

Колледж космического машиностроения и технологий

**КУРСОВОЙ ПРОЕКТ**

**По МДК.01.02 «Прикладное программирование»**

**Тема: «Разработка музыкального андройд-плеера»**

Выполнил студент

Герасимов Д.А.

Группа П1-18

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Подпись)

\_\_\_\_\_\_\_\_\_\_\_ (Дата сдачи работы)

Принял преподаватель

Гусятинер Л.Б.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Оценка)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Подпись)

**Королёв 2021 г.**

Оглавление

[Введение 2](#_Toc69115482)

[1 Теоретическая часть 4](#_Toc69115483)

[1.1 Описание предметной области 4](#_Toc69115484)

[1.2 Описание существующих разработок 6](#_Toc69115485)

[2 Проектная часть 9](#_Toc69115486)

[2.1 Диаграмма прецедентов 9](#_Toc69115487)

[2.2 Выбор инструментов 9](#_Toc69115488)

[2.2.1 Язык программирования 9](#_Toc69115489)

[2.2.2 Среда разработки программного обеспечения 10](#_Toc69115490)

[2.2.3 База данных 10](#_Toc69115491)

[2.3 Проектирование сценария 11](#_Toc69115492)

[2.4 Диаграммы классов 12](#_Toc69115493)

[2.5 Описание главного модуля. 14](#_Toc69115494)

[2.6 Описание спецификаций к модулям 15](#_Toc69115495)

[2.7 Описание модулей 16](#_Toc69115496)

[2.8 Описание тестовых наборов модулей 19](#_Toc69115497)

[2.9 Описание применения средств отладки 20](#_Toc69115498)

[2.10 Анализ оптимальности использования памяти и быстродействия 21](#_Toc69115499)

[3 Эксплуатационная часть 23](#_Toc69115500)

[3.1 Руководство оператора 23](#_Toc69115501)

[3.1.1 Назначение программы 23](#_Toc69115502)

[3.1.2 Условия выполнения программы 23](#_Toc69115503)

[3.1.3 Выполнение программы 23](#_Toc69115504)

[3.1.4 Сообщение оператору 29](#_Toc69115505)

[3.2 To-Do лист 29](#_Toc69115506)

[Заключение 30](#_Toc69115507)

[Список литературы и интернет ресурсов 31](#_Toc69115508)

# Введение

Данный курсовой проект заключается в разработке приложения под операционную систему Android. Приложение представляет собой аудио проигрыватель, который воспроизводит аудиофайлы из памяти устройства. Разработка программы будет происходить на языке программирования Kotlin.

Данный курсовой проект заключается в разработке динамического сайта с тестами. Разработка сайта будет происходить на HTML, CSS, JS, Python Django.

В первой части будут рассмотрены предметная область и существующие продукты по данной теме.

Во второй части будут описаны инструменты разработки программы и модули программы.

В третьей части описаны взаимодействие пользователя с программой и сообщения оператору.

В заключительной части будут приведены общие выводы по проекту.

# Теоретическая часть

## Описание предметной области

**Педагогический тест** – это совокупность взаимосвязанных заданий возрастающей трудности и специфической формы, позволяющих качественно оценить структуру и измерить уровень знаний.

**Функции теста**

Педагогическое тестирование выполняет три основные взаимосвязанные функции: Диагностическую, обучающую и воспитательную:

Диагностическаяфункция благодаря этому свойству педагогическая технология позволяет своевременно получать как актуальную информацию   о состоянии учебного процесса в целом, так и результаты контроля по отдельным этапам обучения. Это позволяет организовать постоянный мониторинг образовательного процесса с целью прогнозирования индивидуальных траекторий, обучаемых в ближайшем и отдаленном будущем.

Обучающаяфункция тестирования состоит в мотивировании учащегося к активизации работы по усвоению учебного материала. Для усиления обучающей функции тестирования, могут быть использованы дополнительные меры стимулирования студентов, такие, как раздача преподавателем примерного перечня вопросов для самостоятельной подготовки, наличие в самом тесте наводящих вопросов и подсказок, совместный разбор результатов теста. Наиболее ярко проявляет себя в программированном обучении

Воспитательнаяфункция проявляется в периодичности и неизбежности тестового контроля. Это дисциплинирует, организует и направляет деятельность учащихся, помогает выявить и устранить пробелы в знаниях, формирует стремление развить свои способности.

**Виды тестовых заданий:**

**1. Задания с выбором ответов (закрытые задание).**

**Задания с выбором одного правильного ответа.**

Пример:

При наборе текста слова отделяются друг от друга …

а) двоеточием;

б) запятой;

в) пробелом;

г) точкой.

**Задания с выбором одного неправильного ответа.**

Пример:

Операция не имеет признака, по которому подобраны остальные операции, представленные в списке…

а) сохранение текста;

б) форматирование текста;

в) удаление фрагмента текста;

г) перемещение фрагмента теста;

д) копирование фрагмента текста.

**2. Задания на установление соответствия.**

Пример:

Установите соответствие.

|  |  |  |
| --- | --- | --- |
| **команда** | **сочетание клавиш** |  |
| 1. Вырезать фрагмент текста; | а) CTRL+C; |  |
| 2. Копировать фрагмент текста; | б) CTRL+V; |  |
| 3. Вставить фрагмент текста. | в) CTRL+X. |  |

**3. Задания с выбором нескольких правильных ответов.**

Пример:

Использование слепого десятипальцевого метода ведет к …

а) снижению напряжения на пальцы;

б) уменьшению скорости печати;

в) уменьшению количеству опечаток и ошибок;

г) быстрой утомляемости пальцев.

**4. Задания с открытым ответом.**

Пример:

Существует два способа освоения клавиатуры при печатании слепым десятипальцевым методом:

1.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Преимущества и недостатки**

По сравнению с другими формами контроля знаний тестирование имеет свои преимущества и недостатки.

Преимущества:

**Простота процедуры.** В отличие от других методов исследования, тесты являются одним из самых простых инструментов. Это совершенно не означает, что они тривиальны. В отличие от целенаправленных исследований или каких-нибудь многофакторных методов, тесты позволяют воспользоваться преимуществами современных технологий. Сегодня можно создавать и проходить тестовые опросы даже на мобильном телефоне.

**Положительно влияет на успеваемость.** Согласно исследованию, в области тестирования более 90% студентов обнаружили, что стандартизированные тесты положительно влияют на их достижения. Студенты чувствуют себя лучше в отношении их способности понимать и знать предметные материалы, которые представлены в тесте.

**Легко добраться до респондентов**. Упомянутая простота связана с легкостью охвата респондентов. Опросы не слишком сложные и не отнимают много времени, поэтому существует больше шансов привлечь кого-то к участию. Тем более на сегодня многие люди сами заинтересованы в прохождении разнообразных тестов для определений своих способностей.

**Автоматизация**. Сегодня технологии позволяют без особых трудностей и даже при отсутствии специализированных знаний создавать тестовые опросы. Самое главное, что система, которая предоставляет респонденту задания, является автоматизированной. Это позволяет существенно сократить издержки на разработку тестовых заданий, их обработку и определение результатов. Помимо этого, с помощью тех данных, которые получены в результате опросов, можно создавать огромные по объёму информации выкладки с применением разного рода фильтров и параметров. Без использования автоматизированных систем, работа с таким объёмом данных является очень затратной, а во многих случаях фактически невозможной.

**Объективность**. Процедура тестирования является объективной. За идентичные решения все испытуемые получают одинаковое количество баллов. Кроме того, многие считают, что именно компьютер обеспечивает абсолютную честность оценки.

Недостатки:

**Нельзя проверить глубину знаний**. Тесты дают поверхностную оценку знаний респондента. Возможна ситуация при которой тестируемый человек хорошо знает тему вопроса и может раскрыть ее достаточно глубоко, но не помнит отдельных фактов или точных значений (например, дат, имен, названий, разного рода высказываний и определений).

**Угадывания.**Очень часто тесты составлены так что, даже не зная правильного ответа можно определить его с помощью метода исключения. Бывает, что все или несколько предоставленных вариантов (кроме правильного) не подходят по контексту, они могут быть просто некорректными, шуточными или глупыми. Поэтому итоговая оценка респондента может быть несправедливой. Но даже если задания составлены правильно, все равно не исключено угадывание, в отличие от других методов оценивания (например, заданий, где нет вариантов ответов и нужно указать полный ответ на вопрос).

