Министерство образования Республики Беларусь

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Факультет компьютерных систем и сетей

Кафедра информатики

Отчет по практике 15.04.2023

“Разработка сайта для детской школы искусств”

Студенты Cнежко Максим, гр. 254505

Дмитрук Богдан, гр. 253505

Сапроненко Вячеслав, гр. 253504

Минск 2023

# **СОДЕРЖАНИЕ**

**1.**[**ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ 2**](https://docs.google.com/document/d/1h9DE0wA4KGrKD-9JCX62elV_-ry4ejZrDOGbSE70Ams/edit#heading=h.vooml75z5ldu)

**2**[**. BACKEND**](https://docs.google.com/document/d/1h9DE0wA4KGrKD-9JCX62elV_-ry4ejZrDOGbSE70Ams/edit#heading=h.irwd9xl5sdmo) **6**

2[.1](https://docs.google.com/document/d/1h9DE0wA4KGrKD-9JCX62elV_-ry4ejZrDOGbSE70Ams/edit#heading=h.irwd9xl5sdmo) Архитектура backend6

2[.2 Структура проекта](https://docs.google.com/document/d/1h9DE0wA4KGrKD-9JCX62elV_-ry4ejZrDOGbSE70Ams/edit#heading=h.irwd9xl5sdmo) 7

2[.3 Работа с запросами](https://docs.google.com/document/d/1h9DE0wA4KGrKD-9JCX62elV_-ry4ejZrDOGbSE70Ams/edit#heading=h.irwd9xl5sdmo) 7

2[.4 Авторизация](https://docs.google.com/document/d/1h9DE0wA4KGrKD-9JCX62elV_-ry4ejZrDOGbSE70Ams/edit#heading=h.irwd9xl5sdmo) 8

2[.5 Подключение БД к проекту](https://docs.google.com/document/d/1h9DE0wA4KGrKD-9JCX62elV_-ry4ejZrDOGbSE70Ams/edit#heading=h.irwd9xl5sdmo) 8

2[.6 Модели](https://docs.google.com/document/d/1h9DE0wA4KGrKD-9JCX62elV_-ry4ejZrDOGbSE70Ams/edit#heading=h.irwd9xl5sdmo) 9

2[.7 forms.py](https://docs.google.com/document/d/1h9DE0wA4KGrKD-9JCX62elV_-ry4ejZrDOGbSE70Ams/edit#heading=h.irwd9xl5sdmo) 9

2[.8 Разграничение доступа](https://docs.google.com/document/d/1h9DE0wA4KGrKD-9JCX62elV_-ry4ejZrDOGbSE70Ams/edit#heading=h.irwd9xl5sdmo) 10

2[.9 Добавленный функционал](https://docs.google.com/document/d/1h9DE0wA4KGrKD-9JCX62elV_-ry4ejZrDOGbSE70Ams/edit#heading=h.irwd9xl5sdmo) 12

2[.10](https://docs.google.com/document/d/1h9DE0wA4KGrKD-9JCX62elV_-ry4ejZrDOGbSE70Ams/edit#heading=h.irwd9xl5sdmo) Планируемые добавления в ближайшее время13

### [**ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ**](https://docs.google.com/document/d/1h9DE0wA4KGrKD-9JCX62elV_-ry4ejZrDOGbSE70Ams/edit#heading=h.vooml75z5ldu)

| Backend | Это та часть веб-приложения или программного обеспечения, которая работает на стороне сервера и отвечает за обработку запросов от клиентской части, хранение и обработку данных, взаимодействие с базами данных и другими сервисами |
| --- | --- |
| База данных (Database) | Это совокупность данных, организованных в определенном формате, который позволяет эффективно хранить, обрабатывать и управлять данными. Базы данных используются для хранения информации в компьютерной системе и могут содержать информацию о людях, товарах, заказах, финансах, научных данных и т.д. |
| Python | Python - это интерпретируемый язык программирования с динамической типизацией, который широко используется для разработки различных приложений, включая веб-приложения, научные вычисления, игры и т.д. |
| Django | Django - это фреймворк для создания веб-приложений на языке Python. С помощью Django возможно создание широкого диапазона веб-приложений, включая: социальные сети, онлайн-магазины,Блоги и новостные сайты,Управление проектами,Управление проектами,Платформы для обмена сообщениями,Мобильные приложения |
| Модуль для базы данных (Database module) | это компонент программного обеспечения, который позволяет приложению взаимодействовать с базой данных. Модуль предоставляет интерфейс для выполнения запросов к базе данных и получения результатов |
| HTML | HTML — стандартизированный язык гипертекстовой разметки документов для просмотра веб-страниц в браузере. |
| Сервер | Сервер приложений, который передаёт "динамические" запросы между сайтом и веб-сервером. |
| Web-сервер | Web-сервер: обрабатывает входящие HTTP-запросы от клиентов и отправляет им соответствующие HTTP-ответы. |
| MySQL | MySQL — свободная реляционная система управления базами данных. |

| WSGI-сервер | WSGI-сервер: принимает HTTP-запросы от Web-сервера и преобразует их в объекты Django, с которыми Django может работать. |
| --- | --- |
| Клиентский код | Клиентский код: это модуль, который содержит код на стороне клиента, который отправляет HTTP-запросы на Web-сервер. |
| Модели | Модели: это модуль Django, который определяет структуру базы данных и методы для работы с данными. Модели могут взаимодействовать с базой данных, а также содержать бизнес-логику. |
| **Класс** | Класс — в объектно-ориентированном программировании, представляет собой шаблон для создания объектов, обеспечивающий начальные значения состояний: инициализация полей-переменных и реализация поведения функций или методов. |

