиториях. | После выполнения действий с компьютерами, графики меняются в соответствии с текущим состоянием компьютеров в аудиториях. | Да |

Продолжение таблицы 2

| | | | | |
|---|---|---|---|---|
| | | 3. Пользователь меняет состояния компьютеров путем выполнения команд на включение или выключение всех компьютеров.<br>4. Пользователь ожидает завершения выбранной команды.<br>5. Пользователь нажимает на кнопку “Обновить состояние всех ПК”.<br>6. Пользователь ожидает завершение команды. | | |
| 5 | Выполнение действий с компьютерами в аудитории. | 1. Пользователь входит на страницу аудитории.<br>2. Пользователь нажимает на одну из кнопок действий: “Включить все ПК” или “Выключить все ПК” или “Перезагрузить все ПК”. | Пользователю показывается уведомление о выполнении выбранного действия. После того как уведомление пропадает все исправные компьютеры в аудитории | Да |

Продолжение таблицы 2

| | | | | |
|---|---|---|---|---|
| | | 3. Пользователь ожидает завершение выполнения команды. Время ожидания зависит от выбранного действия. | должны соответствовать состоянию выбранного действия. | |
| 6 | Поверка отображения страницы 110 или 112 аудитории. | Пользователь входит на страницу 110 или 112 аудитории. | Пользователю показывается запрашиваемая страница, а именно: 6 кнопок управления, карта расстановки компьютеров, IP адреса, статус компьютеров. | Да |
| 7 | Смена режима загрузки ПК. | 1. Пользователь входит на любую страницу веб-приложения.<br>2. Меняет режим загрузки компьютеров путем нажатия на кнопку Win или PXE.<br>3. Переводит любой/любые | 1. Пользователю показывается уведомление что режим загрузки изменен.<br>2. При включении компьютер/компьютеры подключаются к серверу | Да |

Продолжение таблицы 2

| | | | | |
|---|---|---|---|---|
| | | компьютеры в любой | pxe-bot и получа- | |

| | | | | |
|---|---|---|---|---|
| | | аудитории в режим включения | ют с него Linux образ.<br>3. Загружается операционная система Linux. | |

**Тестирование интерфейса пользователя.**

Интерфейс был протестирован на браузерах Google Chrome 50 и выше, Mozilla Firefox 50 и выше, Opera 50 и выше. Была выполнена проверка следующих требований:

- вид элементов при уменьшении и увеличении окна браузера;
- правильность написания текста;
- правильность отображения элементов управления компьютерами;
- правильность расположения элементов карты компьютерах классов.
- унификация дизайна (цвет, шрифт).

В результате тестирования не возникло никаких проблем с верстkой. Интерфейс во всех браузерах был идентичным. Тест пройден успешно.

# ЗАКЛЮЧЕНИЕ

В ходе работы над системой были решены следующие задачи:

1. Изучены существующие отечественные и зарубежные аналоги систем управления компьютерными классами.
2. Выполнено проектирование программной системы.
3. Выполнена реализация программной системы.
4. Выполнено функциональное тестирование разработанной программной системы.
5. Выполнено внедрение разработанной программной системы в лаборатории «Суперкомпьютерного Моделирования» ЮУрГУ.

В результате выполнения работы были решены все поставленные задачи, таким образом, цель работы достигнута в полном объеме.

В качестве направлений дальнейших улучшений системы можно выделить следующие:

1. Внедрение в систему утилиты PsExec, которая расширит возможности управления компьютерами, в частности, позволит выполнять удаленную установку и удаление программ, а также удаленно изменять реестр.
2. Добавление функционала, позволяющего удаленно создавать скриншоты рабочих столов, что позволит преподавателям смотреть чем занимаются студенты на их парах.

Система выполнена по лицензии GNU General Public License и доступна для скачивания в сети интернет с репозитория на сайте git.hpc.susu.ru [22].

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Craig L. Applying UML and Patterns: An Introduction to Object-oriented Analysis and Design and the Unified Process. / Craig L. – USA: Prentice Hall Professional, 2002. – 627 p.

2. Леоненков А.В. Самоучитель UML 2. / Леоненков А.В. – СПб.: БХВ-Петербург, 2007. – 576 с.

3. Канер С. Тестирование программного обеспечения. / Канер С., Фолк Дж., Кек Нгуен Е. – Киев: ДиаСофт, 2001. – 538 с.

4. Total Network Inventory. – https://www.softinventive.ru/products/total-network-inventory/. Дата обращения: 22.12.2018.

5. 10-Страйк: Удаленный Доступ. – https://www.10-strike.ru/remote-access/. Дата обращения: 22.12.2018.

6. Официальный сайт TeamViewer. – https://www.teamviewer.com/ru/. Дата обращения: 22.12.2018.

7. Официальный сайт LiteManager. – http://litemanager.ru/ Дата обращения: 22.12.2018.

8. Официальный сайт Ammyy admin. – http://www.ammyy.com/ru/. Дата обращения: 22.12.2018.

9. Официальный сайт RAdmin. – https://www.radmin.ru/. Дата обращения: 22.12.2018.

10. Django documentation. – https://docs.djangoproject.com/en/2.1/. Дата обращения: 01.02.2019.

11. Kostenetskiy P. SUSU Supercomputer Resources for Industry and fundamental Science / Kostenetskiy P., Semenikhina P. // Proceedings – 2018 Global Smart Industry Conference. – IEEE, 2018. – 7 p.

12. Карл И.В. Разработка требований к программному обеспечению. / Карл И.В. – М: Издательский центр Русская Редакция, 2004. – 576 с.

13. Mark, L. Learning Python / L. Mark. 2013. – 1594 p.

14. Роббинс, Д. HTML5, CSS3 и JavaScript. Исчерпывающее руководство / Д. М. Роббинс. М: Эксмо, 2014. – 528 с.