**Отсутствие творческой составляющей**. Тест не показывает творческих или исследовательских наклонностей респондента. Негативным фактором также является привыкание тестируемых к данному виду опросов, что ведет к потере мотивации развития в сторону свободного творческого мышления. Человек знает, что для прохождения теста ему достаточно знать какие-то разрозненные факты и пользуется этим, не углубляясь в тему.

**Трудности повторного применения**. Для повторного использования теста по той же теме, необходимо пересоздавать тест или хотя бы переформулировать вопросы. Иногда используется перестановка вопросов, но это может привести к механическому запоминанию без вникания в суть задания.

**Предсказуемость стандартизированного тестирования**. Студенты, которые знают о шаблонах, могут определить, какими будут ответы на стандартизированный тест, зная при этом только несколько ответов из всего задания. Это означает, что результаты тестов могут быть высокими без должных знаний респондента.

## Описание существующих разработок

На данный момент существует множество уже разработанных приложений по этой тематике. Практически все из них предоставляют похожий функционал:

* Возможность создания теста
* Разделение прав доступа для использования приложения (Пользователь, Администратор)
* Возможность сравнивать правильные ответы с ответами пользователя, после прохождения теста

Примеры сайтов:

* 16Personalities [1]

При прохождении теста демонстрирует известных людей, у которых такой же результат, как и у вас. Также есть разделение личности человека на 4 группы, каждое из них отличается цветом, при этом каждая группа ещё делится на 4 типа личности (у каждой группы они свои).

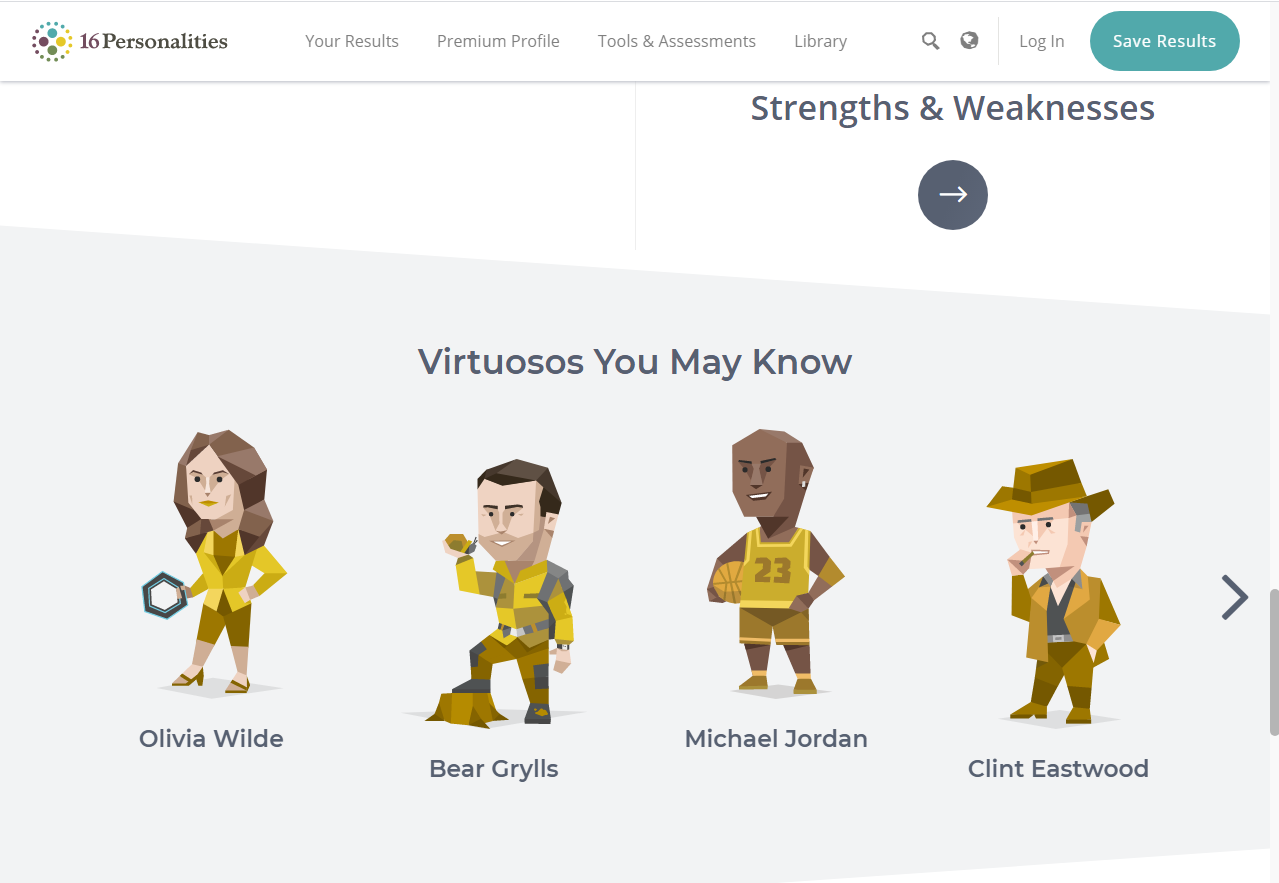


Рисунок 1. Окно сайта 16Personalities

* testometrika [2]

После прохождения тестов рекомендуют обратиться к психологу.

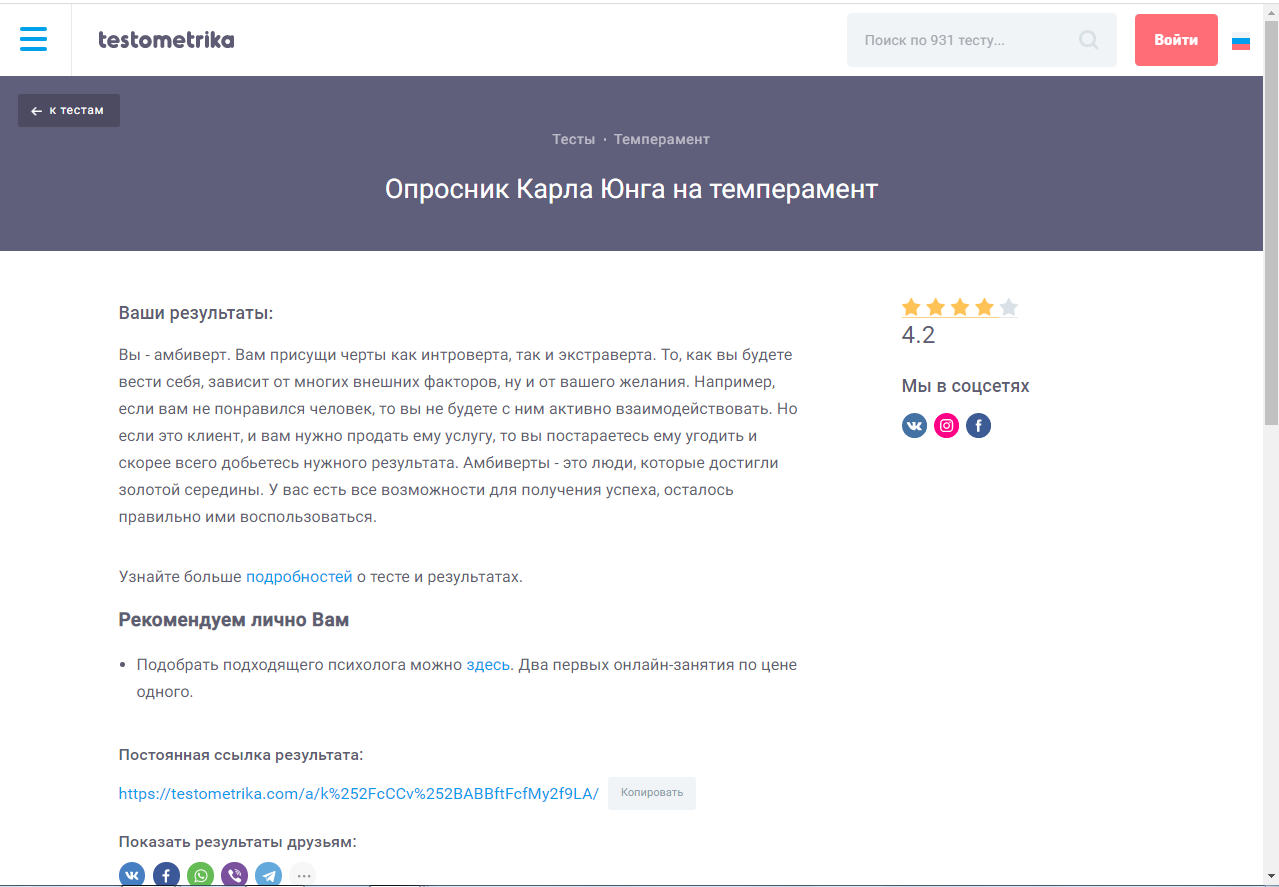


Рисунок 2. Окно приложения testometrika

* Викиум [3]

Хоть и не является сайтом с тестами, но при этом предоставляет возможность тестировать свой мозг, чтобы потом вы могли замечать прогресс при развитии своего мозга на данном сайте.

