Проект Яндекс лицей

Автор: Коротков Илья

Преподаватель: Шайдуров Игорь Александрович

Группа: 1Д1

Дата: 04.05.2025

Название: “Школьный дневник”

Создано на основе Flask

Реализованы: Сервисы для трех ролей (учитель, ученик, администратор). API для сторонних запросов. Синхронизация с api от Яндекса.

В коде запускающим файлом является “app.py”. Все зависимости находятся в “req.txt”

Запустив сайт, вы попадаете в главное меню веб-приложения, где вы можете увидеть две большие кнопки, для перенаправления На две роли - Ученик и Учитель, речь о них пойдет дальше. Ниже располагается блок с яндекс картами, на которых можно найти любую точку мира. В самом низу возле надписи “© 2025 Школьный дневник” Можно заметить ссылку для роли Администратор.

**Ученики:**

После перехода на путь /student/ или нажатии на кнопку “Для Учеников” на главной странице, вы заметите надпись:

“Добро пожаловать в Школьный дневник

Здесь вы можете просматривать свое расписание, оценки и домашние задания.”

В верхнем правом углу есть кнопка входа. От роли ученик Логин: drozdov3000, Пароль: password123

Слева будет меню с выбором (при нажатии неавторизованным пользователем на любой из пунктов, система потребует войти):

* Оценки и посещаемость
* Расписание
* Домашнее задание
* Одноклассники
* Настройки профиля

При нажатии на “Оценки и посещаемость” вы попадете на страницу с выбором предмета, по которому вы хотите посмотреть оценки и свою посещаемость.

При нажатии на “расписание” вы попадете на страницу с таблицей вашего расписания уроков.

При нажатии на “Домашнее задание” вы увидите список всех домашних заданий, с датой их сдачи.

При нажатии на “Одноклассники” вы увидите список учеников в вашем классе (Имя, Фамилия, Номер телефона и электронную почту).

При нажатии на “Настройки профиля” вы попадете на страницу с данными вашего профиля и вы будете вправе их изменить. После замены данных следует нажать на кнопку “Сохранить”. На этой странице вы также можете получить свой личный api-ключ для сторонних запросов (Об этом будет позже).

**Учителя:**

После перехода на путь /teacher/ или нажатии на кнопку “Для Учителей” на главной странице, вы заметите надпись:

“Добро пожаловать в Учительскую

Здесь вы можете ставить оценки, посещаемость и смотреть свое расписание”

У учителей есть две роли: Предметник и Завуч. Пердметник - учитель, который ведет уроки и может быть классным руководителем. Завуч - обладает всеми правами предметника, но дополнительно может управлять классами (Получить доступ ко всем классам школы, создавать их, добавлять в них учеников, удалять классы)

Сверху справа есть кнопка входа. От роли учитель(завуч) Логин: teacher1, Пароль: password123. От роли учитель(предметник) Логин: teacher5, Пароль: password123

Слева будет меню с выбором (при нажатии неавторизованным пользователем на любой из пунктов, система потребует войти):

– Ведомые классы

– Ваш класс

– Ваше расписание

– Настройки профиля

**Завуч:**

При нажатии на “Ведомые классы” или переходе по пути /teacher/classes откроется панель со всеми классами школы. Снизу каждого класса будет две кнопки “Добавить ученика” и “Удалить”.

При нажатии на “Добавить ученика” появится форма для добавления ученика в конкретный класс. Первое поле - логин ученика(username), второе поле - Имя, третье - Фамилия, четвертое - Email, пятое - Пароль, шестое - Дата рождения (в виде календаря), седьмое - Номер телефона, восьмое - Адрес проживания. Ниже две кнопки, одна добавить ученика, другая отменит это действие.

На странице с классами, выше списка есть кнопка “Добавить класс”, после нажатия на нее откроется форма с добавлением класса. Первое поле - Название класса, второе - Выбор учителя (в виде списка). Если нет свободного учителя (без классного руководства) вы не сможете создать класс. Ниже будет две кнопки “Создать” и “Отменить”. Одна создаст класс, другая отменит это действие.