15. Bennett J. Practical Django Projects. / Bennett J. – USA: Apress, 2009. – 272 p.

16. Martelli A. Python Cookbook. / Martelli A., Ravenscroft A., Ascher D. – USA: O'Reilly Media, Inc., 2005. – 846 p.

17. Ravindran A. Django Design Patterns and Best Practices. / Ravindran A. – UK: Packt Publishing Ltd, 2015. – 222 p.

18. Официальный сайт IPXE. – http://ipxe.org/download. Дата обращения: 05.02.2019.

19. Официальная документация по стандартной библиотеке python. – https://docs.python.org/3/library/index.html. Дата обращения: 01.02.2019.

20. Современный учебник JavaScript. – https://learn.javascript.ru/. Дата обращения: 1.02.2019.

21. Русскоязычная документация по семейству операционных систем Ubuntu. – https://help.ubuntu.ru/. Дата обращения: 4.03.2019.

22. Репозиторий с разработанным веб-приложением. – https://git.hpc.susu.ru/class-manager/Class_Manager_WEB. Дата обращения: 1.06.2019.

# ПРИЛОЖЕНИЕ А

# ИСХОДНЫЙ КОД РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ

## Листинг А.1

```
import threading
import requests
import subprocess
import pickle
import codecs
import queue
import time
import re
import os
from datetime import timedelta
from datetime import datetime
from . import models
from Class_Controller.models import ObjPC, models

# Тригеры
if os.name == 'nt':
    DEBUG_MODE = True
    slash = "\\"
else:
    DEBUG_MODE = False
    slash = "/"


# Конфигурации
CONFIG = {}
ARP_DATA = []
OS_WIN_IP = []
OS_LINUX_IP = []
NUMBER_OF_THREADS_EXECUTED_FOR_IP = 0
AUD_INFO = {}


def read_config_map_aud(class_id):
    """
    Функция чтения конфигурации растановки ПК в аудитории.
    :param class_id: номер аудитории
    :return: конфигурация аудитории для составления карпы ПК в аудитории
    """
    path_cfg = os.getcwd()
    data =
eval(open("{0}{2}Class_Controller{2}cfg_{1}_location.cfg".format(path_cfg,
class_id, slash), 'r').read())

    return data

def read_config_pc():
    """
    Функция чтения конфигурации имен и MAC адресов ПК.
    :return: конфигурации имен и MAC адресов ПК.
    """
    _config = {}
    path_cfg = os.getcwd()
    data = eval(open("{0}{1}Class_Controller{1}classes.cfg".format(path_cfg,
slash), 'r').read())
```

```
    for pc in data:
        pc = data.get(pc)
        pc['mac'] = pc['mac'].replace(":", ".")
        pc['mac'] = pc['mac'].upper()

    _config.update({"110": {x: data.get(x) for x in data if "PC110" in
x.upper()}, "112": {x: data.get(x) for x in data if "PC112" in x.upper()}})

    return _config


def get_mac(pcname, class_id):
    """
    Функция получения MAC адреса из конфигурации считанной функцией
read_config_pc
    :param pcname: имя компьютера
    :param class_id: номер аудитории
    :return: MAC адрес
    """
    try:
        if class_id in pcname:
            return CONFIG[class_id][pcname.upper()]['mac']
    except Exception as ex:
        print(pcname, ex)

    return None


def get_macs():
    """
    Функция получения необработанных данных ARP таблицы
    :return: ничего не возвращает
    """
    print("{0} class_controller.py вход get_macs".format(datetime.now()))
    global ARP_DATA
    response = requests.get("http://192.168.110.6:8080/get_macs/?flag=True")
    _answer = response.text
    ARP_DATA = pickle.loads(codecs.decode(_answer.encode(), "base64"))
    print("{0} class_controller.py выход get_macs".format(datetime.now()))


def execute_cmd(command):
    """
    Функция выполнения команд в консоле
    :param command:
    :return: ответ от выполненной команды
    """
    system_info = None
    if os.name == 'nt':
        system_info = subprocess.STARTUPINFO()
        system_info.dwFlags |= subprocess.STARTF_USESHOWWINDOW
    proc = subprocess.Popen(command, stdin=subprocess.PIPE,
stderr=subprocess.PIPE, stdout=subprocess.PIPE,
                            startupinfo=system_info, shell=True)
    info = proc.stdout.read().decode('cp866')
    return info
```

```
def get_ip(pcname, class_id):

"""
    Функция получения IP адресов
    :param pcname: Имя компьютера
    :param class_id: номер аудитории
    :return: IP адресс
    """
    pc_mac = get_mac(pcname, class_id)
    macs = ARP_DATA
    ip = None

    for mac in macs:
        if macs.get(mac).upper() == pc_mac:
            ip = mac.strip()
            break
    return ip


def send_ping(pcname, class_id):
    """
    Функция проверки IP на доступность и запись в БД информации об IP и типе
ОС
    Если главный модуль запускается на Win OS:
        -n 3 -l 30 -w 3 {0}
        -n число — Число отправляемых эхо-запросов.
        -l размер — Размер поля данных в байтах отправляемого запроса.
        -w таймаут — Максимальное время ожидания каждого ответа в миллисекун-
дах.
    Если главный модуль запускается на Linux OS:
        -c 3 -s 30 {0}
        -c - количество пакетов, которые нужно отправить;
        -s - размер одного пакета;
    :param pcname: Имя компьютера
    :param class_id: номер аудитории
    :return: True если есть пинг и False если нет пинга
    """
    global NUMBER_OF_THREADS_EXECUTED_FOR_IP
    values_for_update_ip = {"ip_pc": None, 'os_type': None}
    models.IP.objects.update_or_create(name_pc=pcname, de-
faults=values_for_update_ip)

    pc_ip = get_ip(pcname, class_id)

    def update_ip_and_os_type_in_db(os_type, id):
        """
        Обновляет в базе данных информацию об IP и типе ОС на ПК
        :param os_type: тип ОС
        :param id:
        :return: ничего
        """
        _values_for_update_ip = {"ip_pc": pc_ip, 'os_type': os_type}
        models.IP.objects.update_or_create(name_pc=pcname, de-
faults=_values_for_update_ip)

    if pc_ip:
        if DEBUG_MODE:
            _ping = execute_cmd("ping -n 1 -l 32 -w 2 {0}".format(pc_ip))
        else:
```

```