# **1. BACKEND**

### **1.1 Архитектура backend**

На рисунке 1 изображена схема взаимодействия модулей backend

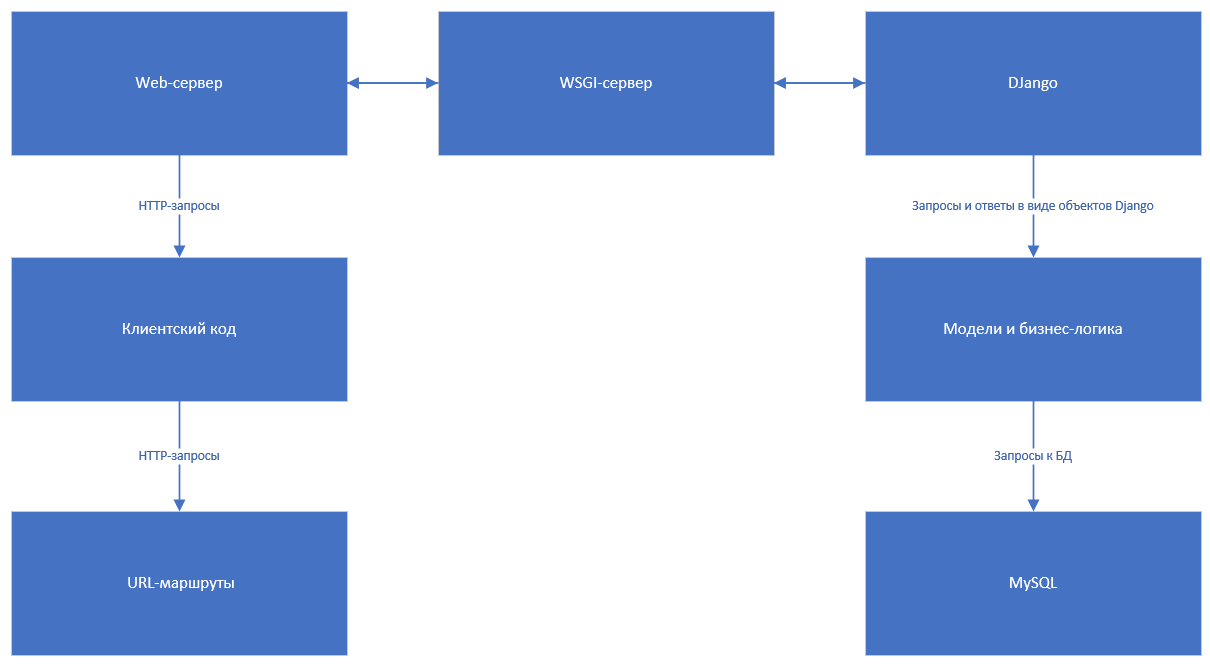


Рисунок 1

## **1.2 Структура проекта**

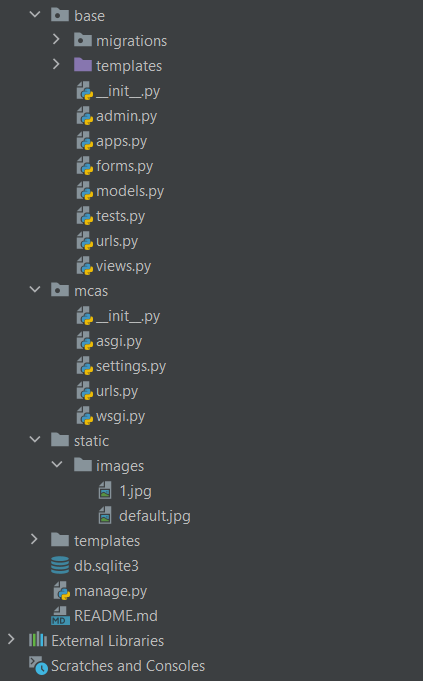


Рисунок 2

**1.3 Работа с запросами**

При получении запроса HTTP на сервере происходит проверка авторизации, был ли пользователь авторизован. Если нет, то запрос будет отклонен. Если всё успешно, то происходит проверка прав: на этом этапе сервер проверяет, есть ли у пользователя необходимые права доступа для выполнения запроса. Если у пользователя нет необходимых прав, то запрос будет отклонен. Дальше обработка запроса: на этом этапе сервер определяет, какой вид запроса был получен (POST, PUT, GET или DELETE). Eсли запрос является GET-запросом, на этом этапе сервер получает данные из базы данных.

Добавление данных: если запрос является POST-запросом, на этом этапе сервер добавляет новые данные в БД.

Изменение данных: если запрос является PUT-запросом, на этом этапе сервер изменяет данные в базе данных.

Удаление данных: если запрос является DELETE-запросом, на этом этапе сервер удаляет данные из базы данных.

Отправка ответа: на этом этапе сервер отправляет ответ клиенту в формате

HTTP. Ответ может содержать данные (например, при GET-запросе) или просто статус выполнения запроса (например, при PUT-запросе).

**1.4 Авторизация**

Регистрация пользователей производится администрацией. Администрация регистрирует персонал и учащихся, присваивает им уникальный логин и пароль, информацию о персонале и учениках, после чего всё это заносится в БД. После, администратор может выдавать зарегистрированным пользователям логин и пароль, с помощью которого они смогут авторизоваться на сайте.

## **1.5 Подключение БД к проекту**

На рисунке 2 изображен способ подключения БД к проекту

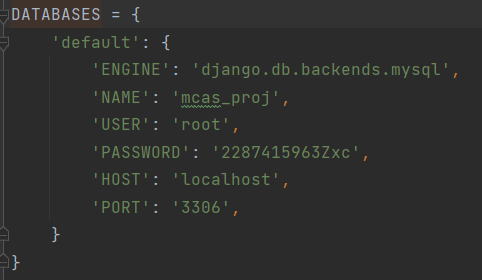


Рисунок 3

**1.6 Модели**

Данные модели являются классами, которые используются в различных функциях сайта.

Класс User - это перегруженная стандартная модель для создания нового пользователя. Наследуется от стандартного класса джанго AbstractUser. Содержит поля имени пользователя, кем он является, его аватар.

Класс New - это модель для создания новостной статьи. Был добавлен ckeditor для возможности детального редактирования текста, добавления изображений и специальных символов.

Класс About - это модель, хранящая в себе общую информацию о школе: заслуги школы, информацию про учителей, общие сведения о школе.

Класс Message - это модель для создания сообщений администрации, которые будут отображаться у персонала школы в зависимости от настроек сообщения.

Класс File - это модель для загрузки файлов учителями в музыкальную библиотеку и их скачки учениками.

Все модели наследуются от models.Model(в Django является базовым классом для создания моделей. Модели используются для определения структуры базы данных и представления данных в приложении Django.), кроме User.

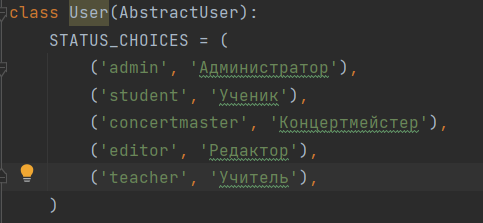
## **1.7 forms.py**

forms.py - это файл в Django, который используется для определения форм, которые используются для сбора данных от пользователей веб-сайта. Формы в Django представляют собой классы Python, которые определяют поля формы и правила проверки данных, введенных пользователем.