Если в списке классов нажать на название класса, то откроется выбор всех предметов из их расписания. После нажатия на “Выбрать” под нужным предметом, вас перебросит на страницу с таблицей посещаемости и оценок и полями для добавления домашнего задания. Выше таблицы есть кнопка “Скачать отчет в Excel”, после нажатия на нее скачается таблица с информацией об этом классе, по выбранному предмету. В таблице есть три колонки

|  |  |  |
| --- | --- | --- |
| **ФИО ученика** | **Оценки** | **Посещаемость** |

В таблице на сайте с оценками и посещаемостью можно добавлять, менять и удалять отметки учеников и изменять состояние посещаемости. если нажать на кнопку “Добавить дату” появится окно с выбором даты, которое создаст новую колонку с этой датой в таблице. Ниже таблицы - поле для добавления домашнего задания и выбором даты его сдачи. После нажатия на кнопку “Добавить ДЗ”, оно появится у учеников. Также ниже можно видеть все домашние задания, что были заданы ранее.

При нажатии на “Ваш класс” в меню или при переходе на /teacher/my\_class вы увидите информацию о вашем классе: Таблицу со всей информацией об учениках и таблицу с расписанием уроков у вашего класса. Выше таблиц находится кнопка “скачать полный отчет по классу в Excel”. после нажатия на нее, скачается Excel таблица с четырмя листами: “Расписание”, “Оценки”, “Посещаемость”, “Ученики”. На листе ”Расписание” будет расписание вашего класса. На листе “Оценки” Будут оценки ваших учеников, в столбцах оценки по определенным предметам. На листе “Посещаемость” таблица с посещаемостью. На листе “Ученики” информация об учениках.  
  
При нажатии на “Ваше расписание” Будут таблицы с вашим расписанием по дням недели. Выше таблиц - кнопка “Скачать расписание в формате Excel”. При нажатии на нее скачивается таблица с вашим расписанием.

При нажатии на “Настройки профиля” вы попадете на страницу с данными вашего профиля и вы будете вправе их изменить. После замены данных следует нажать на кнопку “Сохранить”. На этой странице вы также можете получить свой личный api-ключ для сторонних запросов.

**Предметник:**

Учитель-предметник похож на завуча, только не имеет доступа ко всем классам (только те, что ведет по расписанию). На странице выбора предмета для класса, так же не все предметы (только те, что учитель может вести).

**Администраторы:**

После перехода на страницу /admin/ или нажатии на скрытую кнопку, вы увидите:

«Вы в админ панели.

Здесь вы - Бог (этого сайта).»

Сверху справа есть кнопка входа. От роли администратор Логин: admin1, Пароль: password123.  
  
Слева будет меню с выбором (при нажатии неавторизованным пользователем на любой из пунктов, система потребует войти):

- Ученики

- Учителя

- Расписание

- Предметы

- Настройки профиля

При нажатии на «Ученики» вы увидите плиточки с учениками (их фото, фамилия имя и класс). Последняя плиточка для добавления ученика. После нажатия на любого ученика появляется его профиль где можно изменить все что нужно, либо удалить ученика. А при нажатии на добавления ученика так же появится форма для создания. Выше плиточек находится строка поиска, где можно найти любого ученика. Можно так же скачать учеников в формате excel.  
  
Тоже самое, что и с учениками с учителями. Так же плиточки, добавление, изменение, удаление, поиск.  
  
После нажатия на «Расписание» появится список со всем расписанием всех классов. Есть кнопки редактировать, удалить, для редактирования и удаления записей расписания. Выше списка есть разные строки для фильтрации расписания (По ФИ учителя, классу, предмету, времени проведения, дню недели). Снизу есть кнопки для добавления записей расписания и скачивания Excel файла с ним.   
  
!!!ПРИ СКАЧИВАНИИ EXCEL ФАЙЛОВ СКАЧИВАЮТСЯ, ОНИ ТАК ЖЕ С ФИЛЬТРАМИ!!!!  
  
При нажатии на «Предметы» появляется список со всеми предметами с возможностью добавления предметов, изменением, удалением и скачиванием с Excel.

При нажатии на “Настройки профиля” вы попадете на страницу с данными вашего профиля и вы будете вправе их изменить. После замены данных следует нажать на кнопку “Сохранить”. На этой странице вы также можете получить свой личный api-ключ для сторонних запросов.

