|  |
| --- |
| **МОСКОВСКИЙ КОМИТЕТ ОБРАЗОВАНИЯ**  **ЛИЦЕЙ №1533 (ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ)** |

Выпускная работа

(Специальность «Прикладное программирование»)

Учащихся группы 11-4 Молчанова Михаила Алексеевича и Горбачева Александра Максимовича

**Название работы**

Программные средства для проведения электронного тестирования учащихся с помощью устройств на базе Android и web интерфейса.

Заказчик: Гиглавый Александр Владимирович

Консультанты: Славнов Константин Анатольевич, Завриев Николай Константинович

Оглавление

[1. Введение 4](#_Toc433498841)

[2. Постановка задачи 5](#_Toc433498842)

[3. Обзор аналогов 6](#_Toc433498843)

[3.1. MyTest 7](#_Toc433498844)

[3.2 Moodle 7](#_Toc433498845)

[3.3 Websoft CoursLine 7](#_Toc433498846)

[3.4 LIT Test 7](#_Toc433498847)

[4. Анализ предметной области 7](#_Toc433498848)

[4.1 Что такое тест 7](#_Toc433498849)

[4.2 Виды тестов 8](#_Toc433498850)

[4.3 Виды тестовых заданий 9](#_Toc433498851)

[5 Решение 9](#_Toc433498852)

[5.0 Архитектура системы 9](#_Toc433498853)

[5.1 Хранение данных 10](#_Toc433498854)

[5.1.1 Хранение тестов 11](#_Toc433498855)

[5.1.2 Хранение вопросов 11](#_Toc433498856)

[5.1.3 Хранение результатов теста 11](#_Toc433498857)

[5.1.4 Хранение данных о учениках 12](#_Toc433498858)

[5.1.5 Хранение данных о учителях 12](#_Toc433498859)

[5.2. Структура данных 12](#_Toc433498860)

[5.2.1 Взаимодействие классов 12](#_Toc433498861)

[5.2.2 Типы тестов 15](#_Toc433498862)

[5.2.3 Типы вопросов 15](#_Toc433498863)

[5.3 Прохождение теста 15](#_Toc433498864)

[5.4 Алгоритм работы сервера 16](#_Toc433498865)

[5.5 Программная реализация 16](#_Toc433498866)

[5.5.1 Реализация Python 16](#_Toc433498867)

[5.5.2 Среды разработки 17](#_Toc433498868)

[5.6 Ход работы 18](#_Toc433498869)

[5.6.1 Проблемы и решения 18](#_Toc433498870)

[6 Результат 19](#_Toc433498871)

[7 Выводы 25](#_Toc433498872)

[8 Список литературы 26](#_Toc433498873)

[9 Приложения 26](#_Toc433498874)

[9.1 Python 26](#_Toc433498875)

[9.2 Android 83](#_Toc433498876)

# Введение

Для проведения контроля знаний учащихся применяют множество различных методов:

* Устный ответ
* Письменная работа
* Тестовые задания

Каждый вариант оценивания знаний подходит для разных ситуаций.

Тестирование сейчас находит, в частности в школе, всё большее применение, в особенности из-за введения ЕГЭ. Тесты позволяют быстро получить результат, при относительно малых затратах времени на проверку. Применяют как бумажные, так и электронные тесты.

Электронное тестирование, в отличие от бумажного, имеет ряд преимуществ:

* Позволяют, почти сразу получить результат, не вынуждая учителя самому проверять ответы учащихся.
* На выходе выдаётся информация, готовая к анализу. Например, в виде таблицы Excel.
* Исключают ошибку преподавателя при проверке.

Для проведения электронного тестирования удобно использовать планшеты. Они мобильны и удобны для ученика.

На сегодняшний момент различают следующие виды тестов:

* Множественный выбор - испытуемому необходимо выбрать один или несколько правильных ответов из приведенного списка
* Альтернативный выбор - испытуемый должен ответить «да» или «нет»
* Установление соответствия - испытуемому предлагается установить соответствие элементов двух списков
* Установление последовательности - испытуемый должен расположить элементы списка в определенной последовательности
* Свободное изложение - испытуемый должен самостоятельно сформулировать ответ; никакие ограничения на них в задании не накладываются
* Дополнение - испытуемый должен сформулировать ответы с учетом предусмотренных в задании ограничений (например, дополнить предложение)

# Постановка задачи

Была поставлена задача разработать среду для проведения электронного тестирования с помощью Android устройств и web интерфейса. В неё входит набор инструментов для учителя и тестируемого:

Учитель:

* + Редактор тестов
  + Просмотр результатов тестов

Ученик:

* + Приложение Android и web сайт для прохождения тестирования

И были выдвинуты следующие критерии:

* + Пользователи делятся на учителя и учащихся. Учитель создаёт тест, а ученики проходят его и отправляют данные для проверки на сервер.
  + Данное приложение может использоваться на уроках по различным предметам для оценки учащихся.
  + Данное приложение должна обеспечивать учителей:
    1. Средством для создания и редактированием тестов.
    2. Возможностью раздать его ученикам через интернет.

А ученик, в свою очередь, имеет возможность:

* 1. Открыть тест
  2. Пройти его
  3. Отправить свои ответы на сервер для проверки

# 3. Обзор аналогов

Существует множество систем для электронного тестирования. Большинство из них предназначено для работы в браузере (Moodle, WebSoft CourseLab), есть системы для работы за компьютером на Windows, Mac, Linux (MyTest).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Продукт | Поддержка мобильных устройств | Медиа контент | WEB интерфейс | Стационарная версия |
| MyTest | Нет | Да | Нет | Да |
| Moodle | Нет.  Только web интерфейс | Да | Да | Нет |
| WebSoft CourseLab | Нет.  Только web интерфейс | Да | Да | Нет |
| LIT Test  (Зайдель) | Да(Android) | Нет | Нет | Да |
| Мой проект | ДА(android) | Да | Да | Нет |

Данные аналоги здесь представлены, так все они так или иначе представляют собой именно систему тестирования, которая имеет подразделение на тестируемого (ученик) и тестирующего (учитель).

## 3.1. MyTest

“MyTest - система программ для создания и проведения компьютерного тестирования, сбора и анализа их результатов”.

“Программа MyTestX работает с десятью типами заданий: одиночный выбор, множественный выбор, установление порядка следования, установление соответствия, указание истинности или ложности утверждений, ручной ввод числа, ручной ввод текста, выбор места на изображении, перестановка букв, заполнение пропусков”.

“Программа состоит из трех модулей: Модуль тестирования (MyTestStudent), Редактор тестов (MyTestEditor) и Журнал тестирования (MyTestServer)”.

Таким образом видно, что данный аналог содержит большинство особенностей, которые предполагаются в нашем приложении, но это только PC программа.

## 3.2 Moodle

Для того, чтобы было возможно пользоваться данным приложением, необходимо быть зарегистрированным в системе Moodle.

## 3.3 Websoft CoursLine

“CourseLab – это мощное средство для создания интерактивных учебных материалов (электронных курсов), предназначенных для использования в сети Интернет, в системах дистанционного обучения, на компакт-диске или любом другом носителе”.

Ключевые особенности: Медиа контент, drag & drop, шаблоны оформления, интеграция с Microsoft Office.

## 3.4 LIT Test

Типы заданий: одиночный выбор, множественный выбор, ответ в краткой форме, ответ в виде «да/нет». Автоматическая проверка тестов.

# Анализ предметной области

## Что такое тест

Тест- это измерительная методика оценки, основанная на стандартизированном наборе заданий и формализованных правилах анализа ответов, дающая в результате объективированную оценку испытуемого на определённой количественной шкале, указывающей на выраженность измеряемого свойства или вероятностное отнесение к определённой, заранее заданной диагностической категории.

Примером такого теста может быть часть А и Б в ЕГЭ.

## Виды тестов

Тесты можно классифицировать по различным признакам:

* По целям — информационные, диагностические, обучающие, мотивационные, аттестационные;
* По процедуре создания — стандартизованные, не стандартизованные;
* По способу формирования заданий — детерминированные, стохастические, динамические;
* По технологии проведения — бумажные, в том числе бумажные с использованием оптического распознавания, натурные, с использованием специальной аппаратуры, компьютерные;
* По форме заданий — закрытого типа, открытого типа, установление соответствия, упорядочивание последовательности;
* По наличию обратной связи — традиционные и адаптивные

В образовании в основном используются традиционные тесты, то есть те которые содержат список вопросов, каждый из которых оценивается в определенное количество баллов. Результат такого тестирования зависит от количества вопросов, на которые был дан правильный ответ. В традиционном тесте оценивание происходит с одинаковыми правилами оценивания ответов для всех испытуемых. Главная цель применения традиционных тестов - установить уровень знаний.

Также можно разделить тесты на:

* Образовательные
* Профессиональные
* Медицинские
* Психологические

Образовательные тесты – это «предметно-ориентированные» тесты знаний. В нём задания относятся к определённым академическим предметам-математике, языку, истории, физике и т.д. Этот вид тестов проводят обычно педагоги-предметники. В результате такого теста испытуемый получает свою оценку или балл за выполненные задания.

Профессиональные тесты – это тесты профессиональных знаний, в которых большую роль играют «кейс-задания» - задачи, описывающие практическое производственные ситуации и предлагающие выбор способа действия в этой ситуации.

Психологические тесты – это тесты, ориентированные на установку и измерение индивидуально-психологических отличий испытуемого. К этому виду относятся такие тесты как: IQ, Тест Сонди, Тест Роршаха.

Созданная мною инструментальная среда направлена на работу с образовательными тестами, но также можно также5 работать и с производственными.

## Виды тестовых заданий

Вот список основных видов заданий

1. Контрольные списки суждений или закрытые тесты с ответами «да-нет»
2. Списки суждений с дробной шкалой согласия, например «да-нет-не знаю»
3. Закрытые вопросы, предполагающие качественно различных ответов, из которые надо выбрать только один(multiple-choice).
4. Закрытые тест-вопросы, допускающие выбор нескольких качественно-различных готовых ответов. Это вопросы со множественным ответом (multiple-response).
5. Открытые тест-вопросы с кратким свободным ответом (в виде числа, в виде одного слова). Некоторой визуальной модификацией является суждения с пропусками (gap-filling).
6. Открытые тест-вопросы с развёрнутым ответом. Этот тип используется в части С ЕГЭ.
7. Задания на ранжирование. Испытуемому предлагается выстроить определённое множество элементов в определённой последовательности.
8. Задания на соответствие: в этом случае испытуемый должен сравнить два множества объектов и установить попарные соответствия для всех элементов.

В ЕГЭ используются все из представленных выше типов заданий, но в ответе они сводятся к трём типам: выбор одного варианта из четырёх (часть А), запись краткого текстового ответа (часть Б), развёрнутый ответ (часть С).

# Решение

## 5.0 Архитектура системы

Электронное тестирование, в отличие от остальных бумажного, предполагает использование компьютерных систем для сбора и анализа результатов.

Система для электронной оценке знаний должна состоять из нескольких обязательных модулей:

* Инструмент для подготовки тестовых заданий (конструктор тестов, только web)
* Тест-плейер(Android & web)
* А также инструмент для проведения тестирования (веб-приложение, приложение-сервер и тд).

## 5.1 Хранение данных

Все данные, такие как логины, пароли, тесты и тд хранятся в базе данных MySQL. БД MySQL одна из самых распространенных баз данных. Был выбор между 3-мя БД:PostgreSQL, SQLite, MySQL. Поводы выбора MySQL:

1. имелся опыт работы в данной БД;
2. SQLite имеет меньший набор инструментов для управления этой БД;
3. MySQL наиболее докуметирована

Рассмотрим систему тестирования данного приложения:



Сайт соединяется с сервером напрямую. В свою очередь Android приложение для взаимодействия с сервером использует API сайта. Из API Android приложение берет информацию о тестах, учениках, вопросах и тд. Для того, чтобы отправить какой-то запрос на сервер используются POST запросы. На сервере происходит обработка запроса и возвращение результата обработки обратно на устройство. Результаты записываются в базу данных.

### 5.1.1 Хранение тестов

XML теста

<?xml version="1.0"?>  
<Test><choice\_count>2</choice\_count>

<edit\_date>2015-08-19T15:42:21</edit\_date>

<id type="integer">85</id>

<name>test</name>

<pub\_date>20150819T15:42:21</pub\_date>

<resource\_uri>/api/v1/tests/85/</resource\_uri>

<teacher type="integer">1</teacher>

<theme>testing</theme>

<visibility type="boolean">True</visibility>

</Test>

### Хранение вопросов

<?xml version="1.0"?>

<Question>

<edit\_date>2015-08-28T02:32:23</edit\_date>

<id type="integer">100</id>

<pub\_date>2015-08-28T02:32:23</pub\_date>

<question\_count type="integer">1</question\_count>

<question\_text>question</question\_text>

<resource\_uri>/api/v1/questions/100/</resource\_uri>

<test type="integer">94</test>

<test1>/api/v1/tests/94/</test1>

<theme>Theme</theme>

<visibility type="boolean">True</visibility>

</Question>

### Хранение результатов теста

<?xml version="1.0"?>

<Result>

<answers\_count>4</answers\_count>

<balls>1</balls>

<id type="integer">20</id>

<quest\_count>1</quest\_count>

<resource\_uri>/api/v1/testresults/20/</resource\_uri>

<right\_count>1</right\_count>

<test type="integer">85</test>

<unright\_count>1</unright\_count>

</Result>

### Хранение данных о учениках

<?xml version="1.0"?>

<Student>

<grade type="integer">1</grade>

<id type="integer">1</id>

<login>s</login>

<name>s</name>

<resource\_uri>/api/v1/students/1/</resource\_uri>

<school type="integer">1</school>

<surname>s</surname>

<user type="integer">1</user>

</Student>

### Хранение данных о учителях

<?xml version=”1.0”?>

<Teacher>

<id type="integer">1</id>

<login>t</login>

<name>t</name>

<resource\_uri>/api/v1/teachers/1/</resource\_uri>

<school type="integer">1</school>

<surname>t</surname>

<user type="integer">1</user>

</Teacher>

## 5.2. Структура данных

### Взаимодействие классов

На приведенной ниже диаграмме визуализировано взаимодействие между вопросами, ответами на них и тестами, в состав которых входят данные вопросы:



Ниже показана связь учителей и учеников с соответствующими им школами и классами:

Каждый результат теста привязан к конкретному ученику и тесту:



Таким образом, полная схема связей между классами выглядит так:



### Типы тестов

* Обычный
* На время
* Со случайными вопросами из списка

При создании теста учитель сначала вводит его название, тему, для какого он класса, а также название предмета тестирования. Затем он переходит к заполнению теста вопросами. При создании вопроса тестирующий может добавить недостающие варианты ответов, а также добавить картинку к вопросу.