В нём содержаться классы, которые выполняют внешнее отображение полей, которые пользователи могут заполнить для создания какой-либо модели. Там прописаны поля, которые будут показаны пользователю

## **1.8 Разграничение доступа**

Было добавлено 5 типов пользователей



Каждый тип имеет свои права и обязанности:

* Ученики имеют доступ к общедоступной части сайта и, после авторизации к странице “Дневник ученика”, где размещена такая их личная информация как расписание, домашнее задание, расписание звонков, расписание каникул и ведомость с оценками по всем предметам.
* Концертмейстер имеет доступ к общедоступной части сайта и, после авторизации, к личному кабинету концертмейстера. Там ему будет доступна для просмотра такая информация как расписание(Дата занятия, время, кабинет, ученик, которому он аккомпанирует и инструмент, на котором играет ученик), табло сообщений от администрации, его методический паспорт, ссылка на документ коллективного договора, общее расписание школы(документ), календарь академических концертов и экзаменов, в которых он принимает участие.
* Редактор имеет доступ к общедоступной части сайта и, после авторизации, будет иметь возможность открыть новостную страницу в режиме редактирования, откуда сможет перейти на страницу создания статьи(ее интерфейс будет представлен в наброске, интерфейс интуитивно понятен).
* Учитель имеет доступ к общедоступной части сайта и, после авторизации, к своему личному кабинету, где ему доступно табло сообщений от администрации, просмотр календарей воспитательной, методической и академической работы в виде виджета и документа, переход на страницы методического паспорта авторизованного пользователя, взаимопосещаемости авторизованного пользователя, журналу авторизованного пользователя, к общему расписанию школы(документ), к контингенту школы, к форме, позволяющей зарезервировать время для проведения урока в малом или большом зале(содержит такие фильтры как флаги выбора одного из двух залов, поле для ввода даты занятия, выпадающий список, в котором можно будет выбрать конкретное время-дату из доступных), также на страницы работы с музыкально одаренными и профориентированными учащимися, содержащие таблицу содержащую фио учеников и справочную информацию по их музыкальной деятельности(содержание данной страницы ещё обсуждается с администрацией школы). С данным уровнем доступа появляется возможность добавлять учебную литературу на страницу музыкальной библиотеки при помощи специальной формы, а также открывается доступ к методическому календарю отделения, к контингенту отделения, ко страницам по работе с профориентированными и одаренными учениками всего отделения.
* Администратор имеет доступ к общедоступной части сайта и, после авторизации, к своему личному кабинету. Оттуда он может перейти, помимо всех упоминавшихся выше страниц, к

1. Списку всех учителей школы, странице подобной “Весь контингент”, но содержащей фио учителей. При нажатии на имя учителя пользователь с уровнем доступа “Администратор” может перейти к странице, содержащей личную информацию учителя. и редактировать ее содержимое
2. Странице контроля посещений концертов и внеклассных мероприятий(редактируемая таблица)
3. К общему расписанию школы, представленному в виде таблицы-виджета и отдельной ссылки на документ.
4. Форме создания сообщения, которое потом увидят нужные учителя, на виджете “Табло сообщений от администрации”, содержащей поле для ввода текста сообщения и флаги отделений, оформленные в выпадающий список, учителя которых потом увидят это сообщение.
5. Трудовым договорам(страница содержащая список ссылок на документы)
6. Коллективному договору(ссылка на документ)

Также посетители имеют доступ только к общедоступной части сайта. Это Главная страница, музыкальная библиотека (могут скачать документы, расположенные там), могут посетить страницы отделений, но из всего интерфейса им будет доступен только просмотр информации для посетителей, будут видны ссылки на две последние новости и график сдачи академических экзаменов.

## **1.9 Добавленный функционал**

На данном этапе разработки были добавлены основные возможности для “внешней” работы сайта, такие как: создание, редактирование и отображение общей информации на главной странице сайта, создание, редактирование и удаление новостных статей, редактирование контактной информации, была создан каркас музыкальной библиотеки с ее основным функционалом(добавление и скачка MIDI файлов). Также была добавлена регистрация новых пользователей, распределение каждого пользователя по статусу(выдача уровня доступа производится администрацией школы). Для профилей персонала и обучающихся была переделана модель User для возможности загрузки аватара, биографии учителя, договоров(PDF файлов), использования Ф.И.О для отображения информации на сайте. Были выданы специальные возможности для пользователей определенного уровня доступа(описано выше).

## 

## **1.10 Планируемые добавления в ближайшее время**

В скором времени будет добавлен календарь событий и возможность бронирования концертных залов, а также список экзаменов, редактируемый администрацией и учителями.

# **2. FRONTEND**

# **3. TESTS**

**3.1 Тестирование представлений**

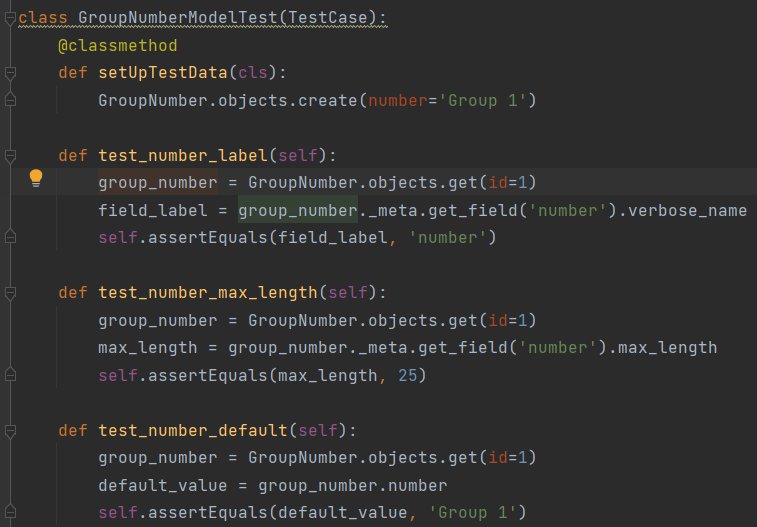
# **Инструмент тестирования: Тестовый клиент**

Тестовый клиент Django - это инструмент, который позволяет тестировать веб-приложения Django без запуска сервера. Он предоставляет удобный способ проверки того, как ваше приложение обрабатывает запросы и возвращает ответы, а также проверяет работу функций и методов, написанных в приложении. Тестовый клиент может использоваться для написания юнит-тестов, интеграционных тестов и функциональных тестов. Он позволяет проверять различные аспекты работы веб-приложения, такие как обработка запросов, правильность вывода данных, обработка ошибок и т.д.