**Документация API приложения "Школьный дневник"**

**Общее описание**

API приложения "Школьный дневник" предоставляет функциональность для трех ролей пользователей:

* **Ученик**: доступ к расписанию, оценкам, посещаемости, домашним заданиям и информации об одноклассниках.
* **Учитель**: доступ к расписанию, предметам, классам, оценкам, посещаемости, домашним заданиям, а также управление этими данными (для завуча — дополнительные права на управление классами).
* **Администратор**: управление данными учеников, учителей, расписания, предметов и экспорт данных в Excel.

**Базовый URL**

Базовый URL для всех запросов: http://127.0.0.1:5000

**Авторизация**

Все запросы к API требуют заголовок X-API-Key с действительным API-ключом, который пользователь может получить в разделе "Настройки профиля". Примеры ключей:

* Ученик: db1586ba-11c1-4acf-9665-85d0e47d4d88
* Учитель (предметник): 1db011ad-1d3f-436c-9721-1d05584d3d4c
* Учитель (завуч): 58323e61-ce12-4418-b340-8d757073b21e
* Администратор: f3caa8cb-57dc-4c16-9726-7e58818a625a

Если ключ отсутствует или недействителен, API вернет ошибку с кодом 401 (Unauthorized) или 403 (Forbidden).

**Формат ответов**

API возвращает ответы в формате JSON. Успешные запросы имеют статус 200 (OK) или 201 (Created), а ответы содержат:

* status: "success" или "error".
* data: данные, возвращаемые запросом (при успехе).
* description или message: описание ошибки или дополнительная информация.

Ошибки возвращаются с соответствующими HTTP-кодами:

* 400: Bad Request (неверный формат запроса).
* 401: Unauthorized (отсутствует или недействителен API-ключ).
* 403: Forbidden (недостаточно прав).
* 404: Not Found (ресурс не найден).
* 500: Internal Server Error (ошибка на сервере).

**Тестирование API**

Для тестирования используйте библиотеку requests в Python. Пример базовой настройки:

import requests

BASE\_URL = "http://127.0.0.1:5000"

API\_KEY = "db1586ba-11c1-4acf-9665-85d0e47d4d88" *# Замените на нужный ключ*

headers = {"X-API-Key": API\_KEY}

**API для учеников**

API для учеников предоставляет доступ к информации о расписании, оценках, посещаемости, домашнем задании и одноклассниках. Все запросы требуют API-ключ с ролью ученика.

**1. Получение расписания ученика**

**Эндпоинт**: GET /api/student/schedule  
**Описание**: Возвращает расписание уроков для класса ученика, отсортированное по дням недели и предметам.  
**Параметры**: Нет.  
**Заголовки**: X-API-Key: <api\_key>  
**Пример запроса**:

response = requests.get(f"{BASE\_URL}/api/student/schedule", headers=headers)

print(response.json())

**Пример ответа**:

{

"status": "success",

"data": {

"class\_id": 1,

"schedule": [

{

"schedule\_id": 1,

"subject\_id": 1,

"subject\_name": "Математика",

"teacher\_id": 1001,

"teacher\_name": "Иван Иванов",

"day\_of\_week": "Понедельник",

"start\_time": "09:00",

"end\_time": "10:00"

}

]

}

}

**Коды ошибок**:

* 404: Студент не найден.
* 401: Недействительный API-ключ.
* 403: Пользователь не является студентом.

**2. Получение оценок и посещаемости**

**Эндпоинт**: GET /api/student/grades\_attendance/<int:subject\_id>  
**Описание**: Возвращает оценки, посещаемость и средний балл ученика по указанному предмету.  
**Параметры**:

* subject\_id (path): ID предмета.  
  **Заголовки**: X-API-Key: <api\_key>  
  **Пример запроса**:

response = requests.get(f"{BASE\_URL}/api/student/grades\_attendance/1", headers=headers)

print(response.json())

**Пример ответа**:

{

"status": "success",

"data": {

"student\_id": 3000,

"subject\_id": 1,

"subject\_name": "Математика",

"grades": [

{

"grade\_id": 1,

"grade": 5,

"date": "2025-05-10"

}

],

"average\_grade": 4.5,

"attendance": [

{

"attendance\_id": 1,

"date": "2025-05-10",

"status": "присутствовал"

}

]

}

}

**Коды ошибок**:

* 404: Студент или предмет не найдены.
* 401: Недействительный API-ключ.
* 403: Пользователь не является студентом.