Правильных ответов может быть от 0 до количества вариантов ответов.

#### Обычный тест

У теста фиксированные вопросы.

#### Тест на время

При создании теста учитель указывает время на прохождение теста в секундах. Ученику, для начала прохождения теста, необходимо нажать на соответствующую кнопку, после чего ему будет доступно выполнение теста.

После истечения времени, тест будет закрыт для редактирования.

#### Тест со случайными вопросами

При создании теста, учитель указывает количество вопросов, которые должны быть показано ученикам. После этого учитель заполняет тест вопросами. Ученикам будут предоставлены случайные вопросы с указанным количеством вопросов из данного теста.

### Типы вопросов

В нашем приложении существует два типа вопросов: с картинкой и без.

При создании вопроса к тесту учитель может выбрать изображение к вопросу, загрузив его с устройства. Также при создании вопроса учитель может добавить варианты ответа в необходимом количестве.

## Прохождение теста

Перед учеником не открывается тест до тех пор, пока он не нажмет соответствующую кнопку для начала тестирования. Если же он начал тестирование и не закончил его выполнение, то ученику будет засчитано ноль правильных ответов. При прохождении теста тестируемый может выбрать несколько вариантов ответа. По окончанию процесса ученик нажимает на кнопку отправления результатов, далее этот тест отмечается как пройденный для данного ученика, и он не может пройти его снова.

## Алгоритм работы сервера

Связь сервера с устройством осуществляется через GET/POST запросы.

1.Сайт: использует GET запросы для получения, POST – для отправления данных на сервер.

2.Android приложение: происходит обмен данными через POST запросы. На сервер отправляется запрос и возвращается ответ в виде строки. Сначала использовался GET метод с помощью API сайта, но для ускорения работы предпочтение было отдано POST запросам.

Например, запрос , состоящий из пары(ключ, значение) для определения доступных тестов ученика(available\_tests, student\_id). Результат запроса: (“test\_id1, test\_id2….”).

## Программная реализация

### Реализация Python

Была поставлена задача не только разработать систему тестирования, но также и изучить программирование, поэтому варианты в виде CMS(Content Management System) были отброшены. Был выбор между 4-мя языками веб программирования: PHP, C#, Python, Java. Python был выбран по следующим причинам:

1)относительная новая платформа Django постепенно внедрялась в различные веб проекты;

2)Python имеет отличный от C# и Java синтаксис, а также Python является скриптовым некомпилируемым языком – все это повлияло на выбор ЯП, так как это еще не было пройдено, хотелось узнать новое в программировании.

#### API сайта

API (Application Programming Interface) — это интерфейс программирования, интерфейс создания приложений. Если говорить более понятным языком, то API — это готовый код для упрощения жизни программисту. API создавался для того, чтобы программист реально мог облегчить задачу написания того или иного приложения благодаря использованию готового кода (например, функций). Для работы с API в Django я использовал библиотеку TastyPie. С помощью этой библиотеки можно ограничить выводимую информацию по таким параметрам, как:

1. Класс для вывода объектов(например, Teacher)
2. Метод(например, GET)
3. Можно изменить вывод определенного параметра объекта

Затем нужно в определенном файле подключить библиотеку, создать ее объект, указать, какие API были созданы. На выходе получим XML с указанными параметрами. Данные XML можно открыть в браузере или загрузить на устройство для дальнейшей обработки.

Пример создания API для вывода списка учителей:

class MyModelResource(ModelResource):

user = fields.ForeignKey(UserResource, 'user', null=True, blank=True)

school = fields.ForeignKey(SchoolResource, 'school', null=True, blank=True)

class Meta:

queryset = Teacher.objects.all()

resource\_name = 'teachers'

allowed\_methods = ['get']

serializer = Serializer(formats=['json', 'jsonp', 'xml', 'yaml', 'html', 'plist'])

Создаем объект API:

from api.resources import MyModelResource

from tastypie.api import Api

v1\_api = Api(api\_name='v1')

v1\_api.register(MyModelResource())

#### Работа с API на Android

//Заполнить Горбачеву Александру

### Среды разработки

Для программирования на Python была выбрана среда разработки JetBrains PyCharm. Также рассматривалась среда Eclipse. Но она не была выбрана, потому что не имеет автодополнения выражений(основной критерий), удобным дизайном в отличие от первой IDE.

Android приложение написано в среде JetBrains IntellijIdea. Данная среда схожа с Android Studio, так как вторая основана на программном обеспечении первой. Главной причиной выбора Intllij Idea является ее большая стабильность по сравнению с Android Studio. Была рассмотрена среда Eclipse, но не использована так как не обладает некоторым функционалом, таким как автодополнением выражений, удобным дизайном в отличие от аналогов.

## Ход работы

Изначально связь между Adnroid приложением и сервером предполагалась за счет API. Использование API довольно тяжело реализовывалось в разработке под смартфон. Поэтому позже, когда появилась необходимость аутентифицировать пользователей, на смену API пришли POST запросы. Они проще и в использовании.

На финальной стадии разработки начали использоваться ajax технологии, но широкого применения они получить не успели.

### Проблемы и решения

1. Прежде чем записать изменения в базу данных Django генерирует файл “migrations.py «номер изменения»” с изменениями. Он генериуется командой “makemigrations”. Затем командой “migrate” изменения записываются окончательно в БД. После непродолжительного использования данной системы возникают проблемы, а именно: при попытке отправить изменения на сервер, Django сообщает, что, допустим, такая таблица в БД уже есть и создать ее не получится, таким образом отменяются все последние изменения. Чтобы исправить данную ошибку, приходилось открывать файл “migrations” с соответствующим номером изменения, находить часть кода, которая не давала успешно завершиться отправке данных и либо удалять ее, либо изменять. На начальном этапе программирования данная неполадка потребовала около трех дней разработки, чтобы научиться обходить эту ошибку.
2. PyCharm хоть и является одной из лучших IDE для веб разработки, данное приложение имеет большой недостаток в отладчике. Во-первых, он работает только при условии запуска сайта так, чтобы к нему имел доступ только компьютер, на котором совершен запуск. Соответственно отладка Android приложения была возможна только благодаря ручному выводу определенного значения в командную строку/терминал. Также отладчик PyCharm не справляется с Javascript, чего сильно не хватало. Система отладки в данной ситуации схожа с отладкой приложения для смартфона – необходимое значение переменной выводилось на экран командой “alert”.
3. В Android приложении подключение к серверу невозможно без использования AsyncTask (синхронных процессов). Потребовалось много времени, чтобы научиться ими пользоваться.

# Результат

В результате работы была разработана система для проведения электронного тестирования на Android устройствах и в web интерфейсе.

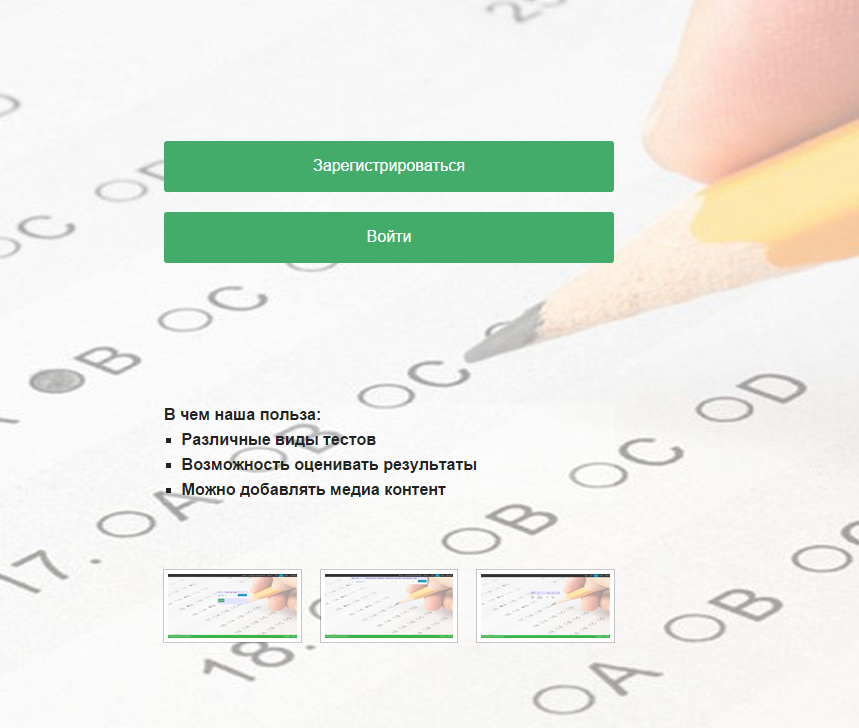
Данная система имеет инструменты для тестируемого и тестирующего:

* + Ученик – Приложение Android и web сайт для прохождения тестирования
  + Учитель – редактор тестов, выставление оценки, просмотр тестов.

Пример работы сайта:

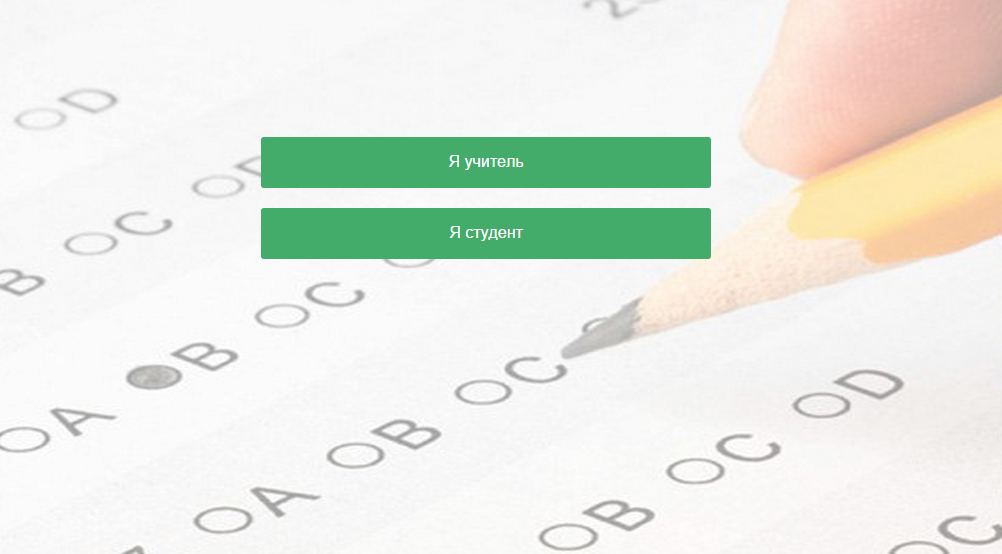
Учитель

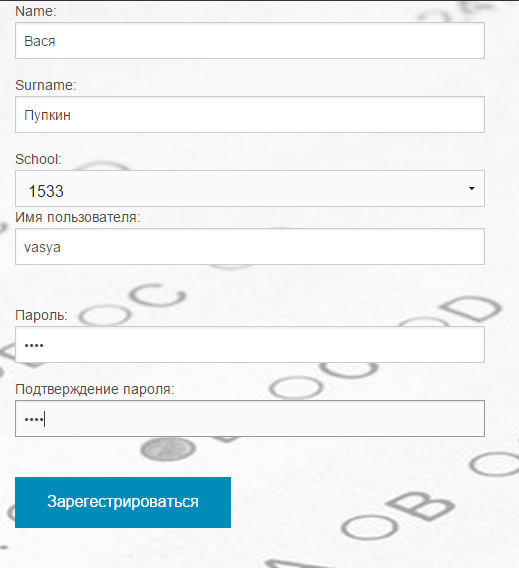
1. (+ ученик) Вводная страница

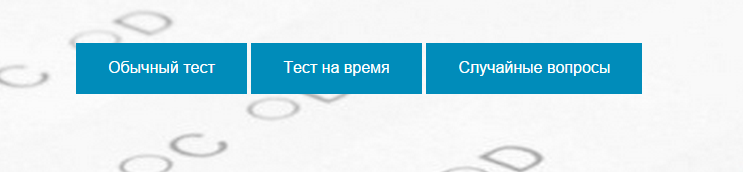




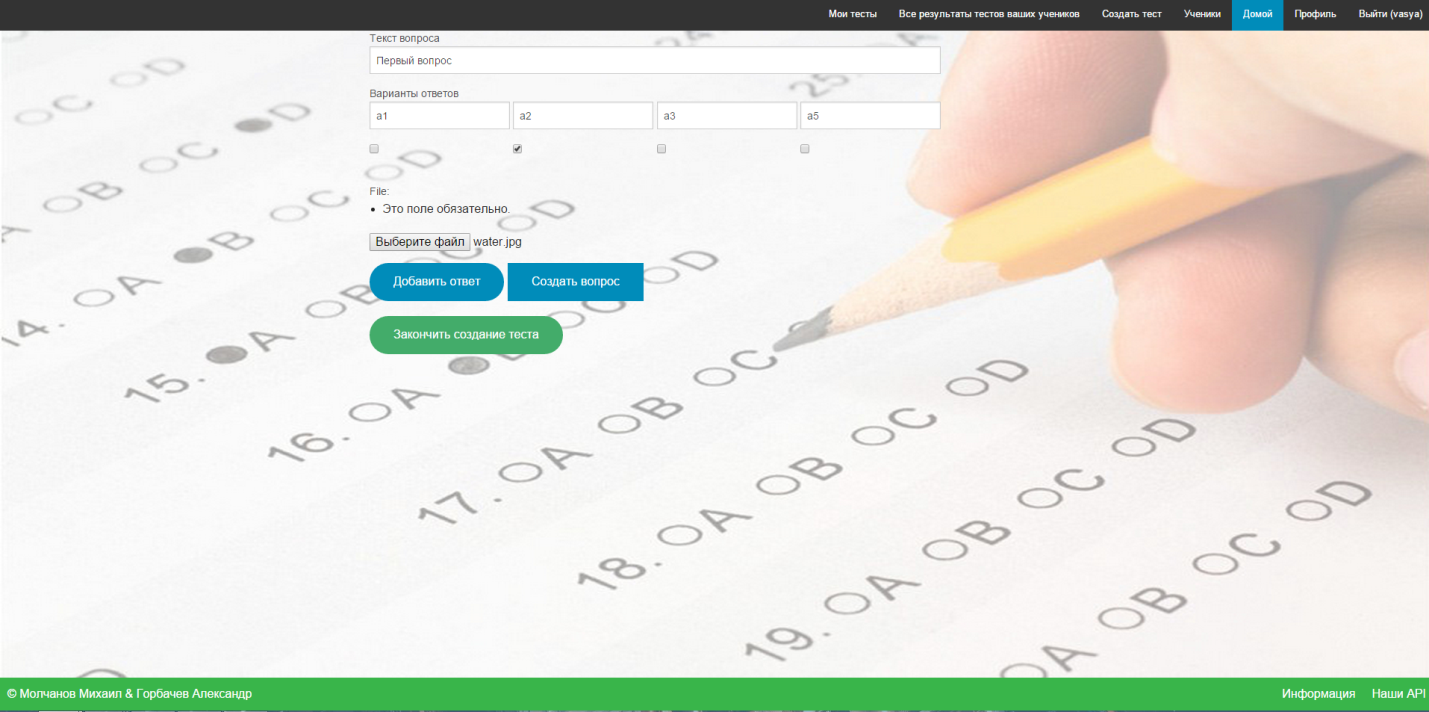
1. (+ ученик) Регистрация

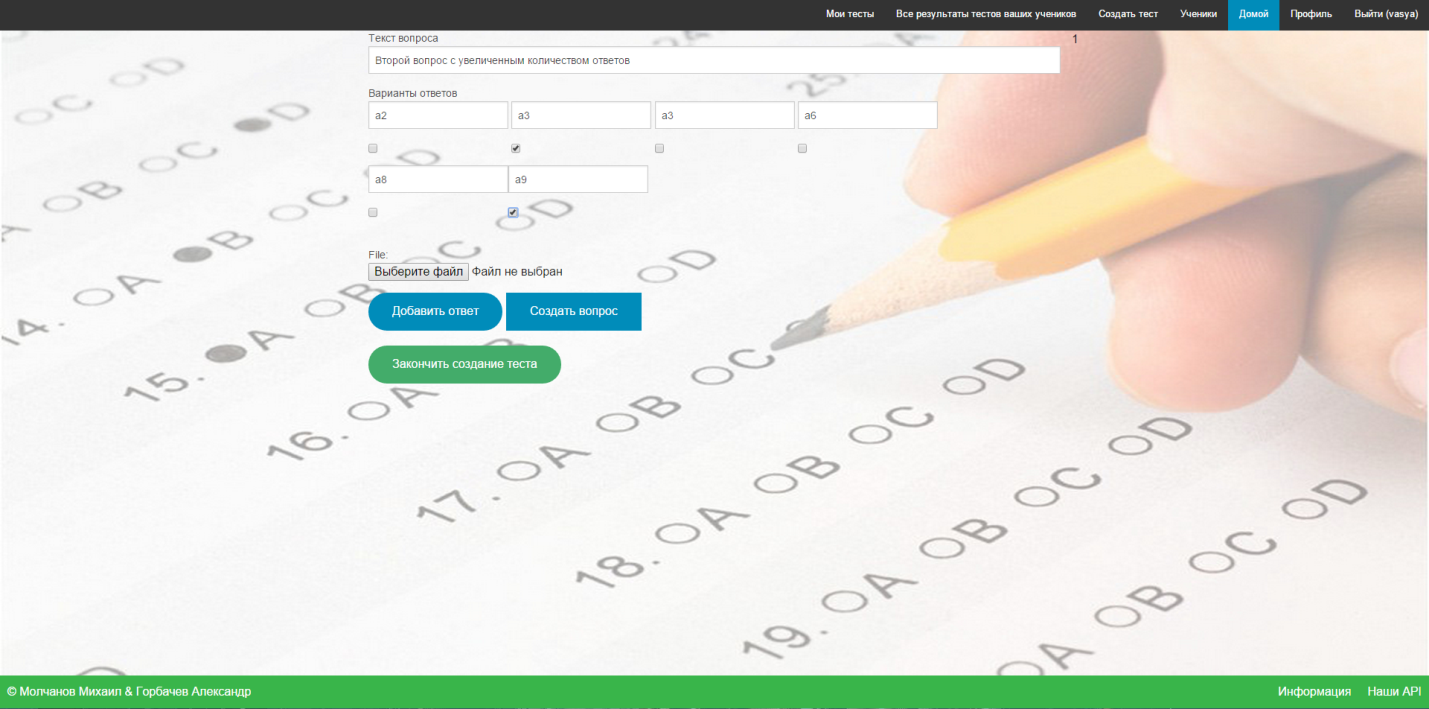


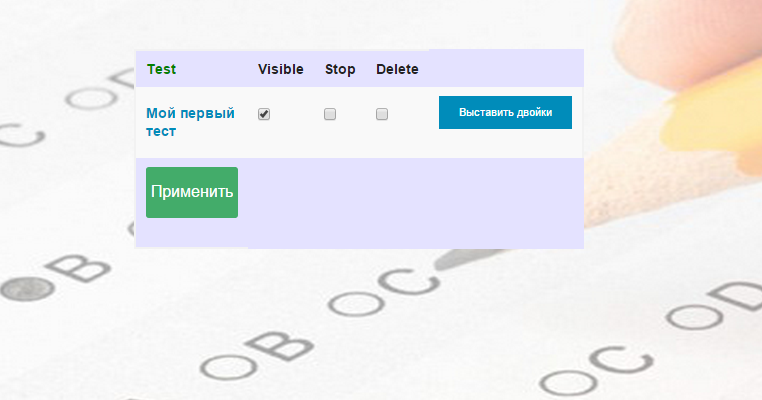


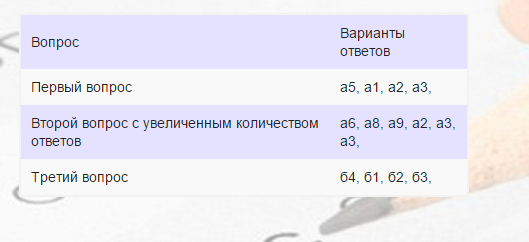
3)Выбор типа теста

4)Настройка теста

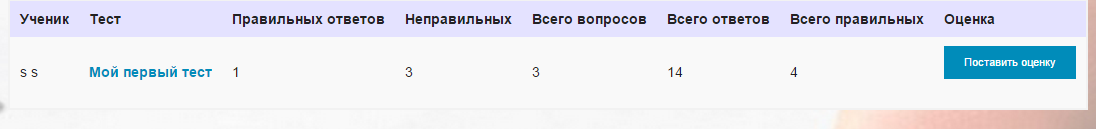
5)Заполнение теста



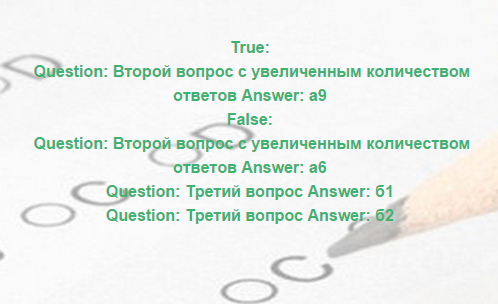
6)Просмотр тестов и управление ими

7)Детализация теста

8)Просмотр результатов и выставление оценок

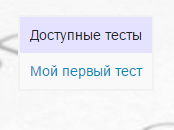


9) Детализация результата

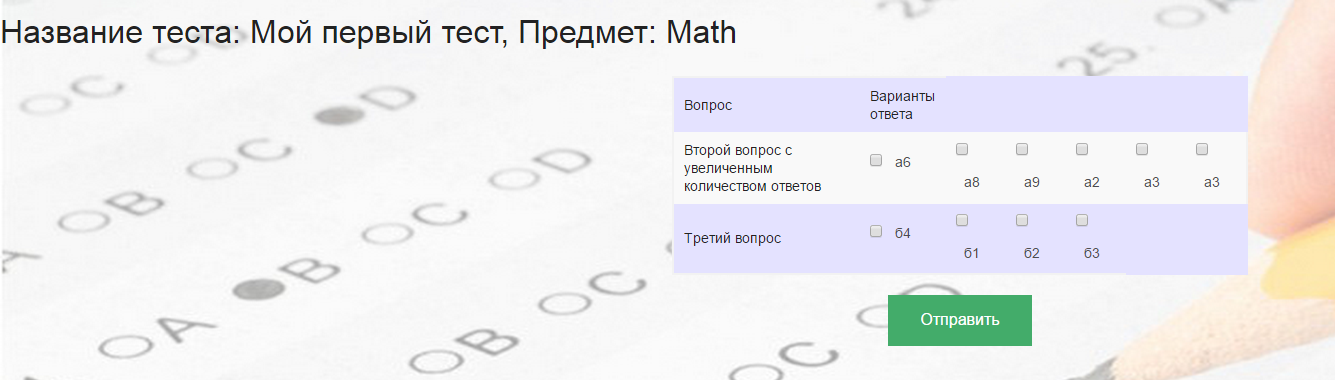


Ученик

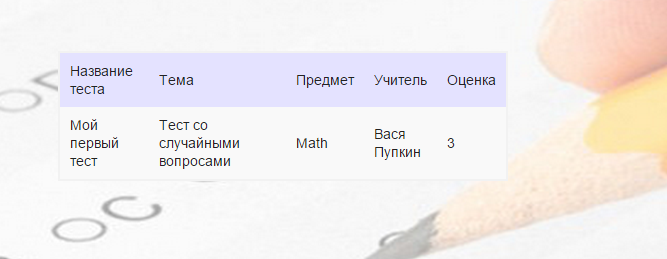
1. Просмотр доступных тестов



2)Прохождение теста

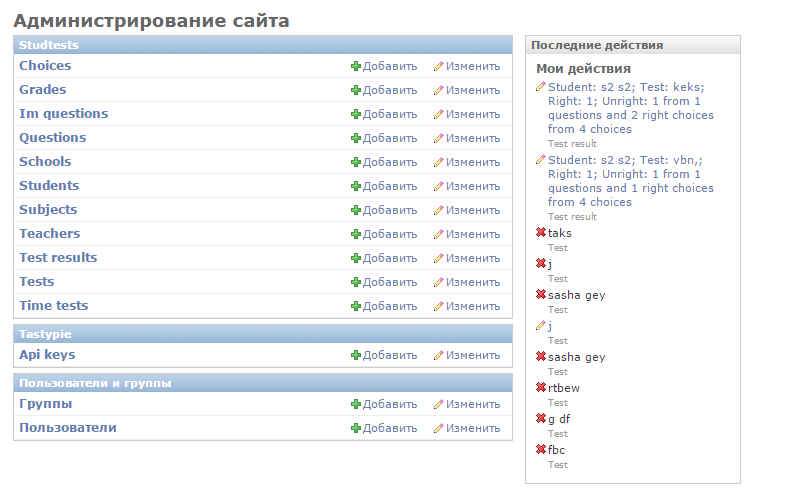


3)Просмотр результатов



Администратор:

Администрирование сайта



# Выводы

На данный момент тема электронного тестирования очень актуальна. И вскоре подобные системы тестирования будут сильно распространены.