            _ping = execute_cmd("ping -c 1 -s 32 {0}".format(pc_ip))
 ttl = re.findall(r'(TTL[=][\d]{2,3}|ttl[=][\d]{2,3})', _ping)
        if ttl:
            ttl_time = re.findall(r'([\d]{3}|[\d]{2})', ttl[0])

            if int(ttl_time[0]) > 64:  # Windows OS
                OS_WIN_IP.append(pc_ip)
                update_ip_and_os_type_in_db('Windows', 1)
                NUMBER_OF_THREADS_EXECUTED_FOR_IP += 1

                return True
            elif int(ttl_time[0]) <= 64:  # Linux OS
                OS_LINUX_IP.append(pc_ip)
                update_ip_and_os_type_in_db('Linux', 2)

                NUMBER_OF_THREADS_EXECUTED_FOR_IP += 1
                return True
        else:
            NUMBER_OF_THREADS_EXECUTED_FOR_IP += 1

            return None
    else:
        NUMBER_OF_THREADS_EXECUTED_FOR_IP += 1

    return None


def ping_thread_arp():
    """
    Отправляет запрос на выполнение пинга диапозона IP
    :return:
    """
    print("{0} class_controller.py вход
ping_thread_arp".format(datetime.now()))
    requests.get("http://192.168.110.6:8080/update_arp/?flag=arp")
    print("{0} class_controller.py выход
ping_thread_arp".format(datetime.now()))


def check_all(class_id):
    """
    Проверяет включены ли ПК в аудитории
    :param class_id: номер аудитории
    :return: состояния ПК (ВКЛ/ВЫКЛ)
    """
    states = {}
    threads = []

    exit_flag = False

    class PingThread(threading.Thread):
        def __init__(self, queue):
            threading.Thread.__init__(self)
            self.queue = queue

        def run(self):
            check_ping(self.queue)

    def check_ping(queue):
```

```
        while not exit_flag:
 if not pingQueue.empty():
                pc_state = "Offline"
                pc_name = queue.get()
                pc_ping = send_ping(pc_name, class_id)
                queueLock.acquire()
                if pc_ping:
                    pc_state = "Online"
                states.update({pc_name: pc_state})
                queueLock.release()

    queueLock = threading.Lock()
    pingQueue = queue.Queue(0)

    for pc in CONFIG[class_id]:
        pingQueue.put(pc)
    for i in range(0, len(CONFIG[class_id])):
        thread = PingThread(pingQueue)
        thread.start()
        threads.append(thread)

    while not pingQueue.empty():
        pass

    exit_flag = 1

    for thread in threads:
        thread.join()

    return states


def response_for_user(_action, _class_id, _calling_function, what_need,
pc_name=None):
    """

    :param _action: Действие которое выбрал пользователь.
    :param _class_id: Номер аудитории. Используется при вывозе функций дейст-
вий для аудиторий power_all
    :param _calling_function: Функция которая вызвала функцию
response_for_user
    :param what_need: Дополнительный индентификатор нужного сообщения
    :param pc_name: Имя компьютера
    :return: Текст сообщения, тип уведомления, время уведомления
    """
    print("response_for_user _action: {0} _class_id: {1} _calling_function:
{2} what_need: {3}".format(_action, _class_id, _calling_function, what_need))
    response_text = 'Ошибка функции ответа пользователю'
    type_notif = 'error'
    time_notif = 4000
    trigers_action_rooms = None

    def action_ru_text():
        """
        Возвращает текстовый ответ выполняемого действия с ПК
        :return:
        """
        if what_need == 'global_action_aud_info_action_aud_block' or
what_need == 'global_action_aud_info':
```

```
            if _action == 'on':
return 'включение'
            elif _action == 'off':
                return 'выключение'
            elif _action == 'reboot':
                return 'перезагрузке'
        elif _calling_function == "power_all":
            if trigers_action_rooms.on_all_pc:
                return 'включение'
            elif trigers_action_rooms.off_all_pc:
                return 'выключение'
            elif trigers_action_rooms.reboot_all_pc:
                return 'перезагрузка'
            elif _action == 'on':
                return 'включение'
            elif _action == 'off':
                return 'выключение'
            elif _action == 'reboot':
                return 'перезагрузка'
        elif _calling_function == "change_status_pc":
            if what_need == "block_action_on":
                return 'Компьютер {0} уже включается'.format(pc_name)
            elif what_need == "block_action_off":
                return 'Компьютер {0} уже выключается'.format(pc_name)
            elif what_need == "block_action_reboot":
                return 'Компьютер {0} уже перезагружается'.format(pc_name)
            elif what_need == "action_not_start":
                if _action == 'on':
                    return 'Компьютер {0} уже включен'.format(pc_name)
                elif _action == 'off':
                    return 'Компьютер {0} не включен'.format(pc_name)
                elif _action == 'reboot':
                    return 'Компьютер {0} не включен'.format(pc_name)
            elif what_need == "action_start_pc":
                if _action == 'on':
                    return 'Включается компьютер {0}'.format(pc_name)
                elif _action == 'off':
                    return 'Выключается компьютер {0}'.format(pc_name)
                elif _action == 'reboot':
                    return 'Перезагружается компьютер {0}'.format(pc_name)
        else:
            return 'Ошибка action_ru_text'

    def action_time():
        """
        Возвращает время выполнения действия
        :return:
        """
        if _action == 'on':
            time_on_win7 = 150
            time_on_kiosk = 40
            return 150
        elif _action == 'off':
            time_off_win7 = 15