**3. Получение домашнего задания**

**Эндпоинт**: GET /api/student/homework  
**Описание**: Возвращает список домашних заданий для класса ученика.  
**Параметры**: Нет.  
**Заголовки**: X-API-Key: <api\_key>  
**Пример запроса**:

response = requests.get(f"{BASE\_URL}/api/student/homework", headers=headers)

print(response.json())

**Пример ответа**:

{

"status": "success",

"data": {

"class\_id": 1,

"homeworks": [

{

"homework\_id": 1,

"subject\_id": 1,

"subject\_name": "Математика",

"task": "Решить задачи 1-5",

"due\_date": "2025-05-15"

}

]

}

}

**Коды ошибок**:

* 404: Студент не найден.
* 401: Недействительный API-ключ.
* 403: Пользователь не является студентом.

**4. Получение списка одноклассников**

**Эндпоинт**: GET /api/student/classmates  
**Описание**: Возвращает список одноклассников (имя, фамилия, телефон, email) для класса ученика.  
**Параметры**: Нет.  
**Заголовки**: X-API-Key: <api\_key>  
**Пример запроса**:

response = requests.get(f"{BASE\_URL}/api/student/classmates", headers=headers)

print(response.json())

**Пример ответа**:

{

"status": "success",

"data": {

"class\_id": 1,

"classmates": [

{

"user\_id": 3001,

"first\_name": "Анна",

"last\_name": "Смирнова",

"phone\_number": "+1234567890",

"email": "anna@example.com"

}

]

}

}

**Коды ошибок**:

* 404: Студент не найден.
* 401: Недействительный API-ключ.
* 403: Пользователь не является студентом.

**API для учителей**

API для учителей предоставляет доступ к расписанию, предметам, классам, оценкам, посещаемости и домашним заданиям. Учителя с ролью "Завуч" имеют дополнительные права на управление классами. Все запросы требуют API-ключ с ролью учителя.

**1. Получение расписания учителя**

**Эндпоинт**: GET /api/teachers/schedule  
**Описание**: Возвращает расписание уроков учителя, отсортированное по дням недели и времени.  
**Параметры**: Нет.  
**Заголовки**: X-API-Key: <api\_key>  
**Пример запроса**:

response = requests.get(f"{BASE\_URL}/api/teachers/schedule", headers=headers)

print(response.json())

**Пример ответа**:

{

"status": "success",

"data": {

"teacher\_id": 1001,

"schedule": [

{

"schedule\_id": 1,

"class\_id": 1,

"class\_name": "10А",

"subject\_id": 1,

"subject\_name": "Математика",

"day\_of\_week": "Понедельник",

"start\_time": "09:00",

"end\_time": "10:00",

"day\_order": 1

}

]

}

}

**Коды ошибок**:

* 404: Учитель не найден.
* 401: Недействительный API-ключ.
* 403: Пользователь не является учителем.

**2. Получение списка предметов для класса**

**Эндпоинт**: GET /api/teachers/classes/<int:class\_id>/subjects  
**Описание**: Возвращает список предметов для указанного класса. Для завуча — все предметы, для обычного учителя — только его предметы.  
**Параметры**:

* class\_id (path): ID класса.  
  **Заголовки**: X-API-Key: <api\_key>  
  **Пример запроса**:

response = requests.get(f"{BASE\_URL}/api/teachers/classes/1/subjects", headers=headers)

print(response.json())

**Пример ответа**:

{

"status": "success",

"message": "Найдено 2 предметов",

"data": [

{

"subject\_id": 1,

"subject\_name": "Математика"

},

{

"subject\_id": 2,

"subject\_name": "Физика"

}

]

}

**Коды ошибок**:

* 403: Пользователь не является учителем.
* 404: Класс не найден.
* 401: Недействительный API-ключ.
* 500: Ошибка сервера.