Перспективы разработки проекта:

* Фильтрация реальных пользователей
* Приближение функционала Android приложения к функционалу сайта
* Реализовать еще несколько видов теста
* Большее применение ajax технологий
* Добавить функцию массового добавления изображений
* Добавить функцию выбора картинки в качестве ответа
* Дополнить возможности личных настроек

# Список литературы

[habrahabr.ru](file:///C:\Users\Леново\Documents\habrahabr.ru) – множество статей по android/django разработке

[startandroid.ru](file:///C:\Users\Леново\Documents\startandroid.ru) – уроки по android разработке

[developer.alexanderklimov.ru](file:///C:\Users\Леново\Documents\developer.alexanderklimov.ru) – android разработка

[djbook.ru](file:///C:\Users\Леново\Documents\djbook.ru) – django учебник

[django-tastypie.readthedocs.org/](http://django-tastypie.readthedocs.org/) - учебник по tastypie библиотеке

[stackoverflow.com](http://stackoverflow.com/) - англоязычный форум

[ru.stackoverflow.com](http://stackoverflow.com/) - русскоязычный форум

[www.cyberforum.ru](http://www.cyberforum.ru/) - русскоязычный форум

[toster.ru](file:///C:\Users\Леново\Documents\toster.ru) – русскоязычный форум

# Приложения

## 9.1 Python

**Models.py**

# -\*- coding: utf-8 -\*-

from django.contrib.auth.tests.custom\_user import CustomUser

from django.db import models

from multiprocessing.managers import public\_methods

from django.db.models import Q, ImageField

from django.db.models.fields.related import ForeignKey

from django import forms

from django.template.defaultfilters import length, default

from django.contrib.auth.models import User, UserManager

from django.contrib.auth.forms import UserCreationForm

import datetime

import os

import PIL

def get\_image\_path(instance, filename):

return os.path.join('photos', str(instance.id), filename)

class School(models.Model):

school = models.CharField(max\_length=200, null=True)

def \_\_unicode\_\_(self):

return self.school

class Grade(models.Model):

grade = models.CharField(max\_length=3)

def \_\_unicode\_\_(self):

return self.grade

class Subject(models.Model):

subject = models.CharField(max\_length=200)

school = models.ForeignKey(School, null=True)

def \_\_unicode\_\_(self):

return self.subject

class Teacher(models.Model):

user = models.OneToOneField(User, null=True)

name = models.CharField(max\_length=200)

surname = models.CharField(max\_length=200)

login = models.CharField(max\_length=200)

school = models.ForeignKey(School, null=True)

def \_\_unicode\_\_(self):

return self.name + ' ' + self.surname

class Test(models.Model):

name = models.CharField(max\_length=200, null=True)

pub\_date = models.DateTimeField('date published', default=datetime.datetime.now())

edit\_date = models.DateTimeField('date edited', default=datetime.datetime.now())

choice\_count = models.CharField(max\_length=3, null=True)

subject = models.ForeignKey(Subject, null=True)

teacher = models.ForeignKey(Teacher, null=True)

school = models.ForeignKey(School, null=True)

grade = models.ForeignKey(Grade)

theme = models.CharField(max\_length=200, null=True)

visibility = models.BooleanField(default=False)

def \_\_unicode\_\_(self):

return self.name

class TimeTest(Test):

time = models.CharField(max\_length=3)

class RndTest(Test):

qcount = models.CharField(max\_length=3)

class Question(models.Model):

question\_text = models.CharField(max\_length=200)

pub\_date = models.DateTimeField('date published', default=datetime.datetime.now())

edit\_date = models.DateTimeField('date edited', default=datetime.datetime.now())

subject = models.ForeignKey(Subject)

teacher = models.ForeignKey(Teacher, null=True)

school = models.ForeignKey(School)

grade = models.ForeignKey(Grade)

theme = models.CharField(max\_length=200)

visibility = models.BooleanField(default=False)

test = models.ForeignKey(Test, null=True)

def \_\_unicode\_\_(self):

return self.question\_text

class ImQuestion(Question):

image = models.ImageField()

class Choice(models.Model):

choice\_text = models.CharField(max\_length=200)

question = models.ForeignKey(Question, null=True)

right\_choice = models.BooleanField(default=False)

def \_\_unicode\_\_(self):

return 'Question: ' + self.question.question\_text + ' Answer: ' + self.choice\_text

class Student(models.Model):

user = models.OneToOneField(User, null=True)

name = models.CharField(max\_length=200, null=True)

surname = models.CharField(max\_length=200, null=True)

login = models.CharField(max\_length=200, null=True)

school = models.ForeignKey(School, null=True)

grade = models.ForeignKey(Grade, null=True)

question\_list = models.ManyToManyField(Test, blank=True, null=True)

def set\_password(self, raw\_password):

self.user.set\_password(raw\_password)

def \_\_unicode\_\_(self):

return self.name + ' ' + self.surname

class TestResult(models.Model):

student = models.ForeignKey(Student)

test = models.ForeignKey(Test)

balls = models.CharField(max\_length=5)

quest\_count = models.CharField(max\_length=5, null=True)

right\_choices = models.ManyToManyField(Choice, related\_name="right")

unright\_choices = models.ManyToManyField(Choice, related\_name="false")

right\_count = models.CharField(max\_length=3, null=True)

unright\_count = models.CharField(max\_length=3, null=True)

answers\_count = models.CharField(max\_length=3, null=True)

def \_\_unicode\_\_(self):

return 'Student: ' + str(self.student) + '; Test: ' + str(self.test) + '; Right: ' + str(

len(self.right\_choices.all())) \

+ '; Unright: ' + str(len(self.unright\_choices.all())) + ' from' + ' ' + self.quest\_count \

+ ' questions and ' + str(self.test.choice\_count) + ' right choices' + ' from ' + str(

4 \* len(self.test.question\_set.all())) \

+ ' choices'

class RegistrTeacherForm(UserCreationForm):

class Meta:

model = Teacher

fields = ['name', 'surname', 'school']

class RegistrStudentForm(UserCreationForm):

class Meta:

model = Student

fields = ['name', 'surname', 'school', 'grade']

class UploadFileForm(forms.Form):

file = forms.FileField()

**loginsys/views.py**

# -\*- coding: utf-8 -\*-

from django.core.urlresolvers import reverse

from django.http import HttpResponseRedirect

from django.shortcuts import render, render\_to\_response, redirect

from django.contrib import auth

from django.core.context\_processors import csrf

from django.contrib.auth.forms import UserCreationForm

from studtests.models import RegistrTeacherForm, RegistrStudentForm, Subject, School, Student, Teacher, Grade

from django.contrib.auth.models import User

import studtests

def login(request):

args = {}

args.update(csrf(request))

if request.POST:

username = request.POST.get('username', '')

password = request.POST.get('password', '')

user = auth.authenticate(username=username, password=password)

if user is not None:

auth.login(request, user)

request.session['username'] = username

teachers = Teacher.objects.all()

students = Student.objects.all()

for teacher in teachers:

if teacher.login == username:

request.session['usertype'] = "t"

request.session['questions'] = []

for student in students:

if student.login == username:

request.session['usertype'] = "s"

return redirect('/')

else:

args['login\_error'] = "Пользователь не найден"

return render\_to\_response('loginsys/login.html', args)

else:

return render\_to\_response('loginsys/login.html', args)

def logout(request):

auth.logout(request)

return redirect('/')

def registrate(request):

args = {}

args.update(csrf(request))

args['form'] = UserCreationForm()

if request.session['usertype'] == "t":

args['form1'] = RegistrTeacherForm()

else:

args['form1'] = RegistrStudentForm()

if request.session['usertype'] is not None:

if request.POST.get("reg"):

newuser\_form = UserCreationForm(request.POST)

newteacher\_form = RegistrTeacherForm(request.POST)

newstudent\_form = RegistrStudentForm(request.POST)

if newuser\_form.is\_valid() and (newteacher\_form.is\_valid() or newstudent\_form.is\_valid):

school = School.objects.get(id=request.POST['school'])

newuser\_form.save()

new\_user = auth.authenticate(username=newuser\_form.cleaned\_data['username'],

password=newuser\_form.cleaned\_data['password2'])

request.session['username'] = newuser\_form.cleaned\_data['username']

auth.login(request, new\_user)

if request.session['usertype'] == "t":

newteacher = Teacher(user=User.objects.get(username=newuser\_form.cleaned\_data['username']),

school=school, name=request.POST['name'], login=request.POST['username'],

surname=request.POST['surname'])

newteacher.save()

elif request.session['usertype'] == "s":

newstudent = Student(grade=Grade.objects.get(id=request.POST['grade']),

user=User.objects.get(username=newuser\_form.cleaned\_data['username']),

school=school, name=request.POST['name'], login=request.POST['username'],

surname=request.POST['surname'])

newstudent.save()

return redirect('/')

else:

if request.session['usertype'] == "t":

args['form1'] = RegistrTeacherForm(request.POST)

else:

args['form1'] = RegistrStudentForm(request.POST)

return render\_to\_response('loginsys/registrate.html', args)

def user\_choose(request):

args = {}

args.update(csrf(request))

if request.POST:

if request.POST.get("teacher"):

request.session['usertype'] = "t"

else:

request.session['usertype'] = "s"

return HttpResponseRedirect(reverse('registrate'))

return render\_to\_response('loginsys/action.html', args)

**loginsys/action.html**

{% extends "studtests/base.html" %}

{% block action %}

<div id="center">

<form action="/auth/action/" method="post">

{% csrf\_token %}

<input class="button radius expand success" type="submit" name="teacher" id="teacher" value="Я учитель">

<input class="button radius expand success" type="submit" name="student" id="student" value="Я студент">

</form>

</div>

{% endblock %}

**Loginsys/lohin.html**

{% extends "studtests/base.html" %}

{% block login %}

<div class="large-offset-3 small-4 columns">

<form action="/auth/login/" method="post">

{% csrf\_token %}

<label for="username">Имя пользователя</label>

<input type="text" name="username" id="username">

<label for="password">Пароль</label>

<input type="password" name="password" id="password">

{% if login\_error %}

<label class="error">{{ login\_error }}</label>

{% endif %}

<input class="button" type="submit" value="Войти">

</form>

</div>

{% endblock %}

**Loginsys/registrate.html**

{% extends "studtests/base.html" %}

{% block login %}

<div class="row">

<div class="small-8 large-6 columns">

<form action="/auth/registrate/" method="post">

{% csrf\_token %}

{{ form1 }}

<input class="button" name="reg" type="submit" value="Зарегестрироваться">

</form>

</div>

</div>

{% endblock %}

Loginsys/urls.py

from django.conf.urls import patterns, url

from loginsys import views

urlpatterns = patterns('',

url(r'^login/', views.login, name='login'),

url(r'^logout/', views.logout, name='logout'),

url(r'^registrate', views.registrate, name='registrate'),

url(r'^action',views.user\_choose, name='action')

)

**Resources.py**

from django.contrib.auth.models import User

from tastypie.authentication import BasicAuthentication, ApiKeyAuthentication, MultiAuthentication

from tastypie.authorization import DjangoAuthorization

from tastypie.resources import ModelResource

from studtests.models import Teacher, Subject, Test, Student, School, Grade, Choice, Question, TestResult, ImQuestion

from tastypie.authentication import Authentication

from tastypie import fields

from tastypie.serializers import Serializer

class SchoolResource(ModelResource):

class Meta:

limit = 0

queryset = School.objects.all()

resource\_name = 'schools'

allowed\_methods = ['get']

serializer = Serializer(formats=['json', 'jsonp', 'xml', 'yaml', 'html', 'plist'])

class GradeResource(ModelResource):

class Meta:

limit = 0

queryset = Grade.objects.all()

resource\_name = 'grades'

allowed\_methods = ['get']

serializer = Serializer(formats=['json', 'jsonp', 'xml', 'yaml', 'html', 'plist'])

class UserResource(ModelResource):

class Meta:

limit = 0

queryset = User.objects.all()

resource\_name = 'users'

excludes = ['is\_superuser']

allowed\_methods = ['get']

serializer = Serializer(formats=['json', 'jsonp', 'xml', 'yaml', 'html', 'plist'])

class MyModelResource(ModelResource):

def dehydrate(self, bundle):

try:

school = School.objects.filter(id=bundle.obj.school.id)

user = User.objects.filter(id=bundle.obj.school.id)

bundle.data['user'] = user[0].id

bundle.data['school'] = school[0].id

except School.DoesNotExist:

pass

except User.DoesNotExist:

pass

return bundle

user = fields.ForeignKey(UserResource, 'user', null=True, blank=True)

school = fields.ForeignKey(SchoolResource, 'school', null=True, blank=True)

class Meta:

limit = 0

queryset = Teacher.objects.all()

resource\_name = 'teachers'

allowed\_methods = ['get']

serializer = Serializer(formats=['json', 'jsonp', 'xml', 'yaml', 'html', 'plist'])

class TestModelResource(ModelResource):

def dehydrate(self, bundle):

try:

teacher = Teacher.objects.filter(id=bundle.obj.teacher.id)

bundle.data['teacher'] = teacher[0].id

except Teacher.DoesNotExist:

pass

return bundle

teacher = fields.ForeignKey(MyModelResource, 'teacher', null=True)

class Meta:

limit = 0

queryset = Test.objects.all()

resource\_name = 'tests'

allowed\_methods = ['get']

serializer = Serializer(formats=['json', 'jsonp', 'xml', 'yaml', 'html', 'plist'])

class QuestionResource(ModelResource):

def dehydrate(self, bundle):

try:

test = Test.objects.filter(id=bundle.obj.test.id)

count = 0

for x in Question.objects.all():

test2 = Test.objects.get(pk=test[0].id)

if x.test == test2:

count += 1

bundle.data['test'] = test[0].id

bundle.data['question\_count'] = count

except Test.DoesNotExist:

pass

return bundle

test1 = fields.ForeignKey(TestModelResource, 'test', null=True)

class Meta:

limit = 0

queryset = Question.objects.all()

resource\_name = 'questions'

allowed\_methods = ['get']

serializer = Serializer(formats=['json', 'jsonp', 'xml', 'yaml', 'html', 'plist'])

class ChoiceResource(ModelResource):

def dehydrate(self, bundle):

try:

question = Question.objects.filter(id=bundle.obj.question.id)

bundle.data['question'] = question[0].id

bundle.data['test'] = question[0].test.id

except Test.DoesNotExist:

pass

return bundle

question = fields.ForeignKey(QuestionResource, 'question', null=True)

class Meta:

limit = 0

queryset = Choice.objects.all()

resource\_name = 'choices'

allowed\_methods = ['get']

serializer = Serializer(formats=['json', 'jsonp', 'xml', 'yaml', 'html', 'plist'])

class TestResultResource(ModelResource):

def dehydrate(self, bundle):

try:

test = Test.objects.filter(id=bundle.obj.test.id)

bundle.data['test'] = test[0].id

except Test.DoesNotExist:

pass

return bundle

class Meta:

queryset = TestResult.objects.all()

resource\_name = 'testresults'

allowed\_methods = ['get']

serializer = Serializer(formats=['json', 'jsonp', 'xml', 'yaml', 'html', 'plist'])

class StModelResource(ModelResource):

def dehydrate(self, bundle):

try:

grade = Grade.objects.filter(id=bundle.obj.grade.id)

school = School.objects.filter(id=bundle.obj.school.id)

user = User.objects.filter(id=bundle.obj.school.id)

bundle.data['user'] = user[0].id

bundle.data['school'] = school[0].id

bundle.data['grade'] = grade[0].id

except Test.DoesNotExist:

pass

except Grade.DoesNotExist:

pass

return bundle

user = fields.ForeignKey(UserResource, 'user', null=True)

school = fields.ForeignKey(SchoolResource, 'school', null=True)

grade = fields.ForeignKey(GradeResource, 'grade', null=True)

class Meta:

limit = 0

queryset = Student.objects.all()

resource\_name = 'students'

allowed\_methods = ['get']

serializer = Serializer(formats=['json', 'jsonp', 'xml', 'yaml', 'html', 'plist'])

class ImquestionsResource(ModelResource):

class Meta:

limit = 0

queryset = ImQuestion.objects.all()

resource\_name = 'imquestions'

allowed\_methods = ['get']

serializer = Serializer(formats=['json', 'jsonp', 'xml', 'yaml', 'html', 'plist'])