            time_off_kiosk = 5
            return 15
        elif _action == 'reboot':
            time_reboot_win7 = 200
            time_reboot_kiosk = 20
```

```
            return 200
 elif _calling_function == 'manual_update_states':
            return 30
        else:
            pass

    if _calling_function == 'change_status_pc':
        return action_ru_text()
    elif _calling_function == "manual_update_states":
        if what_need == 'success_func':
            response_text = "Выполняется обновление информации о
ПК".format(action_ru_text())
            type_notif = 'success'
            _time_notif = action_time()
            time_notif = _time_notif * 1000
        elif what_need == 'block':
            response_text = "Уже выполняется обновление информации о
ПК".format(action_ru_text())
            type_notif = 'info'

        else:
            response_text = 'Ошибка функции ответа пользователю вызвашая
функция manual_update_states'
            type_notif = 'error'
    elif _calling_function == "power_all":
        trigers_action_rooms = mod-
els.TrigersActionRooms.objects.get(classroom=_class_id)
        if what_need == 'action_aud_block':
            time_now = datetime.now()  # .strftime("%Y-%m-%d %H:%M:%S")
            time_start = date-
time.strptime(str(trigers_action_rooms.time_action), '%Y-%m-%d %H:%M:%S')
            time_left = (time_now - time_start).total_seconds()
            time_left = action_time() - time_left
            response_text = "Уже выполняется {0} ПК в {1}
аудитории".format(action_ru_text(), _class_id)
            type_notif = 'info'
            time_notif = int(time_left) * 1000
        elif what_need == 'action_aud_info':
            response_text = "Выполняется {0} ПК в {1}
аудитории".format(action_ru_text(), _class_id)
            type_notif = 'success'
            _time_notif = action_time()
            time_notif = _time_notif * 1000
        elif what_need == 'global_action_aud_info_action_aud_block':
            response_text = "Уже выполняется {0} ПК в аудиториях 110 и
112".format(action_ru_text())
            type_notif = 'info'
        elif what_need == 'global_action_aud_info':
            response_text = "Выполняется {0} ПК в аудиториях 110 и
112".format(action_ru_text())
            type_notif = 'success'
            _time_notif = action_time()
            time_notif = _time_notif * 1000
        elif what_need == 'aud_not_pc':
            print(action_ru_text())
            response_text = "Нет компьютеров для
{0}".format(action_ru_text())
            type_notif = 'info'
        else:
            response_text = 'Ошибка функции ответа пользователю вызвашая
```

```
функция power_all'
            type_notif = 'error'
    elif what_need == 'kiosk_response':
        response_text = "Выбран режим загрузки ПК
Kiosk".format(_calling_function)
        type_notif = 'success'
    elif what_need == 'windows7_response':
        response_text = "Выбран режим загрузки ПК Windows
7".format(_calling_function)
        type_notif = 'success'
    elif what_need == 'error_response':
        response_text = "Ошибка функции {0}".format(_calling_function)
        type_notif = 'info'
    else:
        response_text = 'Ошибка функции ответа пользователю'
        type_notif = 'error'

    return response_text, type_notif, time_notif


def change_status_pc(action, pcname, flag_action_all, time_action, class_id):
    """
    Меняет статус ПК путем выполнения включения, выключения и перезагрузки
    :param action: действие с ПК
    :param pcname: Имя компьютера
    :param flag_action_all: Флаг указывает что вызво прищел от функции дейст-
вий с всеми ПК
    :param time_action: время действия
    :param class_id: номер аудитории
    :return: Текст уведомления, тип уведомления, время уведомления
    """

    pcname = pcname.upper()
    threads = []
    threads_update_pc_flag = []

    pc_ip = get_ip(pcname, class_id)

    if not flag_action_all:
        send_ping(pcname, class_id)

    def time_action_local(_type_os):
        """
        Возвращает время выполнения действий
        :param _type_os: тип операционной системы
        :return:
        """
        time = 15
        if action == "on":
            if _type_os == "win":
                time = 150
            elif _type_os == "linux":
                time = 40
        elif action == "off":
            if _type_os == "win":
                time = 15
            elif _type_os == "linux":
                time = 5
        elif action == "reboot":
            if _type_os == "win":
```

```
                time = 200
    elif _type_os == "linux":
                time = 20
        return time

    def update_trigers_pc_action(pc_name, action_on, action_off, ac-
tion_reboot):
        """
        Изменяет тригеры блокировки действий с ПК
        :param pc_name: Имя компьютера
        :param action_on: True или False при ВКЛ
        :param action_off: True или False при ВЫКЛ
        :param action_reboot: True или False при перезагрузке
        :return:
        """

        values = {"on": action_on, "off": action_off, "reboot": ac-
tion_reboot}
        models.TrigersActionPCS.objects.update_or_create(pc_name=pc_name, de-
faults=values) # нет функции по созднаию обьектов

    def flags_action():
        """
        Возвращает True или False взависимости от действия
        :return:
        """
        if action == "on":
            action_on = True       # On
            action_off = False     # Off
            action_reboot = False  # Reboot
            return update_trigers_pc_action(pcname, action_on, action_off,
action_reboot)
        elif action == "off":
            action_on = False      # On
            action_off = True      # Off