**3. Получение списка классов**

**Эндпоинт**: GET /api/teachers/classes  
**Описание**: Возвращает список классов, к которым у учителя есть доступ. Завуч видит все классы, обычный учитель — только свои.  
**Параметры**: Нет.  
**Заголовки**: X-API-Key: <api\_key>  
**Пример запроса**:

response = requests.get(f"{BASE\_URL}/api/teachers/classes", headers=headers)

print(response.json())

**Пример ответа**:

{

"status": "success",

"data": [

{

"class\_id": 1,

"class\_name": "10А",

"teacher\_id": 1001,

"teacher\_name": "Иван Иванов"

}

]

}

**Коды ошибок**:

* 401: Недействительный API-ключ.
* 403: Пользователь не является учителем.

**4. Создание класса (только для завуча)**

**Эндпоинт**: POST /api/teachers/classes  
**Описание**: Создает новый класс с указанным названием, учителем и списком учеников.  
**Параметры** (JSON):

* class\_name (string): Название класса (например, "11Б").
* teacher\_id (integer): ID учителя (классного руководителя).
* students (array): Список учеников (каждый с полями username, first\_name, last\_name, email, password, birth\_date, phone\_number, address).  
  **Заголовки**: X-API-Key: <api\_key> (требуется роль завуча).  
  **Пример запроса**:

response = requests.post(

f"{BASE\_URL}/api/teachers/classes",

json={

"class\_name": "11Б",

"teacher\_id": 1002,

"students": [

{

"username": "student1",

"first\_name": "Анна",

"last\_name": "Смирнова",

"email": "anna@example.com",

"password": "password123",

"birth\_date": "2005-01-01",

"phone\_number": "+1234567890",

"address": "123 Street"

}

]

},

headers=headers

)

print(response.json())

**Пример ответа**:

{

"status": "success",

"data": {

"class\_id": 12,

"class\_name": "11Б",

"teacher\_id": 1002,

"students": [

{

"student\_id": 3001,

"username": "student1",

"first\_name": "Анна",

"last\_name": "Смирнова",

"email": "anna@example.com"

}

]

}

}

**Коды ошибок**:

* 403: Недостаточно прав (не завуч).
* 400: Неверный формат данных.
* 401: Недействительный API-ключ.

**5. Управление оценками**

**Эндпоинт**:

* GET /api/teachers/classes/<int:class\_id>/grades: Получение оценок для класса.
* POST /api/teachers/grades: Добавление новой оценки.
* PUT /api/teachers/grades: Обновление существующей оценки.
* DELETE /api/teachers/grades/<int:grade\_id>: Удаление оценки.  
  **Описание**:
* GET: Возвращает список оценок для указанного класса.
* POST: Создает новую оценку для ученика по предмету.
* PUT: Обновляет существующую оценку.
* DELETE: Удаляет оценку по ID.  
  **Параметры**:
* class\_id (path, для GET): ID класса.
* grade\_id (path, для DELETE): ID оценки.
* JSON для POST:
  + student\_id (integer): ID ученика.
  + subject\_id (integer): ID предмета.
  + grade (integer): Оценка (1–5).
  + date (string): Дата в формате "YYYY-MM-DD".
* JSON для PUT:
  + grade\_id (integer): ID оценки.
  + student\_id (integer): ID ученика.
  + subject\_id (integer): ID предмета.
  + grade (integer): Новая оценка (1–5).
  + date (string): Новая дата в формате "YYYY-MM-DD".  
    **Заголовки**: X-API-Key: <api\_key>  
    **Пример запроса (POST)**:

response = requests.post(

f"{BASE\_URL}/api/teachers/grades",

json={

"student\_id": 15,

"subject\_id": 1,

"grade": 5,

"date": "2025-05-12"

},

headers=headers

)

print(response.json())

**Пример ответа (POST)**:

{

"status": "success",

"data": {

"grade\_id": 10,

"student\_id": 15,

"subject\_id": 1,

"grade": 5,

"date": "2025-05-12"

}

}

**Коды ошибок**:

* 400: Неверный формат данных (например, оценка вне диапазона 1–5).
* 403: Недостаточно прав.
* 404: Класс, ученик или предмет не найдены.
* 401: Недействительный API-ключ.