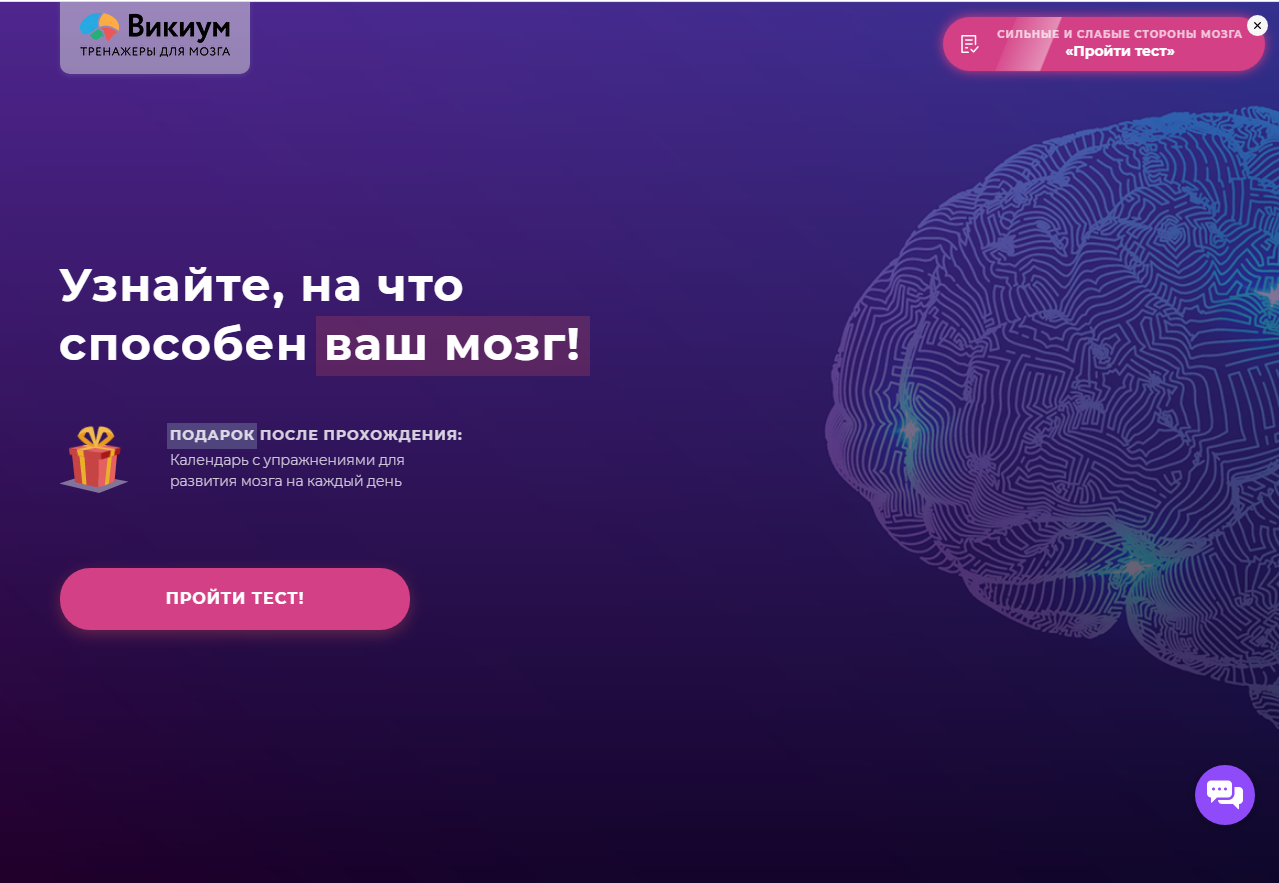


Рисунок 3. Окно приложения Викиум

# Проектная часть

## Диаграмма прецедентов

В данном разделе содержится диаграмма прецедентов для приложения, на которой показаны возможные функциональные отношения.

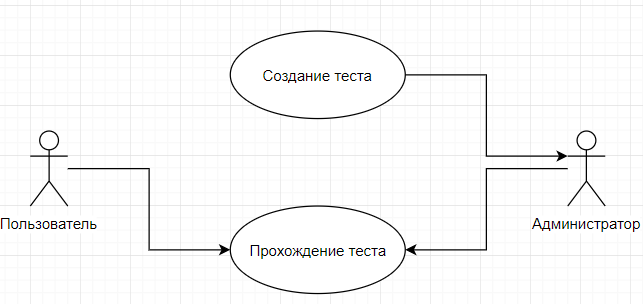


Рисунок 4. Диаграмма прецедентов сайта.

## Выбор инструментов

### Визуальная часть сайта

Для написания визуальной части сайта (frontend), используется HTML, CSS и JavaScript. Альтернатив этой тройке не найти, поэтому данный курсовой проект использует все из вышеперчисленного.

**HTML [4]** (**HyperText Markup Language**) — это язык гипертекстовой разметки страницы. Он используется для того, чтобы дать браузеру понять, как нужно отображать загруженный сайт.

HTML состоит из тегов — это своеобразные команды, которые преобразовываются в визуальные объекты в браузере пользователя. Например, тег <img> используется для размещения изображений на странице. У него есть обязательный атрибут src, в котором указывается ссылка на файл.

**CSS [5]** (Cascading Style Sheets.) – это стандартный язык таблиц стилей, используемый для описания представления (т.е. макета и форматирования) веб-страниц.

До CSS почти все атрибуты представления документов HTML содержались в разметке HTML (внутри тегов HTML); все цвета шрифта, стили фона, выравнивание элементов, границы и размеры должны были быть явно описаны в HTML.

В результате разработка крупных веб-сайтов стала длительным и дорогостоящим процессом, поскольку информация о стиле неоднократно добавлялась на каждую страницу веб-сайта.

Для решения этой проблемы CSS был представлен в 1996 году консорциумом World Wide Web (W3C), который также поддерживает этот стандарт. CSS был разработан, чтобы обеспечить разделение презентации и контента. Теперь веб-дизайнеры могут перемещать информацию о форматировании веб-страниц и ее визуальном представлении в отдельную таблицу стилей, что значительно упрощает разметку HTML и повышает удобство сопровождения.

**JavaScript [6]** – это мультипарадигменный язык программирования, который обычно применяется в качестве встраиваемого инструмента для программного доступа к различным объектам приложений. С точки зрения веб-разработки, без знаний этой технологии невозможно заниматься созданием современных интерактивных сайтов. Язык JS – это то, что «оживляет» разметку страниц (HTML) и пользовательский функционал (CMS) сайтов. С помощью этого языка реализуется возможность реакции страницы или отдельных ее элементов на действия посетителя. Сегодня JavaScript является базовым языком программирования для браузеров. Он полностью совместим с операционными системами Windows, Linux, Mac OS, а также всеми популярными мобильными платформами.

1. **Логическая часть сайта**

Логическая часть сайта, также называемая как backend, писалась на языке программирования Python при использовании фреймоворка Django.

Писалась она по следующим причинам причинам:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Язык программирования | Java | PHP | Python | JavaScript |
| Простота сопровождения | 4 | 5 | 10 | 7 |
| Скорость разработки | 4 | 6 | 10 | 6 |
| Аудитория | 10 | 10 | 8 | 7 |
| ИТОГ | 18 | 21 | 28 | 20 |

**Django [7]** — это высокоуровневый Python веб-фреймворк, который позволяет быстро создавать безопасные и поддерживаемые веб-сайты. Созданный опытными разработчиками, Django берёт на себя большую часть хлопот веб-разработки, поэтому вы можете сосредоточиться на написании своего веб-приложения без необходимости изобретать велосипед. Он бесплатный и с открытым исходным кодом, имеет растущее и активное сообщество, отличную документацию и множество вариантов как бесплатной, так и платной поддержки.