**std/urls.py**

from django.conf.urls import patterns, include, url

from django.contrib import admin

from api.resources import MyModelResource, TestModelResource, UserResource, StModelResource, SchoolResource, \

GradeResource, QuestionResource, ChoiceResource, TestResultResource, ImquestionsResource

from tastypie.api import Api

v1\_api = Api(api\_name='v1')

v1\_api.register(MyModelResource())

v1\_api.register(TestModelResource())

v1\_api.register(UserResource())

v1\_api.register(StModelResource())

v1\_api.register(SchoolResource())

v1\_api.register(GradeResource())

v1\_api.register(QuestionResource())

v1\_api.register(ChoiceResource())

v1\_api.register(TestResultResource())

v1\_api.register(ImquestionsResource())

urlpatterns = patterns('',

url(r'^admin/', include(admin.site.urls)),

url(r'^studtests/', include('studtests.urls', namespace="studtests")),

url(r'^auth/', include('loginsys.urls')),

url(r'^api/', include(v1\_api.urls)),

url(r'^api-auth/', include('rest\_framework.urls', namespace='rest\_framework')),

url(r'^', include('studtests.urls', namespace="studtests")),

)

**Std/settings.py**

MEDIA\_URL = '/images/'

MEDIA\_ROOT = 'C:\\std\\studtests\\static\\studtests\\images'

DEBUG = True

TEMPLATE\_DEBUG = True

TEMPLATE\_DIRS = [os.path.join(BASE\_DIR, 'templates')]

ALLOWED\_HOSTS = []

# Application definition

INSTALLED\_APPS = (

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'studtests',

'loginsys',

'rest\_framework',

'tastypie',

'lxml',

'PIL'

)

REST\_FRAMEWORK = {

'DEFAULT\_PERMISSION\_CLASSES': ('rest\_framework.permissions.IsAdminUser',),

'PAGINATE\_BY': 10

}

MIDDLEWARE\_CLASSES = (

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

'django.middleware.common.CommonMiddleware',

'django.middleware.csrf.CsrfViewMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware',

'django.contrib.auth.middleware.SessionAuthenticationMiddleware',

'django.contrib.messages.middleware.MessageMiddleware',

'django.middleware.clickjacking.XFrameOptionsMiddleware',

)

ROOT\_URLCONF = 'std.urls'

WSGI\_APPLICATION = 'std.wsgi.application'

# Database

# https://docs.djangoproject.com/en/1.7/ref/settings/#databases

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'std',

'USER': 'mike',

'PASSWORD': '188227',

'HOST': '127.0.0.1',

'PORT': '',

}

}

# Internationalization

# https://docs.djangoproject.com/en/1.7/topics/i18n/

LANGUAGE\_CODE = 'ru-RU'

LANGUAGES = (

('ru', 'Russian'),

('en', 'English'),

)

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_L10N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/1.7/howto/static-files/

STATIC\_URL = '/static/'

**Studtests/urls.py**

from django.conf.urls import patterns, url

from std import settings

from studtests import views

urlpatterns = patterns('',

url(r'^$', views.index, name='index'),

url(r'^images/(?P<path>.\*)$', views.image, name='image'),

url(r'^subjects/$', views.subjects, name='subjects'),

url(r'^(?P<test\_id>\d+)/$', views.detail, name='detail'),

url(r'^subjects/(?P<subject\_id>\d+)/$', views.subject\_test, name='subject\_test'),

url(r'^(?P<question\_id>\d+)/results/$', views.results, name='results'),

url(r'^(?P<test\_id>\d+)/vote/$', views.vote, name='vote'),

url(r'^tests/(?P<res>\d+)/$', views.test\_res, name='test\_res'),

# url(r'^tests/(?P<subject\_id>\d+)$', views.Tests, name='test'),

url(r'^create\_test/$', views.create, name='create\_test'),

url(r'^lookresults/$', views.lookresults, name='lookresults'),

url(r'^lookresults/(?P<testres\_id>\d+)/$', views.lookresult, name='testresult'),

url(r'^students/$', views.students, name='students'),

url(r'^tests/$', views.all\_tests, name='tests'),

url(r'^apis/$', views.apis, name='apis'),

url(r'^students/(?P<student\_id>\d+)/student\_results/$', views.student\_results,

name='student\_results'),

url(r'^info/$', views.info, name='info'),

url(r'^mytests/$', views.teachertests, name='teachertests'),

url(r'^mytests/(?P<test\_id>\d+)/$', views.teachertest, name='teachertest'),

url(r'^post/$', views.gettrfromandr, name='gettrfromandr'),

url(r'^raitings/$', views.raitings, name='raitings'),

url(r'^profile/$', views.profile, name='profile'),

url(r'^ajtry/$', views.ajresp, name='ajaxtry'),

)

**Studtests/admin.py**

from django.contrib import admin

from django.contrib.auth.admin import UserAdmin

from django.contrib.auth.models import User

from studtests.models import Question, Choice, Subject, School,Student, Teacher, Test, TestResult, Grade, TimeTest, ImQuestion

class SubjectAdmin(admin.ModelAdmin):

fields = ['subject', 'school']

class ChoiceInline(admin.TabularInline):

model = Choice

extra = 3

class QuestionAdmin(admin.ModelAdmin):

fieldsets = [

(None, {'fields': ['question\_text']}),

('Date information', {'fields': ['pub\_date', 'edit\_date'], 'classes': ['collapse']}),

('Subject', {'fields': ['subject']}),

('School', {'fields': ['school']}),

('Grade', {'fields': ['grade']}),

('Theme', {'fields': ['theme']}),

('Teacher', {'fields': ['teacher']}),

('Visible', {'fields': ['visibility']}),

('Test', {'fields': ['test']}),

]

inlines = [ChoiceInline]

list\_display = ('question\_text', 'teacher', 'school', 'pub\_date', 'edit\_date')

class StudentInline(admin.StackedInline):

model = Student

can\_delete = False

verbose\_name\_plural = 'student'

class UserAdmin(UserAdmin):

inlines = (StudentInline, )

admin.site.unregister(User)

admin.site.register(User, UserAdmin)

admin.site.register(Question, QuestionAdmin)

admin.site.register(Teacher)

admin.site.register(Choice)

admin.site.register(School)

admin.site.register(Subject, SubjectAdmin)

admin.site.register(Student)

admin.site.register(Test)

admin.site.register(TestResult)

admin.site.register(Grade)

admin.site.register(TimeTest)

admin.site.register(ImQuestion)

**students/views.py**

import random

from django.shortcuts import render, get\_object\_or\_404, redirect

from django.http import HttpResponse, Http404

from django.views.decorators.csrf import csrf\_exempt

from django.contrib import auth

from studtests.models import Question, Teacher, Subject, Choice, Test, Student, TestResult, Grade, TimeTest, \

ImQuestion, UploadFileForm, RndTest

def logined(request):

if 'username' in request.session and 'usertype' in request.session:

return [request.session['username'], request.session['usertype']]

else:

return None

"""Index page"""

def index(request):

if not 'username' in request.session:

return render(request, 'studtests/First.html')

latest\_question\_list = Question.objects.all()

context = {'latest\_question\_list': latest\_question\_list}

if 'username' in request.session:

context['username'] = request.session['username']

if 'usertype' in request.session:

context['usertype'] = request.session['usertype']

return render(request, 'studtests/index.html', context)

"""Random questions"""

def RndQuestions(questionlist, count):

return random.sample(questionlist, int(count))

"""Tests where subject = chosen subject"""

def subject\_test(request, subject\_id):

tests = []

subject = Subject.objects.get(id=int(subject\_id))

for test in Test.objects.all():

finished = False

if 'usertype' in request.session:

if request.session['usertype'] == "s":

student = Student.objects.get(login=request.session['username'])

for tr in TestResult.objects.all():

if tr.test.name == test.name and tr.student == student:

finished = True

if test.subject == subject and test.visibility == True and test.school == subject.school and finished == False:

tests.append(test)

context = {'tests': tests}

if 'username' in request.session:

context['username'] = request.session['username']

if 'usertype' in request.session:

context['usertype'] = request.session['usertype']

return render(request, 'studtests/subject\_test.html', context)

"""Subjects where school == student school"""

def subjects(request):

user = Student()

if 'username' in request.session and request.session['usertype'] == "s":

user = Student.objects.get(login=request.session['username'])

subs = Subject.objects.all()

subjects = []

for sub in subs:

if sub.school == user.school:

subjects.append(sub)

context = {'subjects': subjects}

if 'username' in request.session:

context['username'] = request.session['username']

if 'usertype' in request.session:

context['usertype'] = request.session['usertype']

return render(request, 'studtests/subjects.html', context)

"""Here is test's questions and choices for them"""

def detail(request, test\_id):

test = get\_object\_or\_404(Test, pk=test\_id)

timetest = TimeTest()

rndtest = RndTest()

questions = Question.objects.all()

choices = Choice.objects.all()

context = {'test': test, 'questions': questions}

imquests = ImQuestion.objects.all()

type = ""

done = False

"""If test already done"""

for x in TestResult.objects.all():

if x.test == test:

done = True

if done:

return redirect('/')

for x in TimeTest.objects.all():

if x.name == test.name:

type = "time"

timetest = x

test = x

break

for x in RndTest.objects.all():

if x.name == test.name:

type = "rnd"

rndtest = x

test = x

qs = x.question\_set.all()

questions = RndQuestions(qs, x.qcount)

break

if 'username' in request.session:

context = {'test': test, 'username': request.session['username'], 'questions': questions, 'choices': choices,

'type': type, 'usertype': request.session['usertype'], 'test1': timetest, 'imqusests': imquests}

if 'bot' in request.POST:

context['timer'] = 'begin'

return render(request, 'studtests/detail.html', context)

"""for result string"""

def results(request, question\_id):

response = "You're looking at the results of question %s."

question = Question.objects.get(id=question\_id)

if "ch" in request.session:

if request.session["ch"]:

response += "True"

return HttpResponse(response % question\_id)

"""True/False"""

def vote(request, test\_id):

p = get\_object\_or\_404(Test, pk=test\_id)

student = Student.objects.get(login=request.session['username'])

finished = False

for t in TestResult.objects.all():

if t.test == p and t.student == student:

finished = True

if not finished:

checkbox\_list = [x for x in request.POST if x.startswith('choice')]

list = []

a = 0

right\_choices = []

unright\_choices = []

selected\_choices = []

"""get all choices with correct question"""

for c in checkbox\_list:

id = c.replace('choice\_', '')

selected\_choice = Choice.objects.get(id=int(id))

if selected\_choice.right\_choice:

list += 'True'

a += 1

right\_choices.append(selected\_choice)

else:

list += 'False'

unright\_choices.append(selected\_choice)

s = Student.objects.get(login=request.session['username'])

tr = TestResult(student=s, test=p, balls=0, quest\_count=len(p.question\_set.all()))

if (p in RndTest.objects.all()):

rtest = RndTest.objects.get(name=p.name)

tr = TestResult(student=s, test=p, balls=0, quest\_count=rtest.qcount)

tr.save()

right\_count = 0

unright\_count = 0

choicecount = 0

for x in tr.test.question\_set.all():

choicecount += len(x.choice\_set.all())

for x in right\_choices:

tr.right\_choices.add(x)

right\_count += 1

for x in unright\_choices:

tr.unright\_choices.add(x)

unright\_count += 1

tr.right\_count = right\_count

tr.unright\_count = unright\_count

tr.answers\_count = choicecount

tr.save()

if a == len(checkbox\_list):

s.question\_list.add(p)

return redirect('/')

else:

raise Http404("You've already finished this test")

"""Creating Test and Questions for it"""

def create(request):

context = {}

user = Teacher()

grades = Grade.objects.all()

question\_list = [x for x in request.POST if x.startswith('choice')]

if 'usertype' in request.session and 'username' in request.session:

if request.session['usertype'] == "s":

context = {'username': request.session['username'], 'usertype': "s"}

elif request.session['usertype'] == "t" and 'username' in request.session:

user = Teacher.objects.get(login=request.session['username'])

subs = [x for x in Subject.objects.all() if x.school == user.school]

if 'questions' in request.session:

context = {'username': request.session['username'], 'usertype': "t",

'count': len(request.session['questions']),

'sublist': subs, 'grades': grades, 'inputs': 4}

else:

context = {'username': request.session['username'], 'usertype': "t", 'sublist': subs, 'grades': grades,