**6. Управление домашними заданиями**

**Эндпоинт**:

* GET /api/teachers/classes/<int:class\_id>/homework: Получение домашних заданий для класса.
* POST /api/teachers/homework: Добавление нового домашнего задания.  
  **Описание**:
* GET: Возвращает список домашних заданий для указанного класса.
* POST: Создает новое домашнее задание.  
  **Параметры**:
* class\_id (path, для GET): ID класса.
* JSON для POST:
  + subject\_id (integer): ID предмета.
  + class\_id (integer): ID класса.
  + task (string): Описание задания.
  + due\_date (string): Дата сдачи в формате "YYYY-MM-DD".  
    **Заголовки**: X-API-Key: <api\_key>  
    **Пример запроса (POST)**:

response = requests.post(

f"{BASE\_URL}/api/teachers/homework",

json={

"subject\_id": 1,

"class\_id": 2,

"task": "Решить задачи 1-5",

"due\_date": "2025-05-15"

},

headers=headers

)

print(response.json())

**Пример ответа (POST)**:

{

"status": "success",

"data": {

"homework\_id": 5,

"subject\_id": 1,

"class\_id": 2,

"task": "Решить задачи 1-5",

"due\_date": "2025-05-15"

}

}

**Коды ошибок**:

* 400: Неверный формат данных.
* 403: Недостаточно прав.
* 404: Класс или предмет не найдены.
* 401: Недействительный API-ключ.

**7. Управление посещаемостью**

**Эндпоинт**:

* GET /api/teachers/classes/<int:class\_id>/attendance: Получение данных о посещаемости для класса.
* POST /api/teachers/attendance: Добавление записи о посещаемости.
* PUT /api/teachers/attendance: Обновление записи о посещаемости.  
  **Описание**:
* GET: Возвращает список записей о посещаемости для указанного класса.
* POST: Создает новую запись о посещаемости.
* PUT: Обновляет существующую запись.  
  **Параметры**:
* class\_id (path, для GET): ID класса.
* JSON для POST:
  + student\_id (integer): ID ученика.
  + date (string): Дата в формате "YYYY-MM-DD".
  + status (string): Статус ("присутствовал" или "отсутствовал").
* JSON для PUT:
  + attendance\_id (integer): ID записи.
  + student\_id (integer): ID ученика.
  + date (string): Дата в формате "YYYY-MM-DD".
  + status (string): Новый статус ("присутствовал" или "отсутствовал").  
    **Заголовки**: X-API-Key: <api\_key>  
    **Пример запроса (POST)**:

response = requests.post(

f"{BASE\_URL}/api/teachers/attendance",

json={

"student\_id": 15,

"date": "2025-05-11",

"status": "присутствовал"

},

headers=headers

)

print(response.json())

**Пример ответа (POST)**:

{

"status": "success",

"data": {

"attendance\_id": 3,

"student\_id": 15,

"date": "2025-05-11",

"status": "присутствовал"

}

}

**Коды ошибок**:

* 400: Неверный формат данных (например, некорректный статус).
* 403: Недостаточно прав.
* 404: Класс или ученик не найдены.
* 401: Недействительный API-ключ.

**8. Экспорт данных в Excel**

**Эндпоинты**:

* GET /api/teachers/classes/<int:class\_id>/<int:subject\_id>/excel: Экспорт отчета по оценкам и посещаемости для класса и предмета.
* GET /api/teachers/my-class/excel: Экспорт полного отчета по классу (расписание, оценки, посещаемость, ученики).
* GET /api/teachers/schedule/excel: Экспорт расписания учителя.  
  **Описание**: Возвращает файл Excel с соответствующими данными.  
  **Параметры**:
* class\_id (path): ID класса.
* subject\_id (path): ID предмета (для первого эндпоинта).  
  **Заголовки**: X-API-Key: <api\_key>  
  **Пример запроса**:

response = requests.get(f"{BASE\_URL}/api/teachers/schedule/excel", headers=headers)

with open("schedule.xlsx", "wb") as f:

f.write(response.content)

**Коды ошибок**:

* 404: Класс или предмет не найдены.
* 401: Недействительный API-ключ.
* 403: Пользователь не является учителем.

**API для администраторов**

API для администраторов предоставляет полный контроль над данными учеников, учителей, расписания и предметов, а также экспорт данных в Excel. Все запросы требуют API-ключ с ролью администратора.