            action_reboot = False  # Reboot
            return update_trigers_pc_action(pcname, action_on, action_off,
action_reboot)
        elif action == "reboot":
            action_on = False      # On
            action_off = False     # Off
            action_reboot = True   # Reboot
            return update_trigers_pc_action(pcname, action_on, action_off,
action_reboot)

    class ClockThreadChangeTrigersPCFalse(threading.Thread):
        def __init__(self, timed):
            threading.Thread.__init__(self)
            self.pc_name = pcname
            self.timed = timed

        def run(self):
            change_trigers_pc_false(self.pc_name, self.timed)

    def change_trigers_pc_false(_pc_name, _timed):
        """
        После выполнения действий обнуляет тригеры блокировок ПК
        :param _pc_name: Имя компьютера
        :param _timed:
```

```
        time.sleep(_timed)
        update_trigers_pc_action(_pc_name, False, False, False)

    for thread_pc_flag in threads_update_pc_flag:
        thread_pc_flag.join()

    class ChangeState(threading.Thread):
        def __init__(self, _pc_ip, _pcname):
            threading.Thread.__init__(self)
            self._pc_ip = _pc_ip
            self._pcname = _pcname

        def run(self):
            if action == "off":
                pow_off(self._pc_ip, self._pcname)
            elif action == "reboot":
                reboot(self._pc_ip, self._pcname)
            elif action == "on":
                pow_on(self._pc_ip, self._pcname)
            else:
                print("Неверный action {0}".format(action))

    def reboot(ip, name):
        """
        Перезагружает ПК
        :param ip: IP адресс ПК
        :param name: Имя компьютера
        :return:
        """
        if ip in OS_WIN_IP:
            time_action = time_action_local("win")
            thread_pc_flag = ClockThreadChangeTrigersPCFalse(time_action)
            thread_pc_flag.start()
            flags_action()
            re-
quests.get("http://192.168.110.6:8080/reboot_pc/?pc_ip={0}&pc_name={1}".forma
t(ip, name))
            threads_update_pc_flag.append(thread_pc_flag)
        elif ip in OS_LINUX_IP:
            time_action = time_action_local("linux")
            thread_pc_flag = ClockThreadChangeTrigersPCFalse(time_action)
            thread_pc_flag.start()
            flags_action()
            make_action_kiosk("reboot", ip)
            threads_update_pc_flag.append(thread_pc_flag)
        else:
            print("Error неверно имя ПК: {0}".format(name))

    def pow_on(ip, name):
        """
        Включает ПК
        :param ip:  IP адресс ПК
        :param name: Имя компьютера
        :return:
        """
        if ip in OS_WIN_IP or ip in OS_LINUX_IP:
            print("Компьютер {0} уже включен".format(name))
```

```
        else:
            type_os_win = models.ClassBoot.objects.get(boot_name="windows7")
type_os_linux = models.ClassBoot.objects.get(boot_name="kiosk")
            if type_os_win.boot_flag:
                type_os = "win"
            elif type_os_linux.boot_flag:
                type_os = "linux"
            else:
                type_os = "win"
            time_action = time_action_local(type_os)
            thread_pc_flag = ClockThreadChangeTrigersPCFalse(time_action)
            thread_pc_flag.start()
            flags_action()
            pc_mac = get_mac(name, class_id)
            re-
quests.get("http://192.168.110.6:8080/on_pc/?flag=on&pc_mac={0}&pc_name={1}".
format(pc_mac, name))
            threads_update_pc_flag.append(thread_pc_flag)

    def pow_off(ip, name):
        """
        Выключает ПК
        :param ip:  IP адресс ПК
        :param name: Имя компьютера
        :return:
        """
        if ip in OS_WIN_IP:
            time_action = time_action_local("win")
            thread_pc_flag = ClockThreadChangeTrigersPCFalse(time_action)
            thread_pc_flag.start()
            flags_action()
            re-
quests.get("http://192.168.110.6:8080/off_pc/?pc_ip={0}&pc_name={1}".format(i
p, name))
            threads_update_pc_flag.append(thread_pc_flag)

        elif ip in OS_LINUX_IP:
            time_action = time_action_local("linux")
            thread_pc_flag = ClockThreadChangeTrigersPCFalse(time_action)
            thread_pc_flag.start()
            flags_action()
            make_action_kiosk("off", ip)
            threads_update_pc_flag.append(thread_pc_flag)

        else:
            print("Error неверно имя ПК: {0}".format(name))

    for thread in threads:
        thread.join()

    if action == "off":
        pc_state = models.TrigersActionPCS.objects.get(pc_name=pcname)
        if pc_state.off:
            if not flag_action_all:
                print("Компьютер {0} уже ".format(pcname))
                return response_for_user(action, None, "change_status_pc",
"block_action_off", pcname)
        elif pc_state.on:
            if not flag_action_all:
```

```
                print("Компьютер {0} уже ".format(pcname))
                return response_for_user(action, None, "change_status_pc",
```



```
"block_action_on", pcname)
        elif pc_state.reboot:
            if not flag_action_all:
                print("Компьютер {0} уже ".format(pcname))
                return response_for_user(action, None, "change_status_pc",
"block_action_reboot", pcname)
        else:
            if pc_ip in OS_WIN_IP or pc_ip in OS_LINUX_IP:
                thread = ChangeState(pc_ip, pcname)
                thread.start()
                threads.append(thread)
                print("Выключается компьютер: {0}".format(pcname))
                if not flag_action_all:
                    return response_for_user(action, None,