'inputs': "123"}

if request.POST.get("create\_question"):

question = Question(question\_text=request.POST.get("question\_text"),

grade=Grade.objects.get(grade=request.session['grade']),

school=user.school, teacher=user, test=Test.objects.get(name=request.session['testname']),

theme=request.session['theme'], visibility=True,

subject=Subject.objects.get(subject=request.session['sub']))

formup = UploadFileForm(request.POST, request.FILES)

if formup.is\_valid:

try:

file = request.FILES['file']

question = ImQuestion(question\_text=request.POST.get("question\_text"),

grade=Grade.objects.get(grade=request.session['grade']),

school=user.school, teacher=user,

test=Test.objects.get(name=request.session['testname']),

theme=request.session['theme'], visibility=True,

subject=Subject.objects.get(subject=request.session['sub']), image=file)

except:

pass

question.save()

test = Test.objects.get(name=request.session['testname'])

sessionlist = request.session['questions']

sessionlist.append(question.id)

request.session['questions'] = sessionlist

right\_choices = 0

for c in question\_list:

id = c.replace('choice\_', '')

box = "id" + id

if box in request.POST:

choice = Choice(choice\_text=request.POST[str(c)], question=question, right\_choice=True)

right\_choices += 1

else:

if request.POST.get('choice\_' + id).replace(" ", "") == "":

continue

else:

choice = Choice(choice\_text=request.POST[str(c)], question=question, right\_choice=False)

choice.save()

test.choice\_count = right\_choices + request.session['c\_count']

request.session['c\_count'] += right\_choices

test.save()

context = {'username': request.session['username'], 'usertype': "t", 'count': len(request.session['questions']),

'formup': formup}

return render(request, "studtests/create\_question.html", context)

if request.POST.get("end"):

# rndq = RndQuestions(request.session['questions'], 3)

return render(request, "studtests/index.html",

{"username": request.session['username'], 'usertype': request.session['usertype']})

if request.POST.get("usualtest"):

context["testtype"] = "usual"

return render(request, "studtests/create\_test.html", context)

if request.POST.get("timetest"):

context["testtype"] = "time"

request.session['type'] = "time"

return render(request, "studtests/create\_test.html", context)

if request.POST.get("rndtest"):

context["testtype"] = "rnd"

request.session['type'] = "rnd"

return render(request, "studtests/create\_test.html", context)

if request.POST.get("create\_test"):

request.session['c\_count'] = 0

request.session['questions'] = []

nametest = request.POST.get("test\_name")

grade = request.POST.get("test\_grade")

theme = request.POST.get("test\_theme")

sub = request.POST.get("sel")

request.session['testname'] = nametest

request.session['grade'] = str(grade)

request.session['theme'] = theme

request.session['sub'] = str(sub)

if "time" in request.POST:

time = request.POST.get("time")

test = TimeTest(name=nametest, grade=Grade.objects.get(grade=grade), theme=theme,

subject=Subject.objects.get(subject=sub),

school=user.school, visibility=True, teacher=user, time=time)

test.save()

elif "rnd" in request.POST:

qucount = request.POST.get("rnd")

test = RndTest(name=nametest, grade=Grade.objects.get(grade=grade), theme=theme,

subject=Subject.objects.get(subject=sub),

school=user.school, visibility=True, teacher=user, qcount=qucount)

test.save()

else:

test = Test(name=nametest, grade=Grade.objects.get(grade=grade), theme=theme,

subject=Subject.objects.get(subject=sub),

school=user.school, visibility=True, teacher=user)

test.save()

formup = UploadFileForm(request.POST, request.FILES)

context['formup'] = formup

return render(request, "studtests/create\_question.html", context)

return render(request, "studtests/create\_test.html", context)

"""For teachers"""

def test\_res(request, res):

student = Student()

testres = TestResult.objects.get(id=res)

right = testres.right\_choices

unright = testres.unright\_choices

if 'usertype' in request.session and 'username' in request.session:

if request.session['usertype'] == "s":

raise Http404('Go away, student')

if request.session['usertype'] == "t":

student = Student.objects.get(login=testres.student.login)

context = {'username': request.session['username'], 'usertype': "t", 'right': right.all(),

'unright': unright.all()}

return render(request, "studtests/testresult.html", context)

return HttpResponse('Test result with id: ' + str(res) + ' and student: ' + student.login + str(right))

"""For students"""

def lookresults(request):

results = TestResult.objects.all()

ress = []

context = {}

if 'usertype' in request.session and 'username' in request.session:

if request.session['usertype'] == "s":

user = Student.objects.get(login=request.session['username'])

for res in results:

if res.student == user:

ress.append(res)

context = {'username': request.session['username'], 'usertype': "s", 'results': ress}

elif request.session['usertype'] == "t" and 'username' in request.session:

context = {'username': request.session['username']}

raise Http404("Yo're a teacher")

return render(request, "studtests/studresults.html", context)

"""For students"""

def lookresult(request, testres\_id):

context = {}

p = TestResult.objects.get(pk=testres\_id)

try:

user = Student.objects.get(login=request.session['username'])

except:

raise Http404("You do not have permissions")

if p.student == user:

right\_choices = p.right\_choices

unright\_choices = p.unright\_choices

if 'usertype' in request.session and 'username' in request.session:

if request.session['usertype'] == "s":

context = {'username': request.session['username'], 'usertype': "s", 'right': right\_choices.all(),

'unright': unright\_choices.all()}

elif request.session['usertype'] == "t":

raise Http404("You're teacher, mothefucker")

else:

raise Http404("Permission denied")

return render(request, "studtests/testresult.html", context)

"""All teacher's students"""

def students(request):

students\_ = []

context = {}

if 'usertype' in request.session and 'username' in request.session:

if request.session['usertype'] == 't':

user = Teacher.objects.get(login=request.session['username'])

for st in Student.objects.all():

if st.school == user.school:

students\_.append(st)

context = {'username': request.session['username'], 'usertype': request.session['usertype'],

'students': students\_}

return render(request, 'studtests/students.html', context)

"""All tests for one teacher"""

def all\_tests(request):

tests = []

context = {}

testtype = ''

raited = False

for x in TimeTest.objects.all():

f = 0

if 'usertype' in request.session and 'username' in request.session:

if request.session['usertype'] == 't':

user = Teacher.objects.get(login=request.session['username'])

for test in TestResult.objects.all():

if test.test.school == user.school:

tests.append(test)

if test.balls != 0:

raited = test.balls

if request.POST:

balls = [x for x in request.POST if x.startswith('rat')]

edballs = [x for x in request.POST if x.startswith('ed')]

edrats = [x for x in request.POST if x.startswith('eeds') and request.POST[x]]

entrats = [x for x in request.POST if x.startswith('ents') and request.POST[x]]

if len(balls) > 0 and len(entrats) > 0:

tr = TestResult.objects.get(pk=int(entrats[0].replace('ents', '')))

ball = 0

for i, x in enumerate(balls):

if int(entrats[0].replace('ents', '')) == int(x.replace('rat', '')):

ball = i

tr.balls = request.POST.get(balls[ball])

tr.save()

if len(edballs) > 0 and len(edrats) > 0:

edtr = TestResult.objects.get(pk=int(edrats[0].replace('eeds', '')))

edball = 0

for i, x in enumerate(edballs):

if edrats[0].replace('eeds', '') == x.replace('ed', '') or x[len(x) - 1] == 'c':

edball = i

edtr.balls = request.POST.get(edballs[edball])

edtr.save()

return redirect('/tests/')

context = {'username': request.session['username'], 'usertype': 't', 'tests': tests, 'raiting': raited}

return render(request, 'studtests/all\_tests.html', context)

"""User's profile"""

def profile(request):

args = {}

args['username'] = request.session['username']

args['usertype'] = request.session['usertype']

if args['usertype'] == 't':

user = Teacher.objects.get(login=args['username'])

args['tests'] = Test.objects.filter(teacher=user)

else:

user = Student.objects.get(login=args['username'])

trs = []

for x in TestResult.objects.all():

trs.append(x.test)

args['tests'] = [x for x in Test.objects.all() if x not in trs]

args['user'] = user

return render(request, 'studtests/profile.html', args)

"""All student results"""

def student\_results(request, student\_id):

context = {}

trs = []

if 'username' in request.session and 'usertype' in request.session:

if request.session['usertype'] == "t":

student = Student.objects.get(id=student\_id)

for tr in TestResult.objects.all():

if tr.student == student:

trs.append(tr)

context = {'usertype': "t", 'username': request.session['username'], 'trs': trs}

return render(request, 'studtests/student\_results.html', context)

"""Teacher's tests and managing them"""

def teachertests(request):

tests = []

if 'usertype' in request.session and request.session['usertype'] == "t":

teacher = Teacher.objects.get(login=request.session['username'])

tests = Test.objects.filter(teacher=teacher)

context = {'username': request.session['username'], 'usertype': 't', 'tests': tests}

setf = [x for x in request.POST if x.startswith("setf")]

if request.POST.get('apply') or len(setf) > 0:

dels\_ = [x for x in request.POST if x.startswith("del")]

viss\_ = [x for x in request.POST if x.startswith('vis')]

sts\_ = [x for x in request.POST if x.startswith("st")]

for a in viss\_:

test\_id = a.replace('vis', '')

test = Test.objects.get(id=test\_id)

test.visibility = True

test.save()

for a in sts\_:

test\_id = a.replace('st', '')

test = Test.objects.get(id=test\_id)

test.visibility = False

test.save()

for a in dels\_:

questions = []

choices = []

test\_id = a.replace('del', '')

test = Test.objects.get(id=test\_id)

for x in Question.objects.all():

if x.test == test:

questions.append(x)

for x in questions:

for y in Choice.objects.all():

if y.question == x:

choices.append(y)

for x in questions:

x.delete()

for x in choices:

x.delete()

test.delete()

return render(request, 'studtests/created tests.html', context)

for a in setf:

test\_id = a.replace('setf', '')

test = Test.objects.get(pk=int(test\_id))

undone = []

completests = []

for x in TestResult.objects.all():

if x.test == test:

completests.append(x.student)

for x in Student.objects.all():

if x not in completests and x.grade == test.grade and x.school == test.school:

undone.append(x)

for x in undone:

testres = TestResult(student=x, test=test, balls=2, quest\_count=len(test.question\_set.all()),

right\_count=0, unright\_count=0,

answers\_count=len(

Choice.objects.filter(question=Question.objects.get(test=test))))

testres.save()

return render(request, 'studtests/created tests.html', context)

else:

raise Http404('Access denied')

"""Looking a test"""

def teachertest(request, test\_id):

test = Test.objects.get(id=test\_id)

questions = test.question\_set.all()

choices = Choice.objects.all()

return render(request, "studtests/tdetail.html",

{'username': request.session['username'], 'usertype': request.session['usertype'],

'questions': questions, 'choices': choices, 'test': test})

"""Show all apis"""

def apis(request):

try:

return render(request, 'studtests/apis.html',

{'username': request.session['username'], 'usertype': request.session['usertype']})

except:

return render(request, 'studtests/apis.html')

"""Info page"""

def info(request):

try:

return render(request, 'studtests/info.html',

{'username': request.session['username'], 'usertype': request.session['usertype']})

except:

return render(request, 'studtests/info.html')

"""Getting path for question image. Commentaries aren't deleted for not forgetting the method"""

def image(request, path):

path = 'studtests/images/' + path

"""Imtest = ImageTest.objects.get(id=3)

im = Imtest.image.path

im = im[24:]

"""

"""return render(request, 'studtests/getimg.html', {'path':path, 'username':request.session['username']})

a = ImageTest.objects.get(id = 1)

image = ImageTest.objects.get(image=path)

image\_data = open(path, "rb").read()"""

context = {'username': request.session['username'], 'usertype': request.session['usertype'], 'path': path}

# return HttpResponse(image\_data, content\_type="image/png")

return render(request, 'studtests/getimg.html', context)

def raitings(request):

student = Student.objects.get(login=request.session['username'])

trs = TestResult.objects.filter(student=student)

return render(request, 'studtests/raitings.html',

{'trs': trs, 'username': request.session['username'], 'usertype': request.session['usertype']})

"""Getting information from android device and sending callback"""