**1. Управление учениками**

**Эндпоинты**:

* GET /api/admin/students/<string:search>: Получение списка учеников с поиском.
* GET /api/admin/student/<int:student\_id>: Получение данных одного ученика.
* POST /api/admin/student: Создание нового ученика.  
  **Описание**:
* GET /students/<search>: Возвращает список учеников, отфильтрованный по поисковому запросу (например, имя или класс).
* GET /student/<student\_id>: Возвращает данные ученика по ID.
* POST: Создает нового ученика.  
  **Параметры**:
* search (path): Поисковый запрос (например, "В" или "10А").
* student\_id (path): ID ученика.
* JSON для POST:
  + username (string): Логин.
  + password (string): Пароль.
  + first\_name (string): Имя.
  + last\_name (string): Фамилия.
  + email (string): Email.
  + phone\_number (string): Телефон.
  + class\_id (integer): ID класса.
  + birth\_date (string): Дата рождения ("YYYY-MM-DD").
  + address (string): Адрес.  
    **Заголовки**: X-API-Key: <api\_key>  
    **Пример запроса (POST)**:

response = requests.post(

f"{BASE\_URL}/api/admin/student",

json={

"username": "newstudent",

"password": "password123",

"first\_name": "New",

"last\_name": "Student",

"email": "newstudent@example.com",

"phone\_number": "+1234567890",

"class\_id": 1,

"birth\_date": "2005-01-01",

"address": "123 Street"

},

headers=headers

)

print(response.json())

**Пример ответа (POST)**:

{

"status": "success",

"student": {

"student\_id": 3002,

"username": "newstudent",

"first\_name": "New",

"last\_name": "Student",

"email": "newstudent@example.com",

"class\_id": 1

}

}

**Коды ошибок**:

* 400: Неверный формат данных.
* 404: Ученик не найден.
* 401: Недействительный API-ключ.

**2. Управление учителями**

**Эндпоинты**:

* GET /api/admin/teachers/<string:search>: Получение списка учителей с поиском.
* GET /api/admin/teacher/<int:teacher\_id>: Получение данных одного учителя.
* POST /api/admin/teacher: Создание нового учителя.  
  **Описание**:
* GET /teachers/<search>: Возвращает список учителей, отфильтрованный по поисковому запросу.
* GET /teacher/<teacher\_id>: Возвращает данные учителя по ID.
* POST: Создает нового учителя.  
  **Параметры**:
* search (path): Поисковый запрос (например, "М" или "Teacher").
* teacher\_id (path): ID учителя.
* JSON для POST:
  + username (string): Логин.
  + password (string): Пароль.
  + first\_name (string): Имя.
  + last\_name (string): Фамилия.
  + email (string): Email.
  + phone\_number (string): Телефон.
  + position\_id (integer): ID должности (например, 1 для завуча).
  + subject\_ids (array): Список ID предметов.
  + class\_id (integer): ID класса (если учитель — классный руководитель).  
    **Заголовки**: X-API-Key: <api\_key>  
    **Пример запроса (POST)**:

response = requests.post(

f"{BASE\_URL}/api/admin/teacher",

json={

"username": "newteacher",

"password": "password123",

"first\_name": "New",

"last\_name": "Teacher",

"email": "newteacher@example.com",

"phone\_number": "+1234567890",

"position\_id": 1,

"subject\_ids": [1, 2],

"class\_id": 1

},

headers=headers

)

print(response.json())

**Пример ответа (POST)**:

{

"status": "success",

"teacher": {

"teacher\_id": 1003,

"username": "newteacher",

"first\_name": "New",

"last\_name": "Teacher",

"email": "newteacher@example.com",

"position\_id": 1,

"subjects": [1, 2],

"class\_id": 1

}

}

**Коды ошибок**:

* 400: Неверный формат данных.
* 404: Учитель не найден.
* 401: Недействительный API-ключ.