**3.1.1 Подготовка к тестированию моделей**

Для тестирования представлений нужны тестовые записи в базе данных. Для этого переопределим метод **setUp**, который является методом инициализации и вызывается перед каждым тестом. В этом методе создадим объекты моделей пользователя, сообщений, категорий, файлов, статей и ивентов, заданий, групп, а также создадим зарегистрированного клиента. Эти объекты будут впоследствии использоваться в тестировании представлений.

**3.1.2 Тестирование модели GroupNumber**

**GroupNumber** - модель, которая связывает домашку, учителей и учеников по группам, чтобы, например, когда препод будет рассылать домашнее задание, он укажет группу, а обучающимся, которые в этой группе прилетит это домашнее задание.

Данный тест **GroupNumberModelTest** проверяет корректность работы модели **GroupNumber**.

Тесты проверяют:

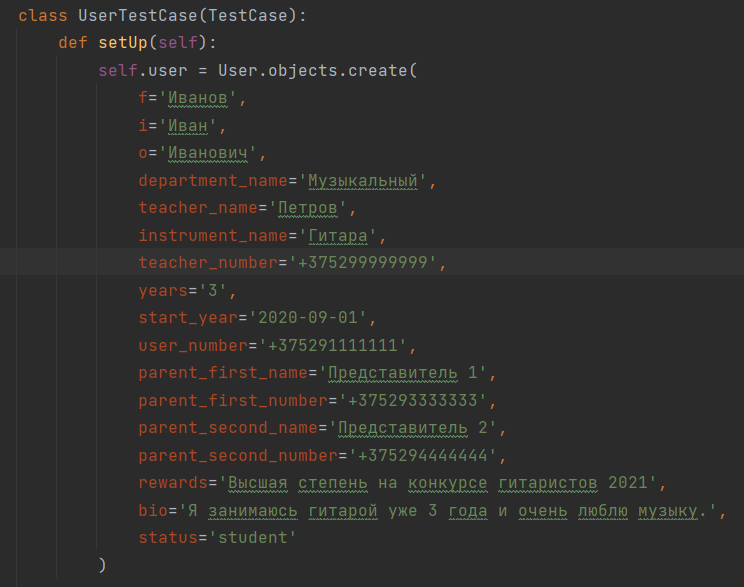
* правильность поля **number**,
* правильность максимальной длины поля “number”, которое должно быть максимум 25 символов,
* правильность значения поля **number** по умолчанию, которое должно быть равно **Group 1**.

Таким образом, эти тесты позволяют убедиться, что модель **GroupNumber** корректно определена, а поля в ней настроены правильно

**3.1.3 Тестирование модели User**

Модель **User** - это перегруженная стандартная модель для создания нового пользователя. Наследуется от стандартного класса джанго **AbstractUser**. Содержит поля имени пользователя, кем он является, его аватар.

Данный набор тестов проверяет, что модель **User** была корректно создана с заданными полями и связями.







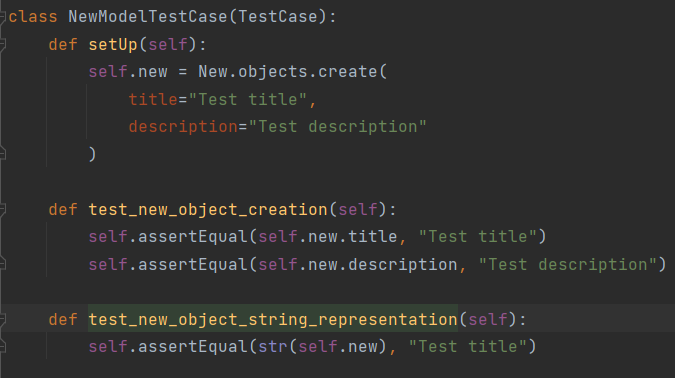
Тесты проверяют:

* **test\_user\_str** проверяет, что метод **str** модели возвращает ожидаемое строковое представление, в данном случае “Иванов Иван Иванович”.
* **test\_user\_full\_name** проверяет, что метод **get\_full\_name** модели возвращает ожидаемое значение, в данном случае также "Иванов Иван Иванович".
* **test\_user\_department\_name**, **test\_user\_teacher\_name**, **test\_user\_instrument\_name**, **test\_user\_teacher\_number**, **test\_user\_years**, **test\_user\_user\_number**, **test\_user\_parent\_first\_name**, **test\_user\_parent\_first\_number**, **test\_user\_parent\_second\_name**, **test\_user\_parent\_second\_number**, **test\_user\_rewards**, **test\_user\_bio**, **test\_user\_status** проверяют, что поля модели имеют ожидаемые значения, заданные изначально.
* **test\_user\_avatar** проверяет, что связь модели с изображением (поле **avatar**) возвращает ожидаемый URL-адрес, в данном случае **/media/images/profile-pictures/default.jpg**.

Таким образом, эти тесты позволяют убедиться в том, что модель **User** была корректно создана и имеет правильную структуру и данные внутри.

**3.1.4 Тестирование модели New**

Модель **New** - это модель для создания новостной статьи. Был добавлен **ckeditor** для возможности детального редактирования текста, добавления изображений и специальных символов.



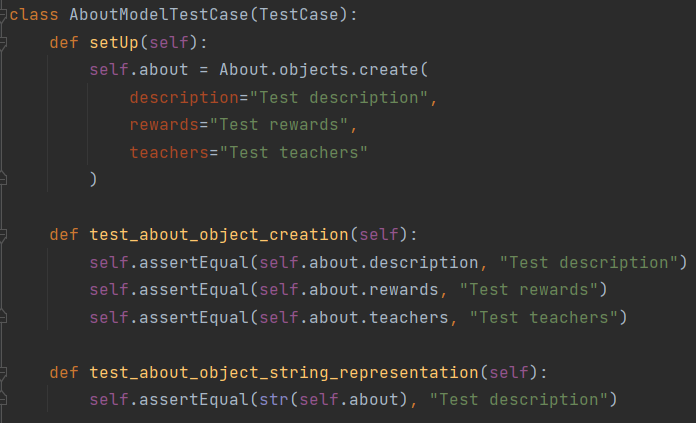
Тесты проверяют:

* **test\_new\_object\_creation** - тест создания объекта модели **New**. Проверяет, что созданный объект содержит правильные данные. В данном случае, что при создании объекта в базе данных были установлены правильные значения для полей **title** и **description**.
* **test\_new\_object\_string\_representation** - тест проверки строкового представления объекта модели **New**. Проверяет, что когда объект выводится как строка, возвращается правильный заголовок новости (**title**).

Таким образом, эти тесты позволяют убедиться в том, что модель **New** была корректно создана и имеет правильную структуру и данные внутри.