### Среда разработки программного обеспечения

**Visual Studio Code [8]** – это легкий, но мощный редактор исходного кода, который работает на рабочем столе и доступен для Windows, macOS и Linux. Он поставляется со встроенной поддержкой JavaScript, TypeScript и Node.js и имеет богатую экосистему расширений для других языков (таких как C++, C#, Java, Python, PHP, Go).

Visual Studio Code имеет огромное количество плагинов, поэтому написание кода ускоряется в разы.

## Проектирование сценария

В данном разделе приведен сценарий использования сайта пользователем (Рисунок 5).

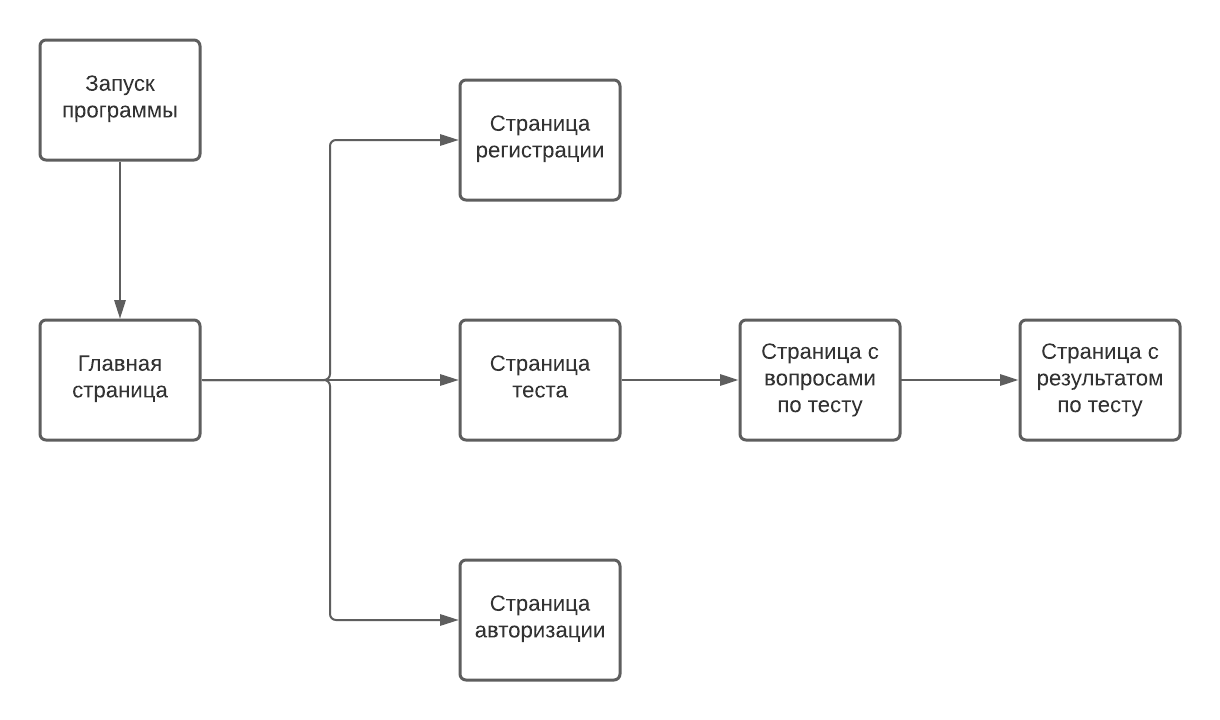


Рисунок 5 [9]. Сценарий использования сайта.

При запуске сайта откроется главная страница, на которой представлены все тесты. Далее пользователь может зарегистрировать аккаунт или войти в уже существующий. Также у пользователя есть возможность выбрать тест и перейти на страницу с тестом, где будет кнопка для старта теста, а также таблица результатов. После нажатия на кнопку старта, откроются вопросы к тесту, ну а после завершения откроется страница с результатом по пройденному тесту.

## Диаграммы классов

В данном разделе располагается диаграмма классов сайта. (Рисунок 6).

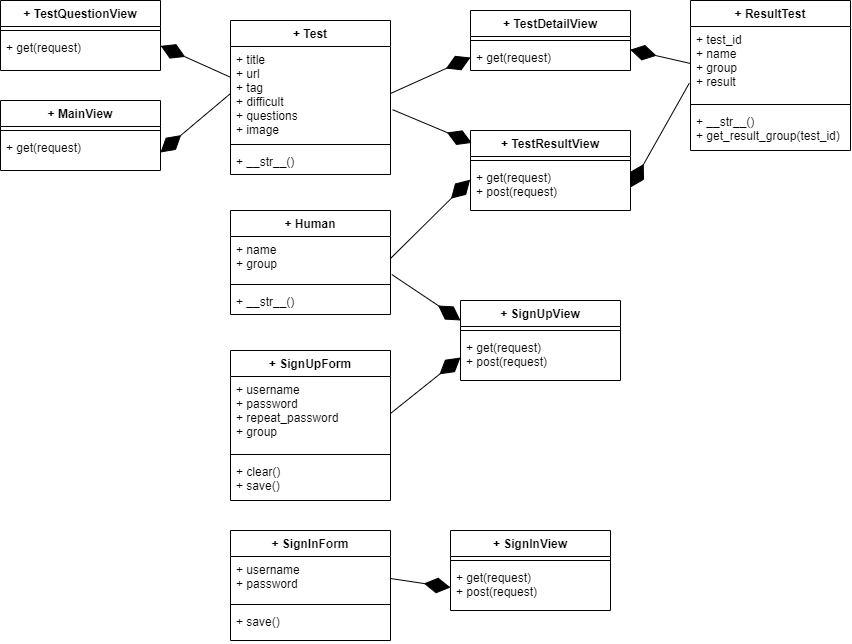


Рисунок 6. Диаграмма классов сайта

## Описание главного модуля.

Главный модуль состоит из класса MainView. Он выполняется при переходе на сайт. Данный модуль передаёт в шаблон html (home.html) все созданные тесты.

Листинг 1. Главный модуль

from django.shortcuts import render

from .models import Test

class MainView(View):

def get(self, request, \*args, \*\*kwargs):

tests = Test.objects.all()

return render( request, 'moon\_test\_app/home.html', context={

'tests': tests

})

## Описание спецификаций к модулям

Всего разработано 5 модулей:

* Аутентификация пользователя (SignUpView)
* Авторизация пользователя (SignInView)
* Страница теста (TestDetailView)
* Страница с вопросами по тесту (TestQuestionsView)
* Страница с результатами прохождения теста (TestResultView)

Взаимодействие между модулями можно наглядно увидеть по диаграммам классов из раздела 2.4.

## Описание модулей

**Аутентификация пользователя** – на данной странице находится форма для регистрации пользователя. Для регистрации нужно заполнить все поля, а именно:

* Имя пользователя
* Пароль
* Повторённый пароль
* Группа

После регистрации вы автоматически авторизуетесь. (Приложение 1)

Рассмотрим функцию регистрации пользователей и построим блок схему:

Листинг 2. Функция регистрации пользователей

def post(self, request, \*args, \*\*kwargs):

form = SignUpForm(request.POST) # Форма на регистрацию

# Если данные валидны

if form.is\_valid():

user = form.save() # Сохраняем пользователя в бд

# Если пользователь существует, то логинем этого пользователя

if user is not None:

human = dict()

human = Human(

name = request.POST['username'],

group = request.POST['group']

)

human.save()

login(request, user)

return HttpResponseRedirect('/') # Возращение на главную страницу

return render(request, 'moon\_test\_app/signup.html', context={

'form': form,

})