"change_status_pc", "action_start_pc", pcname)
            else:
                print("Компьютер {0} не включен".format(pcname))
                if not flag_action_all:
                    return response_for_user(action, None,
"change_status_pc", "action_not_start", pcname)
    elif action == "reboot":
        pc_state = models.TrigersActionPCS.objects.get(pc_name=pcname)
        if pc_state.off:
            if not flag_action_all:
                print("Компьютер {0} уже ".format(pcname))
                return response_for_user(action, None, "change_status_pc",
"block_action_off", pcname)
        elif pc_state.on:
            if not flag_action_all:
                print("Компьютер {0} уже ".format(pcname))
                return response_for_user(action, None, "change_status_pc",
"block_action_on", pcname)
        elif pc_state.reboot:
            if not flag_action_all:
                print("Компьютер {0} уже ".format(pcname))
                return response_for_user(action, None, "change_status_pc",
"block_action_reboot", pcname)
        else:
            if pc_ip in OS_WIN_IP or pc_ip in OS_LINUX_IP:
                thread = ChangeState(pc_ip, pcname)
                thread.start()
                threads.append(thread)
                print("Компьютер {0} перезагружается".format(pcname))
                if not flag_action_all:
                    return response_for_user(action, None,
"change_status_pc", "action_start_pc", pcname)
            else:
                print("Компьютер {0} не включен".format(pcname))
                if not flag_action_all:
                    return response_for_user(action, None,
"change_status_pc", "action_not_start", pcname)
    elif action == "on":
        pc_state = models.TrigersActionPCS.objects.get(pc_name=pcname)
        if pc_state.off:
            if not flag_action_all:
                print("Компьютер {0} уже ".format(pcname))
                return response_for_user(action, None, "change_status_pc",
"block_action_off", pcname)
```

```
        elif pc_state.on:
            if not flag_action_all:
print("Компьютер {0} уже ".format(pcname))
                return response_for_user(action, None, "change_status_pc",
"block_action_on", pcname)
        elif pc_state.reboot:
            if not flag_action_all:
                print("Компьютер {0} уже ".format(pcname))
                return response_for_user(action, None, "change_status_pc",
"block_action_reboot", pcname)
        else:
            if pc_ip in OS_WIN_IP or pc_ip in OS_LINUX_IP:
                print("Компьютер {0} уже включен".format(pcname))
                if not flag_action_all:
                    return response_for_user(action, None,
"change_status_pc", "action_not_start", pcname)
            else:
                thread = ChangeState(pc_ip, pcname)
                thread.start()
                threads.append(thread)
                print("Включается компьютер: {0}".format(pcname))
                if not flag_action_all:
                    return response_for_user(action, None,
"change_status_pc", "action_start_pc", pcname)


    else:
        raise Exception('test exception')


def power_all_action(class_id, action, flag_all):
    """
    print("{0} class_controller.py power_all_action -> if action = {1} ->
{2}".format(datetime.now(), action_ru_text(), class_id))
    Выполняет действия с всеми ПК в аудиторияи
    :param class_id: номер аудитории
    :param action: действие
    :param flag_all: True при действии с ПК 2 аудиторий
    :return:
    """
    global NUMBER_OF_THREADS_EXECUTED_FOR_IP

    time_off_win7 = 15
    time_off_kiosk = 5
    time_reboot_win7 = 200
    time_reboot_kiosk = 20
    time_on_win7 = 150
    time_on_kiosk = 40

    threads = []
    threads_send = []
    NUMBER_OF_THREADS_EXECUTED_FOR_IP = 0


    def number_pc_in_aud():
        """
        Вовзращает количество ПК в аудитории
        :return:
        """
```

```
number_pc = AUD_INFO[class_id]['number_pc']
 return number_pc
    _number_pc_in_aud = number_pc_in_aud()

    class PingThread(threading.Thread):
        def __init__(self, _pc, _class_id):
            threading.Thread.__init__(self)
            self.pc = _pc
            self.class_id = _class_id
        def run(self):
            send_ping(self.pc, self.class_id)

    def update_trigers_room_action(class_id, flag1, flag2, flag3):
        """
        Изменяет тригеры блокировок действий в аудиториях
        :param class_id: номер аудитории
        :param flag1:
        :param flag2:
        :param flag3:
        :return:
        """

        time_now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        values = {"on_all_pc": flag1, "off_all_pc": flag2, "reboot_all_pc":
flag3, "time_action": time_now}
        mod-
els.TrigersActionRooms.objects.update_or_create(classroom=class_id, de-
faults=values)

    def update_list_ip():
        """
        Обновляет переменную содержащую IP адреса
        :return:
        """
        for pc in CONFIG[class_id]:
            _thread_ip = PingThread(pc, class_id)
            _thread_ip.start()
            threads_send.append(_thread_ip)

    for thread_ip in threads_send:
        thread_ip.join()

    def flag_action():
        """
        Изменяет тригеры блокировок действий в аудиториях
        :return:
        """
        if action == "on":
            flag1 = True   # On
            flag2 = False  # Off
            flag3 = False  # Reboot
        elif action == "off":
            flag1 = False  # On
            flag2 = True   # Off
            flag3 = False  # Reboot
        elif action == "reboot":
            flag1 = False  # On
            flag2 = False  # Off
```

```
            flag3 = True   # Reboot
        else:
```