**3.1.5 Тестирование модели About**

Модель **About** - это модель, хранящая в себе общую информацию о школе: заслуги школы, информацию про учителей, общие сведения о школе.



Тесты проверяют:

* **test\_about\_object\_creation -** тест, в котором производится проверка того, что объект модели **About** был создан с корректными параметрами: **description**, **rewards**, **teachers**.
* **test\_about\_object\_string\_representation -** тест, в котором проверяется корректность представления объекта модели **About** в виде строки. В данном случае ожидается, что строковое представление объекта будет равно значению поля **description**.

Таким образом, эти тесты позволяют убедиться в том, что модель **About** была корректно создана и имеет правильную структуру и данные внутри.

**3.1.6 Тестирование модели Message**

Модель **Message** - это модель для создания сообщений администрации, которые будут отображаться у персонала школы в зависимости от настроек сообщения.



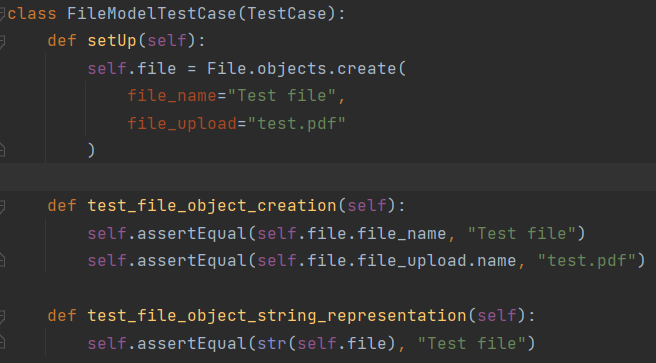
Тесты проверяют:

* **test\_message\_object\_creation -** проверяет, что атрибуты **user** и **body** созданного сообщения соответствуют тестовым данным.
* **test\_message\_object\_string\_representation -** проверяет, что строковое представление объекта сообщения возвращает ожидаемую строку, в данном случае - значение атрибута **body**.

Таким образом, эти тесты позволяют убедиться в том, что модель **Message** была корректно создана и имеет правильную структуру и данные внутри.

**3.1.7 Тестирование модели File**

Модель **File** - это модель для загрузки файлов учителями в музыкальную библиотеку и их скачки учениками.



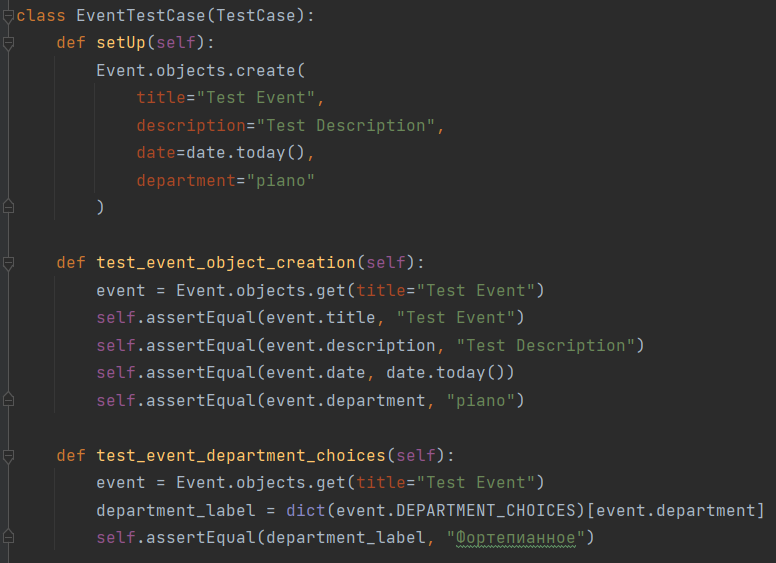
Тесты проверяют:

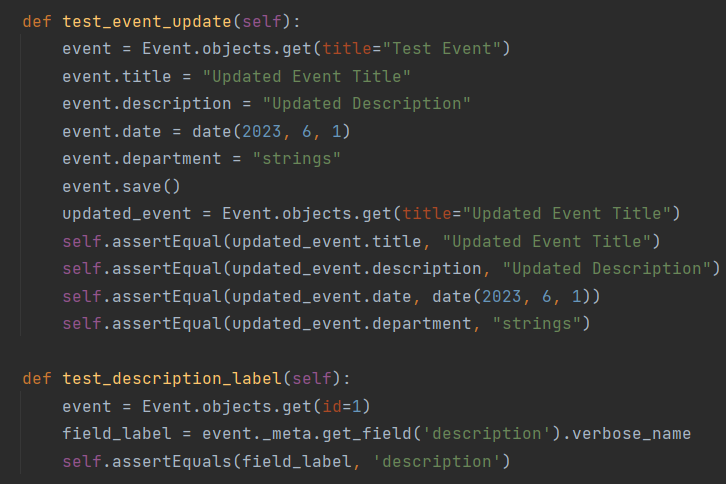
* **test\_file\_object\_creation -** тест, в котором проверяется создание объекта **File** с заданными полями. В этом тесте создается объект **File**, а затем проверяется, что все его поля были корректно заполнены.
* **test\_file\_object\_string\_representation -** тест, в котором проверяется, что строковое представление объекта **File** возвращает ожидаемый результат. В данном случае, возвращается значение поля **file\_name**.

Таким образом, эти тесты позволяют убедиться в том, что модель **File** была корректно создана и имеет правильную структуру и данные внутри.

**3.1.8 Тестирование модели Event**

Модель **Event** - модель для создания администрацией событий для разных отделений, которые потом передаются на страницу соответствующего отделения и отображаются в календаре.