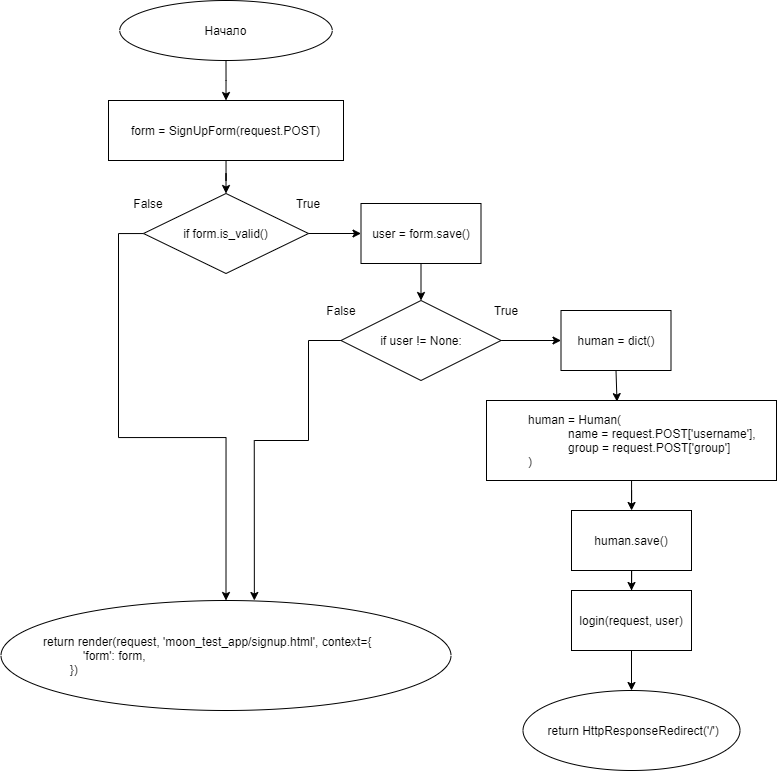


Рисунок 7. Блок схема функции post, класса SignUpView

**Авторизация пользователя** – содержит форму для входа пользователя (Приложение 2). Для этого нужно заполнить 2 поля:

Имя пользователя

Пароль

**Страница теста** – здесь располагается главное фото теста, название его, а также кнопка запуска теста. Также снизу есть таблица рекордов по тесту, который вы выбрали. Если тест ещё никто не проходил, то таблицы там не будет, а будет там некоторый текст (Приложение 3).

**Страница с вопросами по тесту** – на этой странице приведены вопросы (Приложение 4). Точнее будет сказать один вопрос. Но снизу есть кнопки:

Далее – открыть следующий вопрос

Назад – показать предыдущий вопрос

Закончить тест – окончить тест. Данная кнопка появляется, если вы находитесь на последнем вопросе.

**Страница с результатами прохождения теста** – на этой странице можно увидеть процент правильных ответов, по пройденному вами тесту, а также таблицу рекордов этого теста (Приложение 5).

## Описание тестовых наборов модулей

В некоторых модулях производится проверка данных на корректность, чтобы сообщить пользователю о проблеме и предотвратить аварийное завершение программы

**Тест 1. Проверка на правильный url теста**

Если пользователь решил самостоятельно вбить url нужного ему теста, и по каким-то причинам он вбил его с ошибкой, то выскачет ошибка 404. Данная ошибка скажет, что такой страницы не существует.

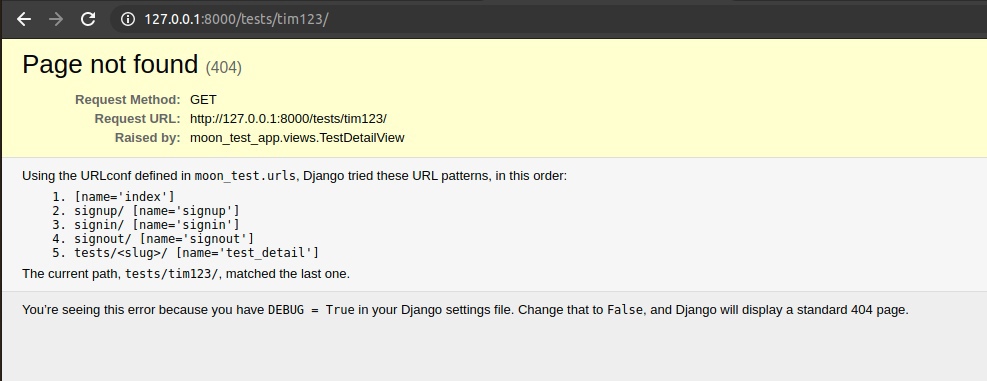


Рисунок 8. Ошбка при не правильном вводе url’a

**Тест 2. Проверка заполнения форм**

Если пользователь при аутентификации или авторизации допустил какую-либо ошибку, сверху ему выскочит плашка с ошибкой. Например, если пользователь неправильно повторил пароль, то плашка будет следующей:

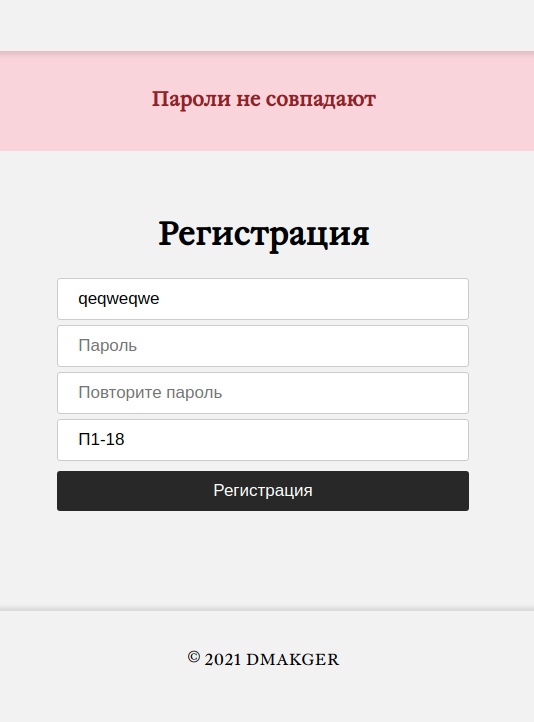


Рисунок 9. Ошибка при неправильном заполнении формы регистрации.

# Эксплуатационная часть

## Руководство оператора

### Назначение программы

Разработанный сайт позволяет проходить тесты, получать, а также с лёгкостью их создавать. Каждый тест хранится в базе данных, как и результаты пользователей и данные, которые ввели они.

### Условия выполнения программы

Любое устройство, выпущенное после 2012 года.

Браузеры: Edge, Firefox, Chrome, Safari, Opera, Safari on iOS, Opera Mini, Android Browser, Opera Mobile, Chrome for Android, Firefox for Android, UC Browser for Android, Samsung Internet, QQ Browser, Baidu Browser, KaiOS Browser.

### Выполнение программы

При запуске сайта вам откроется стартовая страница со всеми тестами:



Рисунок 10. Главная страница

Вы можете зарегистрировать пользователя, для этого надо заполнить данную форму:

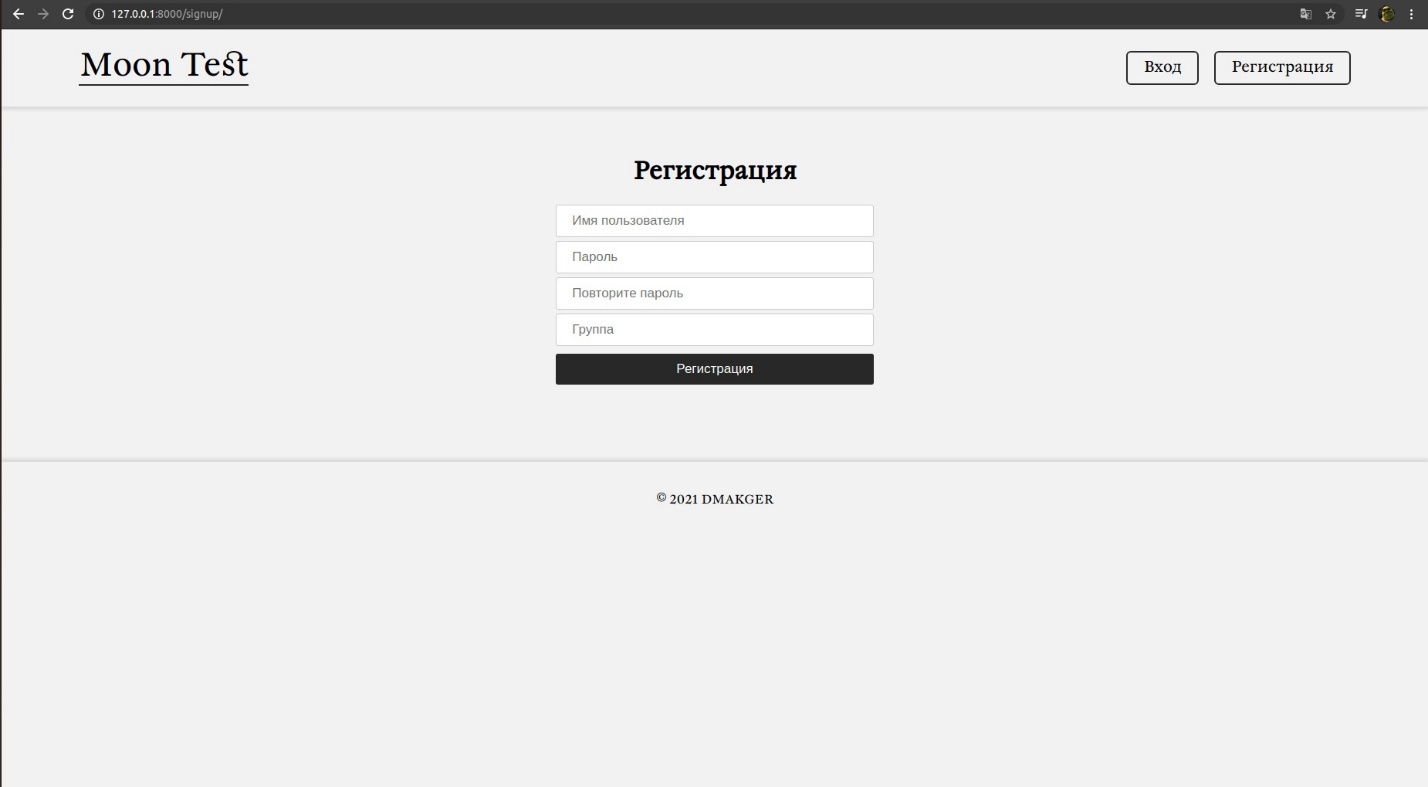


Рисунок 11. Форма аутентификации

Или войти в существующий аккаунт, для этого нужно ввести имя пользователя и пароль от него:

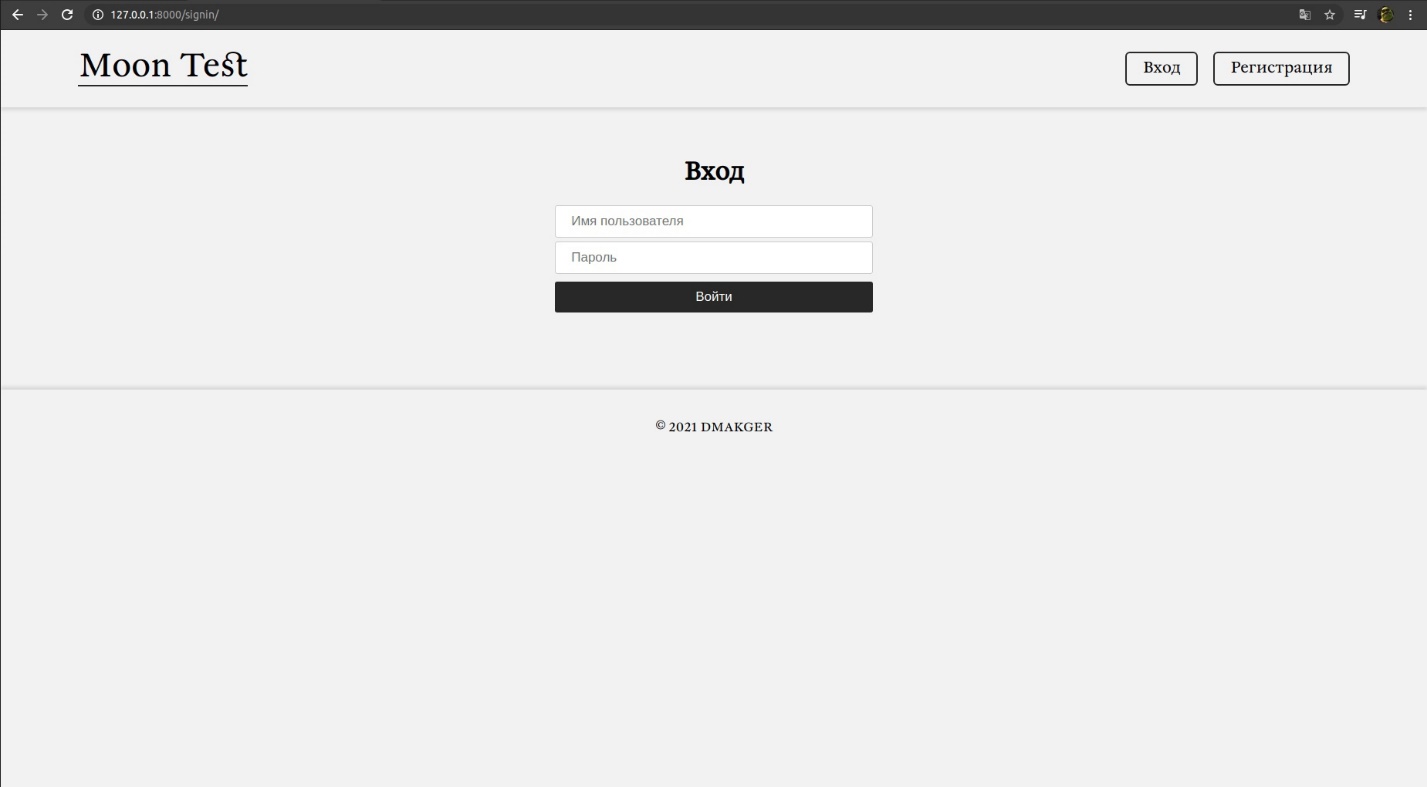


Рисунок 12. Форма авторизации

Либо же вы можете пропустить эти этапы. Тем не менее далее выбирается тест для прохождения, для этого нужно на него нажать.

После открытия теста вы можете увидеть таблицу результатов по данному тесту, а также кнопку «НАЧАТЬ ТЕСТ». Вам следует нажать на неё:

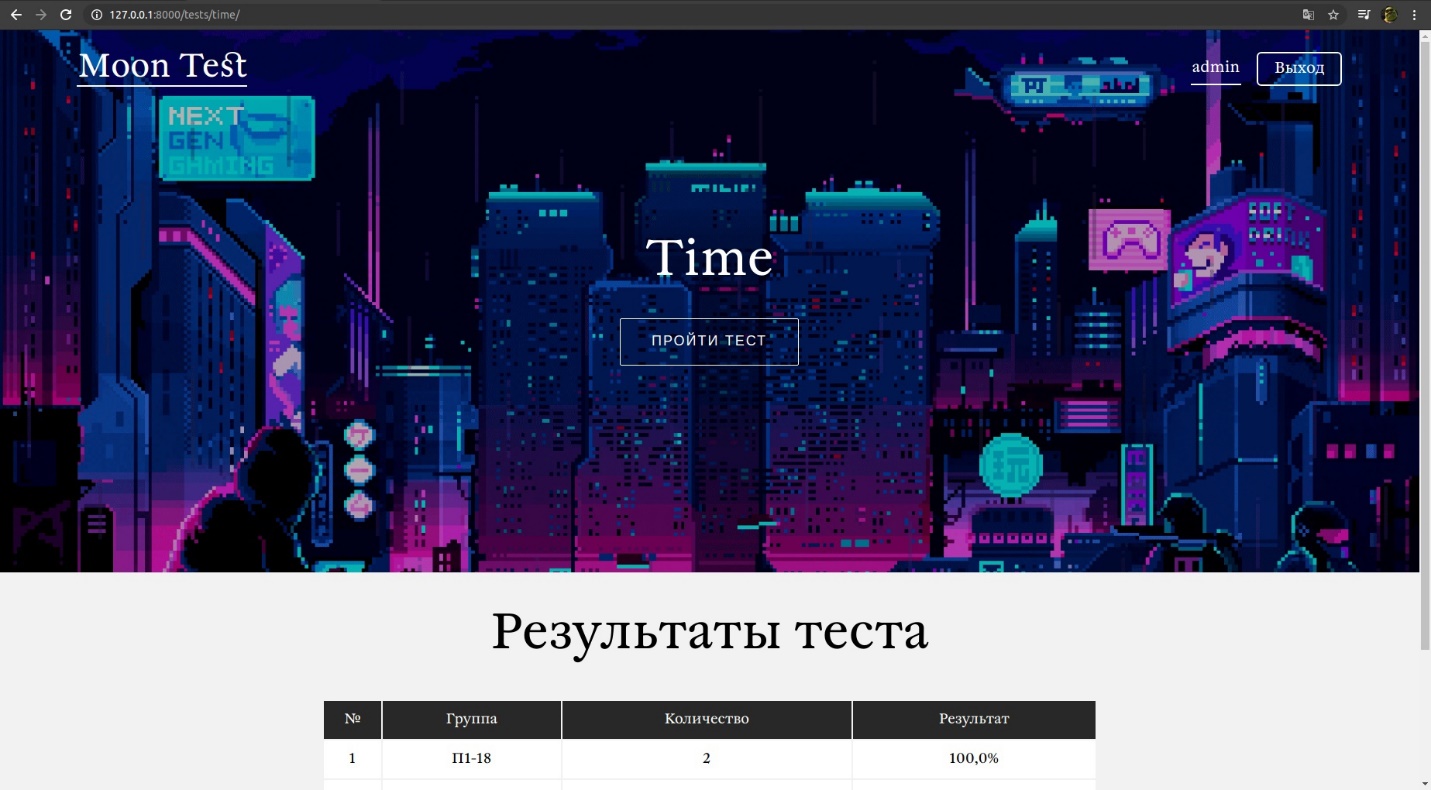


Рисунок 13. Страница теста

Теперь вам представлен вопросы к тесту. Чтобы их листать нажимайте кнопки «Далее» и «Назад».