Продолжение приложения

```
            flag1 = False  # On
 flag2 = False  # Off
            flag3 = False  # Reboot

        update_trigers_room_action(class_id, flag1, flag2, flag3)

    def change_trigers_pc_false(_timed, _class_id):
        """
        Обнуляет тригеры блокировок действий в аудиториях
        :param _timed:
        :param _class_id:
        :return:
        """
        time.sleep(_timed)
        update_trigers_room_action(_class_id, False, False, False)

    def start_thread_action(time_action):
        """
        Запускает действия с ПК в многопоточном режиме
        :param time_action: время действия
        :return: текст уведомления, время уведомления, тип уведомления
        """
        flag_action()

        thread = ClockThreadChangeTrigersPCFalse(time_action)
        thread.start()
        threads.append(thread)

        for pc in CONFIG[class_id]:
            change_status_pc(action, pc, True, time_action, class_id)

        if flag_all:
            response_text, type_notif, _time_notif = re-
sponse_for_user(action, class_id, 'power_all', 'global_action_aud_info')
            flag_for_all = 1
        else:
            response_text, type_notif, _time_notif = re-
sponse_for_user(action, class_id, 'power_all', 'action_aud_info')
            flag_for_all = 0

        return response_text, type_notif, _time_notif, flag_for_all

    class ClockThreadChangeTrigersPCFalse(threading.Thread):
        def __init__(self, timed):
            threading.Thread.__init__(self)
            self.class_id = class_id
            self.timed = timed

        def run(self):
            change_trigers_pc_false(self.timed, self.class_id)

    for thread in threads:
        thread.join()

    def preparation_for_action():
        """
        Подготавливает данные для выполнения действий
```

```
        :return:
        """
```



```
        ping_thread_arp()
get_macs()
        update_list_ip()
        while NUMBER_OF_THREADS_EXECUTED_FOR_IP < _number_pc_in_aud:
            time.sleep(0.2)
        _flag = models.ClassBoot.objects.get(boot_name='windows7')
        _pc_online_all = len(OS_WIN_IP) + len(OS_LINUX_IP)
        _pc_online_windows = len(OS_WIN_IP)
        _pc_online_linux = len(OS_LINUX_IP)
        return _flag, _pc_online_all, _pc_online_windows, _pc_online_linux

    if action == "on":
        __flag, __pc_online_all, __pc_online_windows, __pc_online_linux =
preparation_for_action()
        if __pc_online_all < _number_pc_in_aud:
            if __flag.boot_flag:
                return start_thread_action(time_on_win7)
            else:
                return start_thread_action(time_on_kiosk)
        else:
            text, notification, time_notif = response_for_user(action,
class_id, 'power_all', 'aud_not_pc')
            return text, notification, time_notif, 0
    elif action == "off":
        __flag, __pc_online_all, __pc_online_windows, __pc_online_linux =
preparation_for_action()
        if __pc_online_windows > 0:
            return start_thread_action(time_off_win7)
        elif __pc_online_linux > 0:
            return start_thread_action(time_off_kiosk)
        else:
            text, notification, time_notif = response_for_user(action,
class_id, 'power_all', 'aud_not_pc')
            return text, notification, time_notif, 0
    elif action == "reboot":
        __flag, __pc_online_all, __pc_online_windows, __pc_online_linux =
preparation_for_action()
        if __pc_online_windows > 0:
            return start_thread_action(time_reboot_win7)
        elif __pc_online_linux > 0:
            return start_thread_action(time_reboot_kiosk)
        else:
            text, notification, time_notif = response_for_user(action,
class_id, 'power_all', 'aud_not_pc')
            return text, notification, time_notif, 0
    else:
        return response_for_user(None, None, 'power_all_action',
'error_response')


def power_all(class_id, action, flag_all):
    """
    Проверяет выполнение действий с ПК в аудиториях на предмет блокировки
    print("{0} class_controller.py вход power_all({1},
{2})".format(datetime.now(), class_id, action))
    print("{0} class_controller.py Тригер class_id == 110 ac-
tion".format(datetime.now()))
```

```
    print("{0} class_controller.py power_all({1},
{2})".format(datetime.now(), class_id, action))
```



```
   print("{0} class_controller.py Ошибка power_all({1},
{2})".format(datetime.now(), class_id, action))
    :param class_id: номер аудитории
    :param action: действие
    :param flag_all: True если впыолняет дейтсвие с ПК в 2 аудиториях
    :return:
    """
    global OS_LINUX_IP, OS_WIN_IP

    OS_LINUX_IP = []
    OS_WIN_IP = []
    trigersactionrooms = mod-
els.TrigersActionRooms.objects.get(classroom=class_id)

    if trigersactionrooms.off_all_pc or trigersactionrooms.on_all_pc or tri-
gersactionrooms.reboot_all_pc:
        if flag_all:
            text, notification, time_notif = response_for_user(action,
class_id, 'power_all', 'global_action_aud_info_action_aud_block')
            return text, notification, time_notif, 2
        else:
            text, notification, time_notif = response_for_user(action,
class_id, 'power_all', 'action_aud_block')
            return text, notification, time_notif, 0
    elif not trigersactionrooms.off_all_pc or not trigersactio-
nrooms.on_all_pc or not trigersactionrooms.reboot_all_pc:
        return power_all_action(class_id, action, flag_all)
    else:
        return response_for_user(None, None, 'power_all', 'error_response')


def update_boot_config(boot_name, flag):
    """
    Изменеят тип загрузки ПК
    print("{0} class_controller.py {1} {2}".format(datetime.now(), boot_name,
flag))
    :param boot_name: Тип загрузки
    :param flag:
    :return:
    """
    windows_7 = 'windows7'
    kiosk = 'kiosk'
    acronis = 'Acronis'
    _flag_win7 = True
    _flag_kiosk, _flag_acronis = False, False

    def update_bd_boot(_boot_name, _flag):
        _value = models.ClassBoot.objects.get(boot_name=_boot_name)
        _value.boot_flag = _flag
        _value.save()

    if boot_name == 'windows7' and flag:
        _flag_win7 = True
        _flag_kiosk, _flag_acronis = False, False
    elif boot_name == 'kiosk' and flag:
        _flag_kiosk = True
        _flag_win7, _flag_acronis = False, False
```

```
    update_bd_boot(windows_7, _flag_win7)
```



```
    update_bd_boot(kiosk, _flag_kiosk)
 update_bd_boot(acronis, _flag_acronis)

    response = re-
quests.get("http://192.168.110.4:8080/config/?mode={0}".format(boot_name))


def make_action_kiosk(action, pc_ip):
    """
    Изменеят тип загрузки ПК в модуле pxe-bot
    :param action:
    :param pc_ip:
    :return:
    """
    response = re-
quests.get("http://192.168.110.4:8080/make_action/?action={0}&pcip={1}".forma
t(action, pc_ip))


def who_user(_pc_name, Flag):
    """
    Проверят кто на текущий момент залогинен на ПК
    :param _pc_name: Имя компьютера
    :param Flag:
    :return:
    """
    print("{0} class_controller.py вход who_user {1}".format(datetime.now(),
_pc_name))
    threads = []

    class GetUserName(threading.Thread):
        def __init__(self, _pc):
            threading.Thread.__init__(self)
            self.pc = _pc

        def run(self):
            get_name(self.pc)

    def update_user_name_in_bd(_user_name):
        """
        Обновляет в БД информаци о том кто на текущий момент залогинен на ПК
        :param _user_name:
        :return:
        """
        values_for_update = {"user_name": _user_name}
        models.IP.objects.update_or_create(name_pc=_pc_name, de-
faults=values_for_update)

    def get_name(_pc_name):
        """
        Запрашивает информаци о том кто на текущий момент залогинен на ПК
        :param _pc_name: Имя компьютера
        :return:
        """
        response = re-
quests.get("http://192.168.110.6:8080/who_user/?flag=True&pc_name={0}".format
(_pc_name))
```

```
        answer = response.text
        user_names = pickle.loads(codecs.decode(answer.encode(), "base64"))
```