@csrf\_exempt

def gettrfromandr(request):

data = []

if request.POST:

if request.POST.get('login') and request.POST.get('password'):

print 'login and pass found'

user = auth.authenticate(username=request.POST.get('login'), password=request.POST.get('password'))

auth.login(request, user)

for t in Teacher.objects.all():

if t.login == request.POST.get('login'):

data = ['1', 't', t.id]

for s in Student.objects.all():

if s.login == request.POST.get('login'):

data = ['1', 's', s.id]

elif request.POST.get('testresults'):

print 'te'

questions = []

choices = []

rightq = []

falseq = []

anddata = request.POST.get('testresults')

data = anddata.split('/')

studname = data[0]

student = Student.objects.get(login=studname)

testid = data[1]

test = Test.objects.get(pk=int(testid))

for i, x in enumerate(data[2:len(data) - 1]):

if i % 2 == 0:

questions.append(x)

else:

choices.append(x)

for i, x in enumerate(questions):

question = Question.objects.get(question\_text=x)

if question.test == test:

choices\_ = Choice.objects.filter(question=question)

for j, y in enumerate(choices\_):

for k in choices:

if y.right\_choice and k[j] == '1':

rightq.append(y)

elif (y.right\_choice and k[j] == '0') or (not y.right\_choice and k[j] == '1'):

falseq.append(y)

try:

tr = TestResult(student=student, test=test, balls=0,

quest\_count=len(Question.objects.filter(test=test)),

unright\_count=len(falseq),

answers\_count=len(Choice.objects.filter(question=Question.objects.get(test=test))),

right\_count=len(rightq))

tr.save()

for x in rightq:

tr.right\_choices.add(x)

for x in falseq:

tr.unright\_choices.add(x)

tr.save()

except Exception:

print Exception

print 'right: ' + str(rightq)

print 'false' + str(falseq)

print request.POST.get('testresults')

elif request.POST.get('delete'):

test = Test.objects.get(pk=int(request.POST.get('delete')))

test.clean()

elif request.POST.get('getav'):

print request.POST.get('getav')

student = Student.objects.get(login=request.POST.get('getav'))

tested = False

strtests = ""

tests = Test.objects.all()

for x in tests:

for tr in TestResult.objects.all():

if tr.test == x and tr.student == student:

tested = True

if not tested:

strtests += str(x.id) + "/"

print "finished"

return HttpResponse(strtests)

elif request.POST.get('gettresults'):

trs = []

teacher = Teacher.objects.get(pk=int(request.POST.get('gettresults')))

for x in TestResult.objects.all():

if x.test.teacher == teacher:

trs.append(x.id)

strtrs = ""

for x in trs:

strtrs += x + "/"

return HttpResponse(strtrs)

elif request.POST.get('gettestinfo'):

test = Test.objects.get(pk=int(request.POST.get('getquestions')))

return test.theme + "/" + test.school + "/" + test.grade

elif request.POST.get('getquestions'):

test = Test.objects.get(pk=int(request.POST.get('getquestions')))

strqu = ""

for x in Question.objects.filter(test=test):

strqu += x.id + "/"

return HttpResponse(strqu)

elif request.POST.get('getchoices'):

question = Question.objects.get(pk=int(request.POST.get('getchoices')))

strch = ""

for x in Choice.objects.filter(question=question):

strch += x.id + "/"

return HttpResponse(strch)

elif request.POST.get('ttests'):

teacher = Teacher.objects.get(pk=int(request.POST.get('ttests')))

strtests = ""

for x in Test.objects.filter(teacher=teacher):

strtests += x.id + "/"

return HttpResponse(strtests)

return HttpResponse(data)

"""Ajax technologie for menu with subjects"""

def ajresp(request):

subs = []

ids = []

if logined(request)[1] == "s":

subs = [str(x.subject) for x in Subject.objects.all() if x.school == Student.objects.get(login=logined(request)[0]).school]

ids = [str(x.id) for x in Subject.objects.all() if x.school == Student.objects.get(login=logined(request)[0]).school]

strsubs = ""

for i, x in enumerate(subs):

strsubs += x + ',' + ids[i] + ','

return HttpResponse(strsubs)

**studtests/all\_tests.html**

{% extends "studtests/base.html" %}

{% load staticfiles %}

{% block all\_tests %}

<form action="/tests/" method="post">

{% csrf\_token %}

<div align="center">

<table>

<tr>

<td><b>Ученик</b></td><td><b>Тест</b></td><td><b>Правильных ответов</b></td><td><b>Неправильных</b></td>

<td><b>Всего вопросов</b></td><td><b>Всего ответов</b></td><td><b>Всего правильных</b></td><td><b>Оценка</b></td>

</tr>

{% for t in tests %}

<tr>

<td>{{t.student}}</td><td><a href="/tests/{{ t.id }}"><b>{{ t.test }}</b></a></td>

<td>{{ t.right\_count }}</td><td>{{ t.unright\_count }}</td>

<td>{{ t.quest\_count }}</td><td>{{ t.answers\_count }}</td>

<td>{{ t.test.choice\_count }}</td>

{% if t.balls == 0 or t.balls == "0" or t.balls == "" %}

<td><a href="#" name="rating" class="button tiny" id="rating" onclick="rait({{ t.id }})">Поставить оценку</a> </td>

<td style="display: none" id="tdrat{{ t.id }}"><input style="width: 80px;height: 30px" type="text" placeholder="Оценка" name="rat{{ t.id }}" id="r{{ t.id }}"></td>

<td style="display: none" id="tdentrat{{ t.id }}"><input type="submit" name="ents{{ t.id }}" class="button tiny" value="Поставить"> </td>

{% else %}

<td>{{ t.balls }}</td>

<td><a href="#" name="edit" class="button tiny" id="edit" onclick="edit({{ t.id }})">Изменить оценку</a> </td>

<td style="display: none" id="edrat{{ t.id }}"><input style="width: 80px;height: 30px" type="text" placeholder="Оценка" name="ed{{ t.id }}" id="d{{ t.id }}"></td>

<td style="display: none" id="edentrat{{ t.id }}"><input type="submit" name="eeds{{ t.id }}" class="button tiny" value="Изменить" onclick="chname({{ t.id }})"></td>

{% endif %}

</tr>

{% endfor %}

</table>

</div>

</form>

{% endblock %}

**Studtests/apis.html**

{% extends "studtests/base.html" %}

{% block apis %}

<div align="center">

<a href="/api/v1/students/?format=xml">Ученики</a><br>

<a href="/api/v1/teachers/?format=xml">Учителя</a><br>

<a href="/api/v1/tests/?format=xml">Тесты</a><br>

<a href="/api/v1/users/?format=xml">Пользователи</a><br>

<a href="/api/v1/schools/?format=xml">Школы</a><br>

<a href="/api/v1/questions/?format=xml">Вопросы</a><br>

<a href="/api/v1/choices/?format=xml">Ответы</a><br>

<a href="/api/v1/testresults/?format=xml">Результаты</a><br>

<a href="/api/v1/imquestions/?format=xml">Вопросы с изображением</a><br>

</div>

{% endblock %}

**Studtests/base.html**

<!DOCTYPE html>

{% load staticfiles %}

<link rel="stylesheet" type="text/css" href="{% static 'studtests/foundation.css' %}" />

<link rel="stylesheet" type="text/css" href="{% static 'studtests/foundation.min.css' %}" />

<link rel="stylesheet" type="text/css" href="{% static 'studtests/normalize.css' %}" />

<link rel="stylesheet" type="text/css" href="{% static 'studtests/style1.css' %}" />

<script src="{% static 'studtests/modernizr.js' %}"></script>

<script src="{% static 'studtests/timer.js' %}"></script>

<script src="{% static 'studtests/jquery-1.11.3.min.js' %}"></script>

<script src="{% static 'studtests/foundation.js' %}"></script>

<script src="{% static 'studtests/foundation.topbar.js' %}"></script>

<script src="{% static 'studtests/foundation.min.js' %}"></script>

<script type="text/javascript" src="http://code.jquery.com/jquery-1.11.3.min.js"></script>

<script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/swfobject/2.2/swfobject.js"></script>

<!--[if lt IE 9]>

<script type="text/javascript" src="{% static 'studtests/js/excanvas/excanvas.js' %}"></script>

<![endif]-->

<script type="text/javascript" src="{% static 'studtests/js/spinners/spinners.min.js' %}"></script>

<script type="text/javascript" src="{% static 'studtests/js/lightview/lightview.js' %}"></script>

<link rel="stylesheet" type="text/css" href="{% static 'studtests/css/lightview/lightview.css' %}"/>

<html>

<head lang="en">

<meta charset="UTF-8">

<title>School tests</title>

</head>

<body>

<nav class="top-bar" data-topbar role="navigation">

<div class="top-bar">

<section class="top-bar-section">

<ul class="right">

{% if usertype == "t" %}

<li><a href="/mytests">Мои тесты</a></li>

<li><a href="/tests">Все результаты тестов ваших учеников</a></li>

<li><a href="/create\_test">Создать тест</a></li>

<li><a href = "/students">Ученики</a></li>

{% endif %}

{% if usertype == "s" %}

<li>

<ul class="dropdown1">

<li class="dropdown1-top" onmouseover="dropdownsubjects()">

<a href="{% url 'studtests:subjects' %}" class="dropdown1-top">Уроки &raquo;</a>

<ul class="dropdown1-inside" id="drin">

</ul>

</li>

</ul>

</li>

<li><a href="{% url 'studtests:raitings' %}">Оценки</a> </li>

{% endif %}

<li class="active"><a href="/">Домой</a></li>

{% if username %}

<li><a href="/profile">Профиль</a> </li>

<li><a href="/auth/logout/">Выйти ({{ username }})</a> </li>

{% else %}

<li><a href="/auth/login">Войти</a> </li>

<li><a href="/auth/action">Регистрация</a> </li>

{% endif %}

</ul>

</section>

</div>

</nav>

<div id="footer">

&copy; Молчанов Михаил & Горбачев Александр <div class="right">

<a href="/info" class="A">Информация &nbsp&nbsp&nbsp</a>

<a href="/apis" class="A">Наши API</a></div>

</div>

{% if username %}

{% block getimg %}{% endblock %}

{% block testresult %}{% endblock %}

{% if username and usertype == "s" %}

{% block index %}{% endblock %}

{% block subjects %}{% endblock %}

{% block detail %}{% endblock %}

{% block subject\_test %}{% endblock %}

{% block test %}{% endblock %}

{% block studresults %}{% endblock %}

{% elif username and usertype == "t" %}

{% block tdetail %}{% endblock %}

{% block createdtests %}{% endblock %}

{% block student\_results %}{% endblock %}

{% block create\_test %}{% endblock %}

{% block students %}{% endblock %}

{% block all\_tests %}{% endblock %}

{% block classes %}{% endblock %}

{% endif %}

{% else %}

{% block First %}{% endblock %}

{% endif %}

{% block login %}{% endblock %}

{% block action %}{% endblock %}

{% block apis %}{% endblock %}

{% block info %}{% endblock %}

{% block raitings %}{% endblock %}

{% block profile %}{% endblock %}

</body>

</html>

**Studtests/create\_question.html**

{% extends "studtests/base.html" %}

{% block create\_test %}

{{ count }}

<div class="large-offset-3 large-6 columns">

<form action="/create\_test/" method="post" class="inl" enctype="multipart/form-data">

{% csrf\_token %}

<label for="question\_text">Текст вопроса</label>

<input type="text" name="question\_text" id="question\_text">

<label for="choice\_1">Варианты ответов</label>

<div id="choices">

{% for i in "1234" %}

<div class="inl">

<input type="text" name = "choice\_{{ forloop.counter }}" id ="choice\_{{ forloop.counter }}">

<input type="checkbox" name="id{{ forloop.counter }}" id = "id{{ forloop.counter }}">

</div>

{% endfor %}

</div>

<br>

{{ formup }}

<a href="#" class="button round" onclick="addchoice()">Добавить ответ</a>

<input class="button" type="submit" name="create\_question" value="Создать вопрос"><br>

<input class="button success round" type="submit" name="end" value="Закончить создание теста">

</form>

</div>

{% endblock %}

**Studtests/create\_tests.html**

{% extends "studtests/base.html" %}

{% block create\_test %}

<div class="large-offset-3 large-6 columns" style="margin-top: 10%">

<form action="/create\_test/" method="post">

{% csrf\_token %}

{% if testtype %}

<label for="test\_name">Название теста</label>

<input type="text" name="test\_name" id="test\_name">

<label for="test\_theme">Тема</label>

<input type="text" name="test\_theme" id="test\_theme">

<label for="test\_grade">Класс</label>

<select name="test\_grade" id="test\_name">

<option selected = "selected">Выберите класс</option>

{% for grade in grades %}

<option>{{ grade.grade }}</option>

{% endfor %}

</select>

<label for="sel">Предмет</label>

<select name ="sel">

<option selected = "selected">Выберите предмет</option>

{% for sub in sublist %}

<option>{{ sub.subject }}</option>

{% endfor %}

</select>

{% if testtype == "usual" %}

{% elif testtype == "time" %}

<label for="time">Время на тест в секундах</label>

<input type="text" name="time" id="time">

{% elif testtype == "rnd" %}

<label for="rnd">Количество вопросов в тесте</label>

<input type="text" name="rnd" id="rnd">

{% endif %}

<input class="button" type="submit" name="create\_test" value="Приступить к заполнению теста">

{% else %}

<input class="button" type="submit" name="usualtest" value="Обычный тест">

<input class="button" type="submit" name="timetest" value="Тест на время">

<input class="button" type="submit" name="rndtest" value="Случайные вопросы">

{% endif %}

</form>

</div>

{% endblock %}

**Studtests/created\_test.html**

{% extends "studtests/base.html" %}

{% block createdtests %}

<div id="center">

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

{% csrf\_token %}

<table>

<tr><td class="tdgr"><b>Test</b></td><td><b>Visible</b></td><td><b>Stop</b></td><td><b>Delete</b></td></tr>

{% for test in tests %}

<tr>

<td><b><a href = {% url "studtests:teachertest" test.id %}>{{ test }}</a></b></td>

{% if test.visibility %}

<td><input type="checkbox" name="vis{{ test.id }}" class="inl" checked></td>

{% else %}

<td><input type="checkbox" name="vis{{ test.id }}" class="inl"></td>

{% endif %}

<td><input type="checkbox" name="st{{ test.id }}" class="inl"></td>

<td><input type="checkbox" name="del{{ test.id }}" class="inl"></td>

<td><input type="submit" name="setf{{ test.id }}" id="setf" value="Выставить двойки" class="button tiny"></td>

</tr>

{% endfor %}

<tr><td><input class="button radius expand success" type="submit" name="apply" id="apply" value="Применить"></td></tr>

</table>

</form>

</div>

{% endblock %}

**Studtests/detail.html**

{% extends "studtests/base.html" %}

{% load staticfiles %}

<link href="{% static 'studtests/lightgallery/skins/default/style.css' %}" type="text/css" media="screen" rel="stylesheet" />