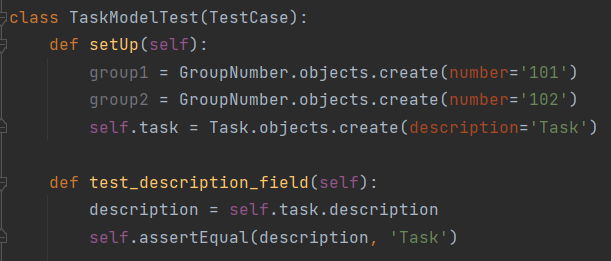
Тесты проверяют:

* **test\_event\_object\_creation** - тест, который проверяет создание объекта **Event** с заданными полями. В этом тесте создается объект **Event**, а затем проверяется, что все его поля были корректно заполнены.
* **test\_event\_department\_choices** - тест, который проверяет, что поле **department** объекта **Event** содержит одно из заданных значений выбора, а именно "piano". Поле **department** содержит строковое значение, но выбор задается в виде кортежей, поэтому для проверки используется словарь, чтобы получить соответствующее текстовое значение.
* **test\_event\_update проверяет** - тест, который показывает что обновление объекта **Event** происходит корректно. В этом тесте объект **Event** сначала извлекается из базы данных, затем его поля обновляются, и объект сохраняется в базе данных. Затем создается новый объект **Event** с обновленными полями, и проверяется, что все поля были корректно обновлены.
* **test\_description\_label, test\_date\_label**, и **test\_department\_label** - тесты, которыe проверяют, что каждое поле модели **Event** имеет корректную метку. Эти тесты используют метод **verbose\_name**, чтобы получить метки полей, которые заданы внутри класса модели.
* **test\_title\_max\_length** и **test\_department\_max\_length** - тесты, которые проверяют, что максимальная длина для каждого поля соответствует заданному значению. Эти тесты используют методы **max\_length**, чтобы получить максимальную длину для соответствующего поля.

Таким образом, эти тесты позволяют убедиться в том, что модель **Event** была корректно создана и имеет правильную структуру и данные внутри.

**3.1.9 Тестирование модели Task**

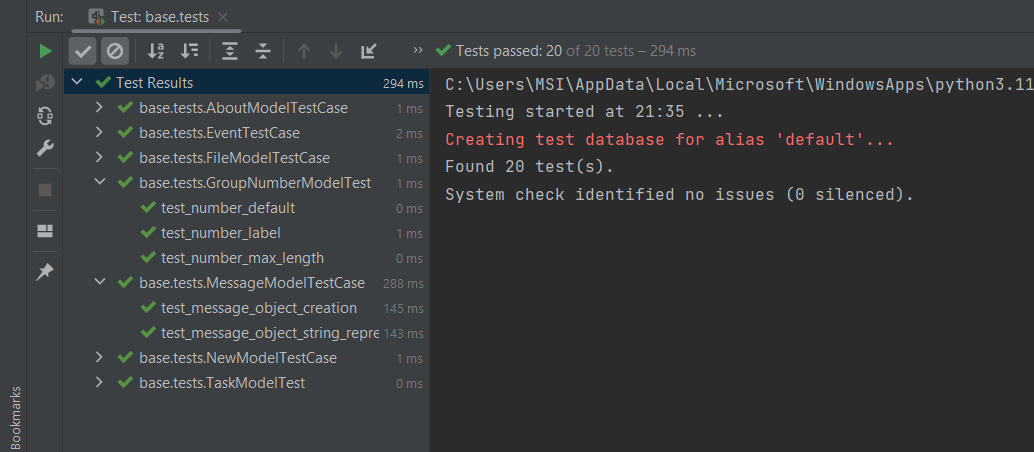
Модель **Task -** это модель для домашнего задания, которая сохраняет в себе дату, на которое надо это домашнее задание сделать, номер урока, к какому его надо сделать, само задание, групповое оно или нет и если нет, то ученика, которому это домашнее задание надо отослать. Потом все задания фильтруются при передаче их из БД по нужным критериям и отдаются на отображение фронту



Тест проверяeт:

* **test\_description\_field** проверяется, что поле **description** объекта **Task** было корректно заполнено значением **Task**.

**3.1.10 Результат**



Таким образом, эти тесты позволяют убедиться в том, что модель **Task** была корректно создана и имеет правильную структуру и данные внутри.

**3.2 Тестирование URL-адресов**

**3.2.1 Подготовка к тестированию URL-адресов**

Тестирование URL-адресов (URL testing) веб-приложения позволяет убедиться в том, что все маршруты (routes), используемые приложением, корректно настроены и работают так, как ожидается.

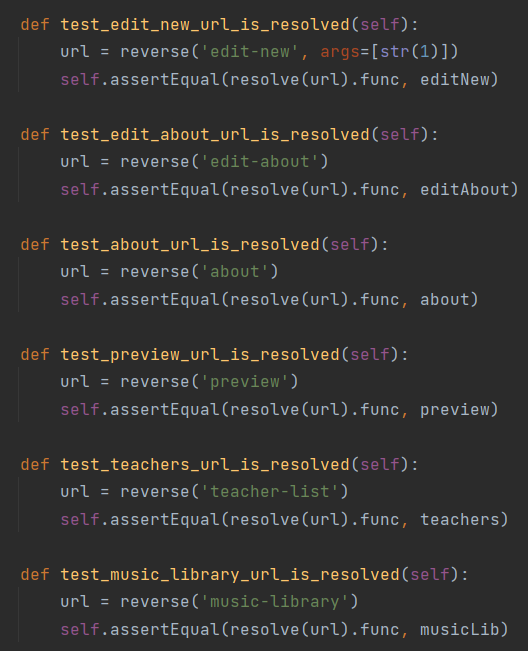
Тестирование URL-адресов может включать в себя проверку следующих аспектов:

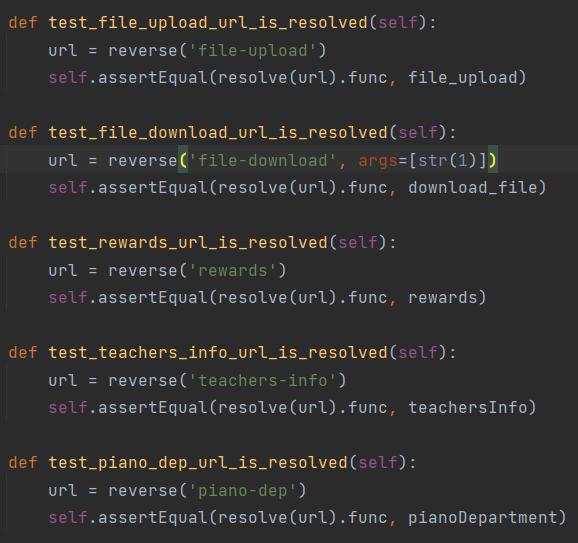
* проверка корректной настройки URL-адресов (например, проверка регулярных выражений);
* проверка доступности URL-адресов (например, проверка ответа на запросы GET и POST);
* проверка соответствия ожидаемому поведению при обращении к URL-адресам (например, проверка редиректов и вызова правильных контроллеров).