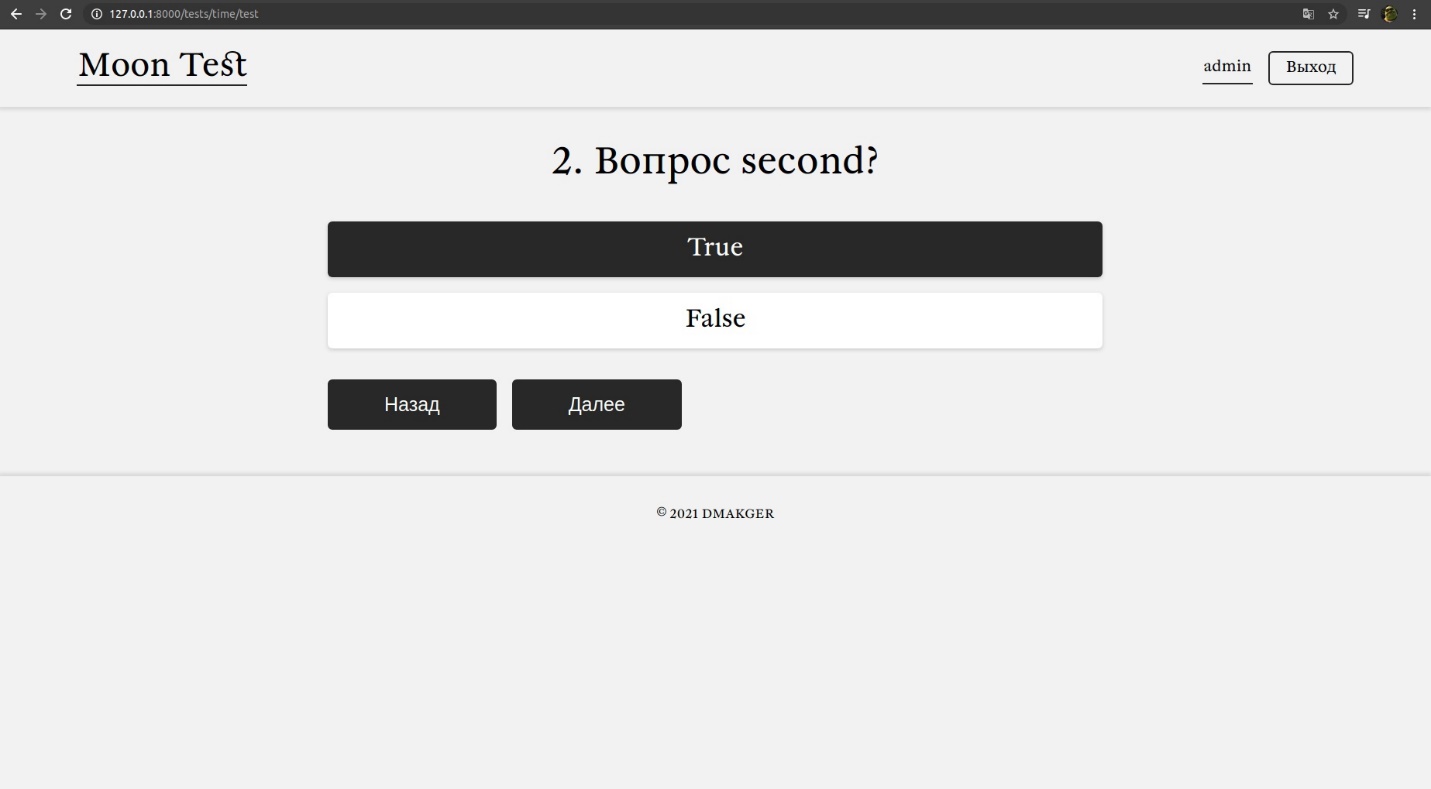


Рисунок 14. Вопросы к тесту

Для завершения теста нажмите кнопку, «Закончить тест». Она появляется при открытии последнего вопроса:

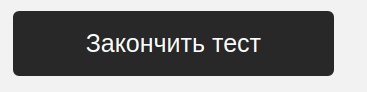


Рисунок 15. Кнопка завершения теста

После завершения теста вы можете увидеть процентное количество правильных ответов, а также таблицу результатов по группам.

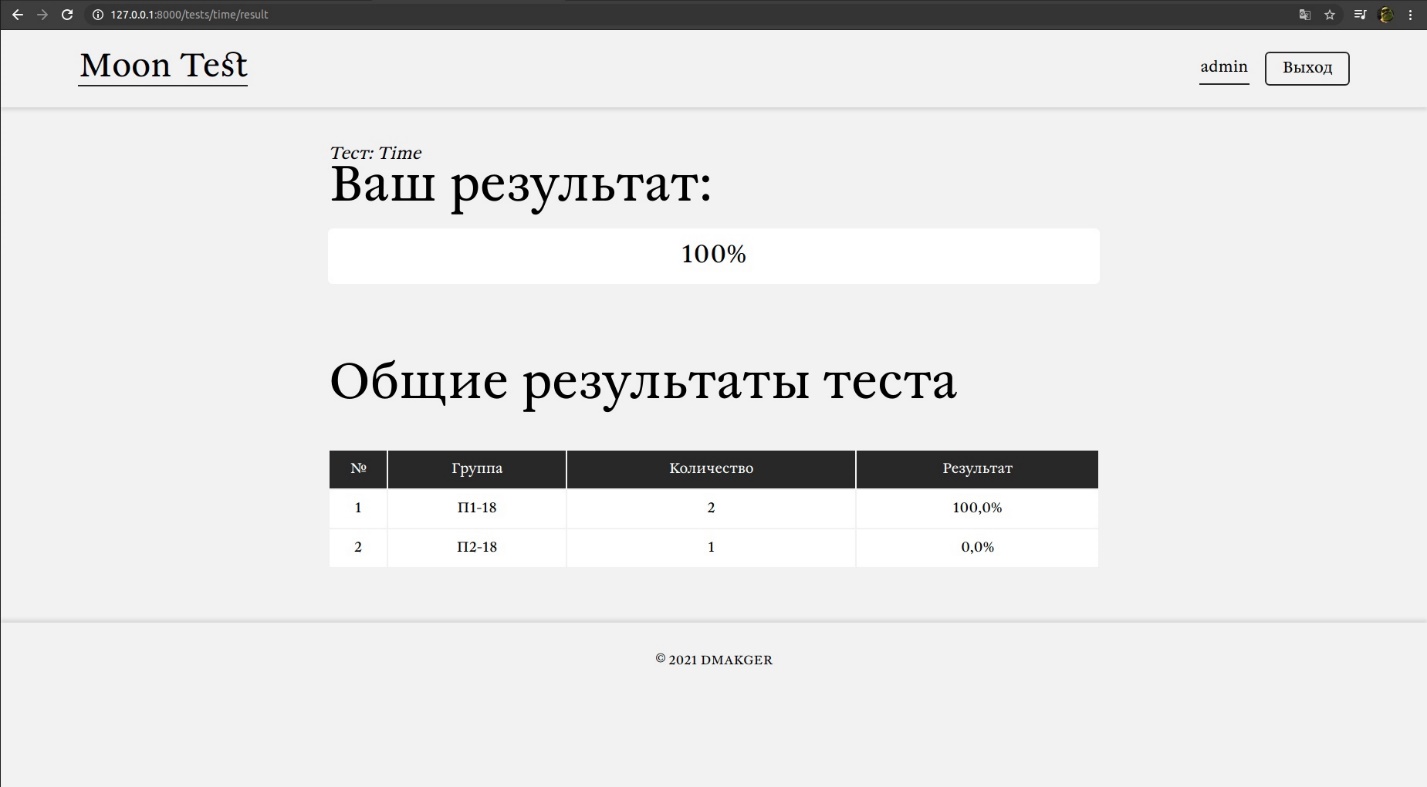


Рисунок 16. Результат теста

На любом из этих этапов, вы можете вернуться на главный экран. Для этого вам нужно нажать на логотип.