<script src="{% static 'studtests/lightgallery/lightgallery.min.js' %}" type="text/javascript"></script>

<script>lightgallery.init();</script>

{% block detail %}

<h1>Название теста: {{ test.name }}, Предмет: {{ test.subject }}

{% if type == "time" %}

{{ test1.time }} секунд на выполнение

{% if timer == 'begin' %}

Начать

{% endif %}

{% endif %}</h1>

{% if type == 'time' %}

<p id = "time"></p>

<button name="bot" id = "bot" onclick="testTimer ({{ test1.time }})">Начать</button>

<!--<button name="stop" id="stop" onclick="stop ()">Остановить</button>-->

<p id = "rezult">

</p>

{% endif %}

{% if error\_message %}<p><strong>{{ error\_message }}</strong></p>{% endif %}

{% if type == "time" %}

<form action="{% url 'studtests:vote' test.id %}" method="post" style="display: none" id="gettest">

{% elif type == "" or type == "rnd" %}

<form action="{% url 'studtests:vote' test.id %}" method="post" id="gettest">

{% endif %}

{% csrf\_token %}

<div align="center" style="margin-left: 0%">

<table width="30%">

<tr><td>Вопрос</td><td>Варианты ответа</td></tr>

{% for question in questions %}

<tr>

{% if question.test == test and question.school == test.school and test.visibility %}

<td>{{ question.question\_text }}

{% for im in imqusests %}

{% if im.test == question.test and im.question\_text == question.question\_text %}

{% with "studtests"|add:im.image.url as ima %}

<a href="{% static ima %}" class="lightview" title="Leaves after rain"><img src="{% static ima %}" width="80%"></a></td>

{% endwith %}

{% endif %}

{% endfor %}

{% for choice in choices %}

{% if question == choice.question %}

<td><input class="inl" type="checkbox" name="choice\_{{ choice.id }}" id="choice{{ forloop.counter }}" value="{{ choice.id }}" />

<label class="inl" for="choice{{ forloop.counter }}">{{ choice.choice\_text }}</label></td>

{% endif %}

{% endfor %}

{% endif %}

</tr>

{% endfor %}

</table>

<input type="submit" class="button success" value="Отправить" />

</div>

</form>

{% endblock %}

**Studtests/first.html**

{% extends "studtests/base.html" %}

{% load staticfiles %}

{% block First %}

<div id="center">

<a href="/auth/action" class="button radius expand success">Зарегистрироваться</a><br>

<a href="/auth/login" class="button radius expand success">Войти</a>

<br><br><br><br><br><br>

<div style="background-color: rgba(255, 255, 255, 0.22)"><b>В чем наша польза:</b>

<ul style="list-style-type: square;">

<li><b>Различные виды тестов</b></li>

<li><b>Возможность оценивать результаты</b></li>

<li><b>Можно добавлять медиа контент</b></li>

</ul> </div>

<br><br>

<ul class="large-block-grid-3">

<li><a href="{% static 'studtests/images/mainpage/1.png' %}" class="lightview"><img class="th" src="{% static 'studtests/images/mainpage/1.png' %}"></a></li>

<li><a href="{% static 'studtests/images/mainpage/2.png' %}" class="lightview"><img class="th" src="{% static 'studtests/images/mainpage/2.png' %}"></a></li>

<li><a href="{% static 'studtests/images/mainpage/3.png' %}" class="lightview"><img class="th" src="{% static 'studtests/images/mainpage/3.png' %}"></a></li>

</ul>

</div>

{% endblock %}

**Studtests/getimg**

{% extends 'studtests/base.html' %}

{% load staticfiles %}

{% block getimg %}

<img src="{% static path %}" alt="kek"/>

{% endblock %}

**Studtests/index.html**

{% extends "studtests/base.html" %}

{% block index%}

{% if usertype == "s" %}

{% endif %}

{% if usertype == "t" %}

Вы учитель

<a href="/create\_test">Создать тест</a>

{% endif %}

{% if latest\_question\_list %}

<ul>

</ul>

{% else %}

<p>No tests are available.</p>

{% endif %}

{% endblock %}

**Studtests/info.html**

{% extends "studtests/base.html" %}

{% block info %}

Here will be information about us

{% endblock %}

**Studtests/profile.html**

{% extends 'studtests/base.html' %}

{% block profile %}

<div id="center">

<b>Имя: </b>{{ user.name }}<br>

<b>Фамилия: </b>{{ user.surname }}<br>

<b>Логин: </b>{{ user.login }}<br>

<b>Школа</b> {{ user.school }}<br>

{% if usertype == "t" %}

{% elif usertype == "s" %}

<b>Тесты: </b>{% for x in tests %}<a href="/studtests/{{ x.id }}">{{ x }},</a> {% endfor %}

{% endif %}

</div>

{% endblock %}

**Studtests/raitings.html**

{% extends 'studtests/base.html' %}

{% load staticfiles %}

{% block raitings %}

<div id="center">

<table>

<tr><td>Название теста</td><td>Тема</td><td>Предмет</td><td>Учитель</td><td>Оценка</td></tr>

{% for tr in trs %}

<tr><td>{{ tr.test.name }}</td><td>{{ tr.test.theme }}</td><td>{{ tr.test.subject }}</td><td>{{ tr.test.teacher }}</td><td>{{ tr.balls }}</td></tr>

{% endfor %}

</table>

</div>

{% endblock %}

**Studtests/subject\_test**

{% extends "studtests/base.html" %}

{% block subject\_test %}

{% if tests and username %}

<div id="center">

<table>

<tr><td>Доступные тесты</td></tr>

{% for test in tests %}

<tr><td><a href="{% url 'studtests:detail' test.id %}">{{ test.name }}</a></td></tr>

{% endfor %}

</table>

</div>

{% else %}

<p>No tests are available.</p>

{% endif %}

{% endblock %}

**Studtests/subjects.html**

{% extends "studtests/base.html" %}

{% block subjects %}

{% if subjects %}

<div id="center" class="greendiv">

<table>

{% for subject in subjects %}

<tr><td><b><a href="{% url 'studtests:subject\_test' subject.id%}">{{ subject.subject }}</a></b></td></tr>

{% endfor %}

</table>

</div>

{% else %}

<p>No tests are available.</p>

{% endif %}

{% endblock %}

**Studtests/tdetail.html**

{% extends "studtests/base.html" %}

{% block tdetail %}

<h1>Тест: {{ test }}</h1>

<div id = "center">

<table>

<tr><td>Вопрос</td><td>Варианты ответов</td></tr>

{% for q in questions %}

<tr><td>{{ q }}</td>

<td>

{% for c in choices %}

{% if c.question == q %}

{{ c.choice\_text }},

{% endif %}

{% endfor %}

</td>

</tr>

{% endfor %}

</table>

</div>

{% endblock %}

**Studtests/testresult.html**

{% extends "studtests/base.html" %}

{% block testresult %}

<div align="center" id="testrestr">

<b>

{% if right %}

True: <br>

{% endif %}

{% for r in right %}

{{ r }} <br>

{% endfor %}

{% if unright %}

False: <br>

{% endif %}

{% for u in unright %}

{{ u }} <br>

{% endfor %}

</b>

</div>

{% endblock %}

**Studtests/timer.js**

var chid = 4;

var mmoved = false;

/\*Timer code beginnig\*/

//Объявим переменную

var stopTimer;

//Функция для старта

function testTimer(startTime) {

var gettest = document.getElementById("gettest");

//для повторного запуска очистим rezult

document.getElementById("rezult").innerHTML = '';

//выключим кнопку запуска

var bot = document.getElementById("bot");

bot.setAttribute("disabled","");

gettest.style.display = "";

//bot.style.display = "none";

//сколько будет длится обратный отчет

var time = startTime;

//определим сколько минут

var min = parseInt(time / 60);

if ( min < 1 ) min = 0;

time = parseInt(time - min \* 60);

if ( min < 10 ) min = '0'+min;

//определим сколько секунд

var seconds = time;

if ( seconds < 10 ) seconds = '0'+seconds;

//отрисовываем время

document.getElementById("time").innerHTML='<span>Осталось времени- '+min+' мин '+seconds+' секунд</span>';

//уменьшаем общее время на одну секунду

startTime--;

//смотрим время не закончилось

if ( startTime >= 0 ) {

//если нет то повторяем процедуру заново

stopTimer = setTimeout(function(){testTimer(startTime); }, 1000);

//если закончилось, то выводим сообщение на экран, и делаем кнопку запуска активной

} else {

document.getElementById("time").innerHTML='<span>Осталось времени- 00 мин 00 секунд</span>';

var rezult = document.getElementById("rezult");

rezult.innerHTML ="Время вышло";

clearTimeout(stopTimer);

var bot = document.getElementById("bot");

bot.removeAttribute("disabled","disabled");

bot.removeChild(bot.childNodes[0]);

var text = document.createTextNode("Начать заново");

bot.appendChild(text);

}

}

//Функция для остановки обратного отчета

function stop(){

var gettest = document.getElementById("gettest");

gettest.style.display = "none";

//очистим переменную с таймером

clearTimeout(stopTimer);

//и включим кнопку запуска

var bot = document.getElementById("bot");

bot.removeAttribute("disabled","disabled");

}

/\*Timer code ending\*/

//Functions for raitings

function rait(st){

document.getElementById('tdrat'+st).style.display = 'inline-block';

document.getElementById('tdentrat'+st).style.display = 'inline-block';

}

function edit(st){

document.getElementById('edrat'+st).style.display = 'inline-block';

document.getElementById('edentrat'+st).style.display = 'inline-block';

}

//Adding more choices to question

function addchoice() {

chid++;

var newDiv = document.createElement('div');

var newinp = document.createElement('input');

var newch = document.createElement('input');

var name = "id"+chid.toString();

newDiv.className='inl';

newinp.type = 'text';

newinp.name = 'choice\_'+chid;

newinp.id='choice\_'+chid;

newch.type='checkbox';

newch.name=name;

newch.id=name;

newDiv.appendChild(newinp);

newDiv.appendChild(newch);

block = document.getElementById('choices');

block.appendChild(newDiv);

}

function chname(id){

elem = document.getElementById('d'+String(id));

elem.name += 'c';

}

//Ajax crossplatform

function getXmlHttp(){

var xmlhttp;

try {

xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");

} catch (e) {

try {

xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");

} catch (E) {

xmlhttp = false;

}

}

if (!xmlhttp && typeof XMLHttpRequest!='undefined') {

xmlhttp = new XMLHttpRequest();

}

return xmlhttp;

}

//Ajax menu

function dropdownsubjects()

{

var responseserv;

var schools;

if(!mmoved) {

var req = getXmlHttp();

var el = document.getElementById("drin");

mmoved = true;

req.onreadystatechange = function() {

if (req.readyState == 4) {

if(req.status == 200) {

responseserv = req.responseText;

schools = responseserv.split(',');

addschools(schools);

}

}

};

req.open('GET', '/ajtry', true);

req.send(null);

}

}

function addschools(schools)

{

var el = document.getElementById("drin");

var sub;

for(var i = 0; i < schools.length; i++)

{

if(i % 2 == 0)

{

sub = schools[i];

}

else

{

var newli = document.createElement('li');

var newa = document.createElement('a');

newa.href = "/studtests/subjects/" + schools[i] + "/";

newa.innerHTML = sub;

newli.appendChild(newa);

el.appendChild(newli);

}

}

}

**Studtests/style1.css**

table{

background: rgb(228, 226, 255);

border: 2px solid #F2F2F2;

}

tr{

background: rgba(255,255,255,0);

}

td{

background: rgba(255, 233, 175, 0);

}

mtd{

background: rgba(255,233,175,0);

}

.tdgr{

color:green;

}

ul a {

color: green;

}

body {

background-image: url("images/Test.jpg");

background-size: 100% 100%;

background-repeat:no-repeat;

background-attachment: fixed;

}

.A {

color: White;

}

.rightpic {

float: right;

}

.greendiv{

color: #2E9AFE;

}

.inl{

display:inline-block;

}

#footer {

position: fixed;

left: 0; bottom: 0;

padding: 10px;

background: #39b54a;

color: #fff;

width: 100%;

}

#center {

top: 50%;

left: 50%;

width: 450px;

height: 450px;

position: absolute;

margin-top: -225px;

margin-left: -225px;

}

#center1 {

top: 50%;

left: 50%;

width: 800px;

height: 450px;

position: absolute;

margin-top: -225px;

margin-left: -225px;

}

#testrestr

{

top: 50%;

left: 50%;

width: 450px;

height: 450px;

position: absolute;

margin-top: -225px;

margin-left: -225px;

color: #3CB371;

}

#testresfa

{

top: 50%;

left: 50%;

width: 450px;

height: 450px;

position: absolute;

margin-top: -225px;

margin-left: -225px;

color: #EE0000;

}

ul.dropdown1 li { position: relative; }

ul.dropdown1,

ul.dropdown1-inside {

list-style-type: none;

padding: 0;

}

ul.dropdown1-inside {

position: absolute;

left: -9999px;

}

ul.dropdown1 li.dropdown1-top {

display: inline;

float: left;

margin: 0 1px 0 0;

}

ul.dropdown1 li.dropdown1-top a {

padding: 3px 10px 4px;

display: block;

}

ul.dropdown1 a.dropdown1-top { background: #333333; }

ul.dropdown1 a.dropdown1-top:hover { }

ul.dropdown1 li.dropdown1-top:hover .dropdown1-inside {

display: block;

left: 0;

}

ul.dropdown1 .dropdown1-inside { background: #c8cece; }

ul.dropdown1 .dropdown1-inside a:hover { background: #c8cece; }

## 9.2 Android