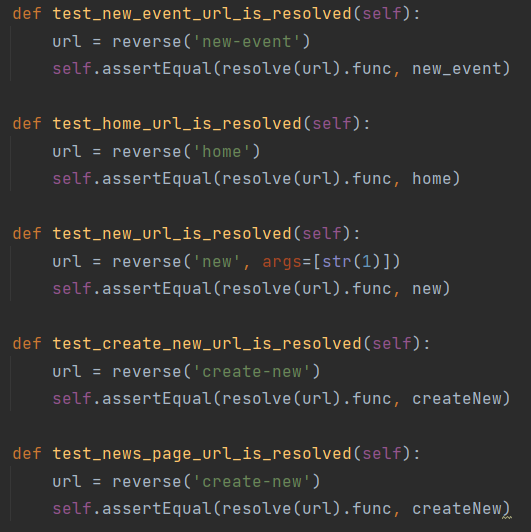
Тестирование URL-адресов важно, так как некорректная настройка маршрутов может привести к непредсказуемому поведению приложения, а также к проблемам с безопасностью.

**3.2.2 Тестирование URL-адресов**



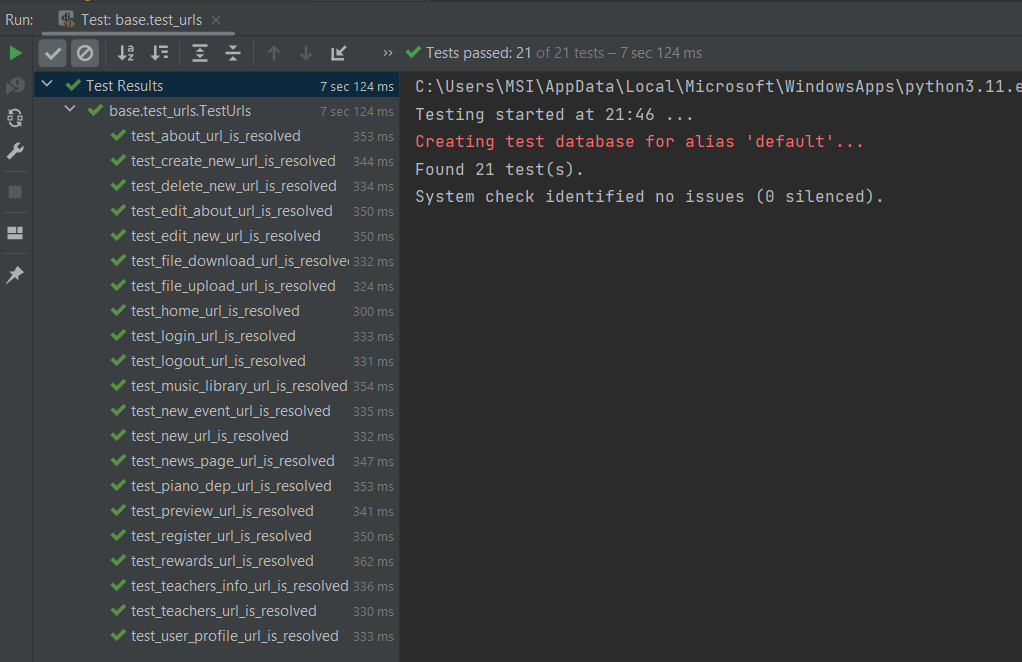






Эти тесты проверяют, что все URL-адреса, используемые в приложении, имеют соответствующие представления, которые обрабатывают запросы по этим URL-адресам. Они используют функцию **reverse** для получения URL-адреса из его имени и, затем, используют функцию **resolve** для определения функции представления, связанной с этим URL-адресом. В результате тесты проверяют, что функции представлений, связанные с каждым URL-адресом, соответствуют ожиданиям.

**3.2.3 Результат**

****

**3.3.1 Тестирование форм**

Для тестирования представлений нужны тестовые записи в базе данных. Для этого переопределим метод **setUp**, который является методом инициализации и вызывается перед каждым тестом. В этом методе создадим словари с верными входными данными.

Тестирование форм используется для проверки правильности работы форм, которые используются для ввода и отправки данных на сервер веб-приложения. Формы могут быть различных типов, например, формы регистрации, формы авторизации, формы поиска и т.д.

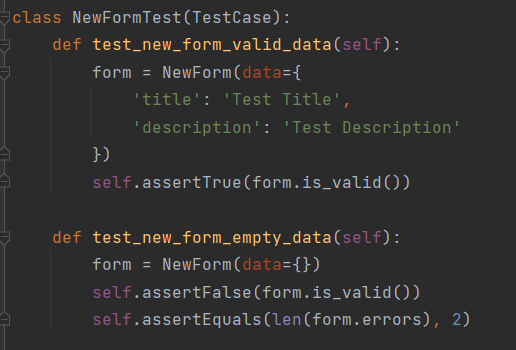
Тестирование форм позволяет проверить корректность работы следующих аспектов:

* Проверка корректности валидации данных на стороне клиента и сервера.
* Проверка обработки ошибок ввода.
* Проверка корректности возвращаемых значений и сообщений об ошибках.
* Проверка корректности отправки данных на сервер.
* Проверка корректности обработки отправленных данных на сервере.
* Проверка корректности работы форм в различных условиях, таких как: с различными комбинациями входных данных, с различными параметрами настройки формы и т.д.

Правильное тестирование форм помогает улучшить пользовательский опыт и повысить качество веб-приложения.

**3.3.2 Тестирование формы NewForm**

Форма **New -** это форма для создания новостной статьи, там прописывается заголовок и описание(для описания подключен специальный виджет **CKEditorWidget**).