Продолжение приложения

```
        if not user_names:

  update_user_name_in_bd('')

        #  Извлекает не так как хотелось бы ['LocalAdmin']
        for user_name in user_names:
            if user_names.get(user_name) == 'LocalAdmin':
                if len(user_names) > 1:
                    pass
                else:
                    _user = user_names[user_name]
                    _user = str(_user)
                    _user = _user[2:-2]
                    print(_user)
                    update_user_name_in_bd(_user)
            else:
                _user = user_names[user_name]
                _user = str(_user)
                _user = _user[2:-2]
                print(_user)
                update_user_name_in_bd(_user)

    for thread in threads:
        thread.join()

    if Flag:
        _thread = GetUserName(_pc_name)
        _thread.start()
        threads.append(_thread)
    else:
        values_for_update = {"user_name": None}
        models.IP.objects.update_or_create(name_pc=_pc_name, de-
faults=values_for_update)

    print("{0} class_controller.py выход who_user {1}".format(datetime.now(),
_pc_name))


def update_pc_state():
    """
    Обновляет информацию об состояниях ПК
    :return:
    """
    global OS_WIN_IP, OS_LINUX_IP
    threads = []
    OS_WIN_IP = []
    OS_LINUX_IP = []

    values_for_update = {"property": True}
    models.OtherFucntions.objects.update_or_create(function='manual_update',
defaults=values_for_update)

    class TimerUpdate(threading.Thread):
        def __init__(self):
            threading.Thread.__init__(self)
```

```
        def run(self):
            thread2 = TimerUpdate2()
            thread2.start()
```



```
            ping_thread_arp()

 get_macs()
            for aud in AUD_INFO:
                load_color_state(aud, AUD_INFO.get(aud).get('property'))

    class TimerUpdate2(threading.Thread):
        def __init__(self):
            threading.Thread.__init__(self)

        def run(self):
            change_trigers_manual_update(30)

    def change_trigers_manual_update(_timed):
        """
        Обновляет тригер блокировки ручного обновления
        :param _timed:
        :return:
        """
        time.sleep(_timed)
        _values_for_update = {"property": False}
        mod-
els.OtherFucntions.objects.update_or_create(function='manual_update', de-
faults=_values_for_update)


    def load_color_state(class_id, bd_id):
        """
        Обновляет информацию об состояниях ПК
        :param class_id: номер аудитории
        :param bd_id:
        :return:
        """
        print("{0} class_controller.py вход
load_color_state_{1}".format(datetime.now(), class_id))

        num_pc_online = 0
        states = check_all(class_id)

        for state in states:
            if "ONLINE" in states.get(state).upper():
                num_pc_online += 1
                who_user(state, True)

                values_for_update_on = {"name_pc": state, "color_pc":
"#86FF0D"}
                _color_on = mod-
els.Color.objects.update_or_create(name_pc=state, de-
faults=values_for_update_on)
            else:

                who_user(state, None)

                values_for_update_off = {"name_pc": state, "color_pc":
"#FF0D0D"}
                _color_off = mod-
```

```
els.Color.objects.update_or_create(name_pc=state, de-
faults=values_for_update_off)


        values_for_update_off = {"number_pc": num_pc_online}
```

Окончание приложения

```
        _color_off = mod-
els.NumberOn.objects.update_or_create(classroom=bd_id, de-
faults=values_for_update_off)

        print("{0} class_controller.py выход
load_color_state_{1}".format(datetime.now(), class_id))

    for thread in threads:
        thread.join()

    thread = TimerUpdate()
    thread.start()
    threads.append(thread)


def initial():
    """
    Запускает в момент включаения сервера для чтения статичной конфигурации
    :return:
    """
    global AUD_INFO
    items = models.AudName.objects.select_related().all()
    for item in items:
        AUD_INFO.update({item.classroom: {"number_pc": item.number_pc, "prop-
erty": item.property}})


initial()
CONFIG = read_config_pc()
```