Рисунок 17. Логотип

### Сообщение оператору

При неправильном вводе данных в форму регистрации или авторизации появиться сообщение с допущенной ошибкой:

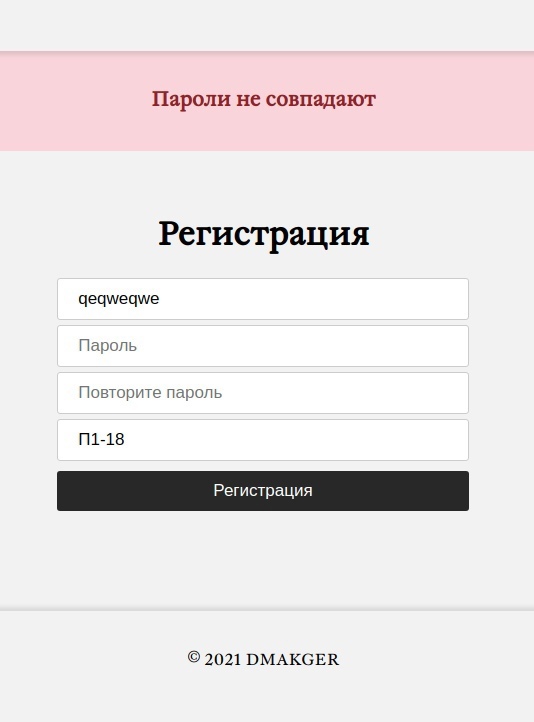


Рисунок 18. Сообщение об ошибке

**Причина:** неправильное заполнение форм.

**Действия сайта:** сайт не регистрирует/авторизует пользователя.

**Действия оператора:** правильно заполнить форму.

## To-Do лист

* Добавить профиль пользователю
* Пагинацию
* Возможность создавать каждому пользователю тест на глобальном и локальном уровне.
* Добавить другие виды тестов
* Создание опросников

**Заключение**

Благодаря данному сайту проводить тестирование станет намного легче, так как не придётся тратить бумагу, чернила на написание теста. Также решение теста возможно будет в режиме реального времени из любой точки мира, то есть дистанционно. Данный проект может быть использован в учебных целях, а также для обучения созданию динамических сайтов.

# Список литературы и интернет ресурсов

1. 16Personalities

https://www.16personalities.com/

1. testometrika

<https://testometrika.com/tests/>

1. Викиум

https://wikium.ru/

1. HTML

<https://skillbox.ru/media/code/chto_takoe_html/>

1. **CSS**

<https://programfiles.info/css/chto-takoe-css/>

1. JavaScript

<https://promo.ingate.ru/seo-wikipedia/java-script/>

1. Django

<https://developer.mozilla.org/ru/docs/Learn/Server-side/Django/Introduction>

1. Visual Studio Code

<https://code.visualstudio.com/docs>

1. Lucid – сайт для построения диаграмм

<https://lucid.app/>

**Приложение 1. views.py**

from django.http.response import HttpResponseRedirect

from .forms import SignUpForm

from django.shortcuts import render

from django.views import View

from .models import Human

from django.contrib.auth import login

class SignUpView(View):

def get(self, request, \*args, \*\*kwargs):

form = SignUpForm() # Форма на вход

return render(request, 'moon\_test\_app/signup.html', context={

'form': form,

})

def post(self, request, \*args, \*\*kwargs):

form = SignUpForm(request.POST) # Форма на регистрацию

# Если данные валидны

if form.is\_valid():

user = form.save() # Сохраняем пользователя в бд

# Если пользователь существует, то логинем этого пользователя

if user is not None:

human = dict()

human = Human(

name = request.POST['username'],

group = request.POST['group']

)

human.save()

login(request, user)

return HttpResponseRedirect('/') # Возращение на главную страницу

return render(request, 'moon\_test\_app/signup.html', context={

'form': form,

})

**Приложение 2. views.py**

from django.http.response import HttpResponseRedirect

from .forms import SignInForm

from django.shortcuts import render

from django.views import View

from django.contrib.auth import login

from django.contrib.auth import authenticate

class SignInView(View):

def get(self, request, \*args, \*\*kwargs):

form = SignInForm() # Форма на вход

return render(request, 'moon\_test\_app/signin.html', context={

'form': form,

})

def post(self, request, \*args, \*\*kwargs):

form = SignInForm(request.POST) # Форма на регистрацию

# Если данные валидны

if form.is\_valid():

username = request.POST['username']

password = request.POST['password']

user = authenticate(request, username=username, password=password)

# Если пользователь существует, то логинем этого пользователя

if user is not None:

login(request, user)

return HttpResponseRedirect('/')

return render(request, 'moon\_test\_app/signin.html', context={

'form': form,

})

**Приложение 3. views.py**

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

from django.views import View

from .models import Test, ResultTest

class TestDetailView(View):

def get(self, request, slug, \*args, \*\*kwargs):

test = get\_object\_or\_404(Test, url=slug)

result\_groups = ResultTest().get\_result\_group(test.id)

return render(request, 'moon\_test\_app/test\_detail.html', context={

'test': test,

'groups': result\_groups

})

**Приложение 4. views.py**

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

from django.views import View

from .models import Test

class TestQuestionsView(View):

def get(self, request, slug, \*args, \*\*kwargs):

test = get\_object\_or\_404(Test, url=slug)

questions = dict()

question = dict()

a\_true = dict()

count = 0

number = 1

for row in test.questions.split('\n'):

if row[-1] == '\r':

row = row[:-1]

if row == '':

questions[number] = question

question = dict()

count = -1

number += 1

else:

if count == 0:

question['q'] = row

else:

if row[0] == '\*':

row = row[1:]

a\_true[number] = count

if not question.get('a'):

question['a'] = dict()

question['a'][count] = row

count += 1

if question:

questions[number] = question

number += 1

return render(request, 'moon\_test\_app/test\_qa.html', context={

'questions': questions,

'a\_true': a\_true,

'test': test,

})

**Приложение 5. views.py**

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

from django.views import View

from .models import Test, Human, ResultTest

class TestResultView(View):

def get(self, request, \*args, \*\*kwargs):

test = ResultTest() # Форма на вход

return render(request, 'moon\_test\_app/<slug>/test\_result.html', context={

'test': test,

})

def post(self, request, slug, \*args, \*\*kwargs):

print('---')

print(request.POST)

# Получаем список ответов пользователя. (При этом удалив username и csrfmiddlewaretoken)

result = [key for key in sorted(request.POST.values()) if key != '' and key[0].isdigit()]

# Формируем список правельных ответов

a\_true = list()

test = get\_object\_or\_404(Test, url=slug)

count = 0

number = 1

for row in test.questions.split('\n'):

if row[-1] == '\r':

row = row[:-1]

if row == '':

count = -1

number += 1

elif row[0] == '\*':

a\_true.append(f"{number}-{count}")

count += 1

# Кол-во правельных ответов

n\_answer\_true = 0

for answer in result:

if answer in a\_true:

n\_answer\_true += 1

# Процент правильных ответов

percent\_true\_answer = round(n\_answer\_true/len(a\_true) \* 100, 2)

# Если незарегистрированный пользователь

if request.POST['username'] == '':

result\_test = {

'test\_id': test.id,

'result': percent\_true\_answer,

}

# Если зарегистрированный пользователь

else:

result\_test = ResultTest.objects.filter(name=request.POST['username'], test\_id=test.id)

print(result\_test)

# Если пользователь ещё не проходил тест, то регистрируем его

if not result\_test:

print('not')

group = Human.objects.filter(name=request.POST['username'])

# Пользователь не указал свою группу

if not group:

group = 'Аноним'

# Пользователь указал свою группу

else:

group = group[0].group

result\_test = ResultTest(

name=request.POST['username'],

test\_id=test.id,

group=group,

result=percent\_true\_answer,

)

# Пользователь уже проходил тест

else:

print('yes')

result\_test = result\_test[0]

result\_test.result = percent\_true\_answer

# Если десятичние числа равны 0, то приводим к int

if result\_test.result == int(result\_test.result):

result\_test.result = int(result\_test.result)

result\_test.save()

return render(request, 'moon\_test\_app/test\_result.html', context={

'test': test,

'result\_test': result\_test,

'all\_result\_test': ResultTest().get\_result\_group(test.id)

})