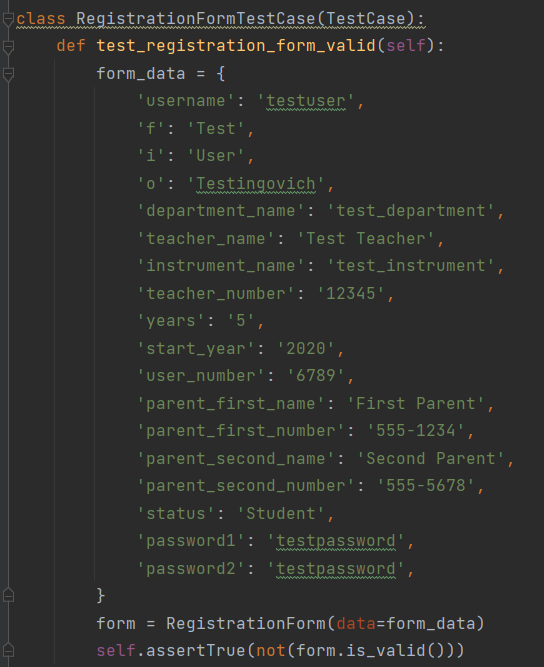


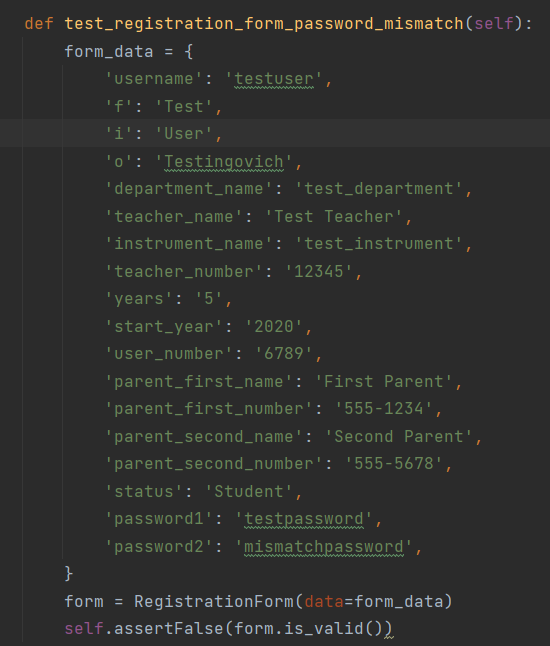
Тесты проверяют:

* **test\_new\_form\_valid\_data** - тест, который проверяет, что форма работает правильно, если в нее передаются правильные данные: создается экземпляр формы с правильными данными и возвращает true при вызове **is\_valid**.
* **test\_new\_form\_empty\_data** – тест, который проверяет, что форма работает правильно, если ей передаются пустые данные: создается экземпляр формы без данных, и **is\_valid** возвращает false и выводит 2 ошибки, соответствующие двум полям формы: “This field is required.”

**3.3.3 Тестирование формы RegistrationForm**

Форма **RegistrationForm -** это форма для администрации, чтобы она могла создавать новых пользователей(статус и все остальные поля).



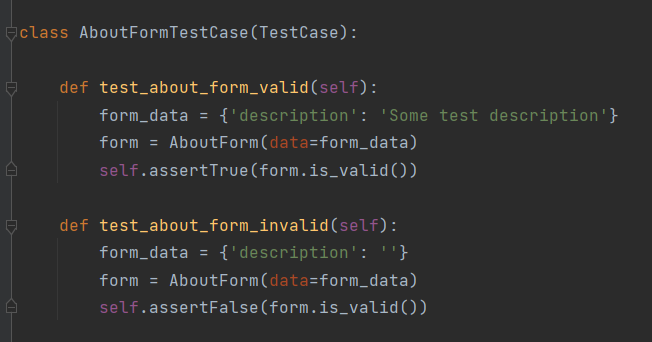


Тесты проверяют:

* **test\_registration\_form\_valid** – тест, который проверяет, что форма с правильными данными проходит валидацию. Для этого форме передаются корректные данные, и ожидается, что результат вызова метода **is\_valid** будет true.
* **test\_registration\_form\_password\_mismatch -** тест, который проверяет, что форма не проходит валидацию, если введенные пароли не совпадают. Для этого форме передаются данные, в которых два поля паролей (**password1** и **password2**) содержат разные значения. Ожидается, что результат вызова метода **is\_valid** будет false, так как введенные пароли не совпадают.

**3.3.4 Тестирование формы AboutForm**

Форма **AboutForm -** специальная форма для заполнения общей информации о школе, что нибудь в таком духе



Тесты проверяют:

**test\_about\_form\_valid** – тест, который проверяет, что форма считается действительной при передаче допустимых данных в виде заполненной строки в поле **description**. То есть ожидается, что форма будет проходить проверку на валидность при передаче правильных данных.

**test\_about\_form\_invalid** - тест, который проверяет, что форма считается недействительной при передаче неправильных данных в виде пустой строки в поле description. То есть ожидается, что форма не будет проходить проверку на валидность при передаче неверных данных.

**3.3.5 Тестирование формы FileUploadForm**

Форма **FileUploadForm** и **FileUUploadForm -** это две формы для загрузки файлов, одна для загрузки файлов в музыкальную библиотеку, второе для загрузки файлов администрации для персонала школы.



Тесты проверяют:

* **test\_file\_upload\_form\_invalid** - тест, который проверяет, что форма не является допустимой, если данные отсутствуют.
* **test\_file\_upload\_form\_save** - тест, который проверяет, что форма является допустимой и сохраняет загруженный файл, если переданы файл и имя файла.
* **test\_file\_upload\_form\_invalid** - тест, который проверяет, что форма не является допустимой, если передано только имя файла, без самого файла.

**3.3.6 Тестирование формы EventCreationForm**

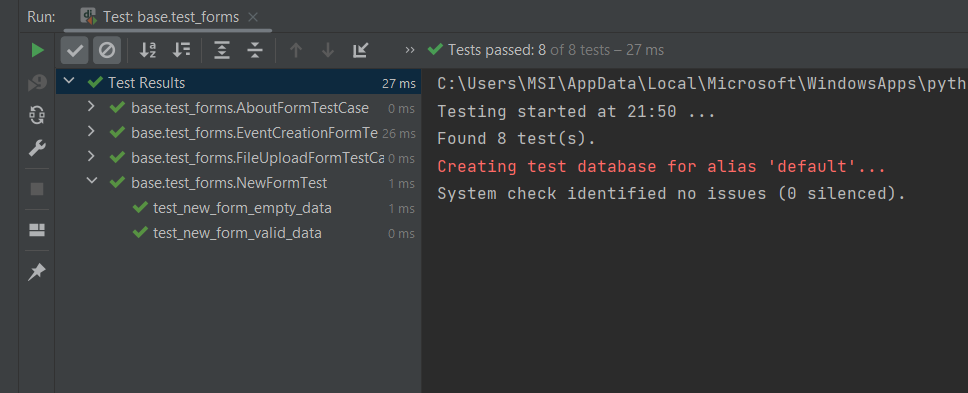
Форма **EventCreation -** это форма для создания события для календаря для каждого отдельного отделения.



Тесты проверяют:

* **test\_event\_creation\_form\_valid -** тест, который проверяет, что форма проходит валидацию с корректными данными. В данном случае, форме передаются корректные значения для полей **title**, **description**, **date** и **department**, и ожидается, что форма будет считаться валидной.
* **test\_event\_creation\_form\_invalid -** тест, который проверяет, что форма не проходит валидацию с некорректными данными. В данном случае, форме передаются некорректные значения для поля **title** путем передачи пустой строки, а все остальные поля имеют корректные значения. Ожидается, что форма будет считаться невалидной.

**3.3.7 Результат**

****