**3. Управление расписанием**

**Эндпоинты**:

* GET /api/admin/schedules: Получение списка расписания с фильтрами.
* GET /api/admin/schedule/<int:schedule\_id>: Получение данных одного расписания.
* POST /api/admin/schedule: Создание нового расписания.  
  **Описание**:
* GET /schedules: Возвращает список расписания с возможностью фильтрации по учителю, классу, предмету, дню или времени.
* GET /schedule/<schedule\_id>: Возвращает данные расписания по ID.
* POST: Создает новую запись в расписании.  
  **Параметры**:
* Для GET /schedules (query):
  + teacher (string): Фильтр по имени учителя.
  + class (string): Фильтр по названию класса.
  + subject (string): Фильтр по названию предмета.
  + day (string): Фильтр по дню недели (например, "Понедельник").
  + time (string): Фильтр по времени (например, "09:00").
* schedule\_id (path): ID расписания.
* JSON для POST:
  + class\_id (integer): ID класса.
  + subject\_id (integer): ID предмета.
  + teacher\_id (integer): ID учителя.
  + day\_of\_week (string): День недели (например, "Понедельник").
  + start\_time (string): Время начала (например, "09:00").
  + end\_time (string): Время окончания (например, "10:00").  
    **Заголовки**: X-API-Key: <api\_key>  
    **Пример запроса (POST)**:

response = requests.post(

f"{BASE\_URL}/api/admin/schedule",

json={

"class\_id": 1,

"subject\_id": 1,

"teacher\_id": 1002,

"day\_of\_week": "Понедельник",

"start\_time": "09:00",

"end\_time": "10:00"

},

headers=headers

)

print(response.json())

**Пример ответа (POST)**:

{

"status": "success",

"schedule": {

"schedule\_id": 5,

"class\_id": 1,

"subject\_id": 1,

"teacher\_id": 1002,

"day\_of\_week": "Понедельник",

"start\_time": "09:00",

"end\_time": "10:00"

}

}

**Коды ошибок**:

* 400: Неверный формат данных (например, некорректное время).
* 404: Расписание не найдено.
* 401: Недействительный API-ключ.

**4. Управление предметами**

**Эндпоинты**:

* GET /api/admin/subjects: Получение списка всех предметов.
* GET /api/admin/subject/<int:subject\_id>: Получение данных одного предмета.
* POST /api/admin/subject: Создание нового предмета.  
  **Описание**:
* GET /subjects: Возвращает список всех предметов.
* GET /subject/<subject\_id>: Возвращает данные предмета по ID.
* POST: Создает новый предмет.  
  **Параметры**:
* subject\_id (path): ID предмета.
* JSON для POST:
  + subject\_name (string): Название предмета.  
    **Заголовки**: X-API-Key: <api\_key>  
    **Пример запроса (POST)**:

response = requests.post(

f"{BASE\_URL}/api/admin/subject",

json={"subject\_name": "Информатика"},

headers=headers

)

print(response.json())

**Пример ответа (POST)**:

{

"status": "success",

"subject": {

"subject\_id": 3,

"subject\_name": "Информатика"

}

}

**Коды ошибок**:

* 400: Неверный формат данных.
* 404: Предмет не найден.
* 401: Недействительный API-ключ.

**5. Экспорт данных в Excel**

**Эндпоинты**:

* GET /api/admin/students/excel/<string:search\_query>: Экспорт списка учеников.
* GET /api/admin/teachers/excel/<string:search\_query>: Экспорт списка учителей.
* GET /api/admin/schedules/excel: Экспорт расписания.
* GET /api/admin/subjects/excel: Экспорт списка предметов.  
  **Описание**: Возвращает файл Excel с соответствующими данными.  
  **Параметры**:
* search\_query (path): Поисковый запрос для фильтрации (опционально).
* Для /schedules/excel (query):
  + class (string): Фильтр по названию класса.  
    **Заголовки**: X-API-Key: <api\_key>  
    **Пример запроса**:

response = requests.get(f"{BASE\_URL}/api/admin/students/excel", headers=headers)

with open("students.xlsx", "wb") as f:

f.write(response.content)

**Коды ошибок**:

* 401: Недействительный API-ключ.

**Тестирование API**

Тестовые скрипты (tests\_student.py, test\_teachers.py, tests\_admin.py) покрывают большинство эндпоинтов и сценариев:

* Проверка валидных запросов с корректными API-ключами.
* Проверка ошибок при использовании недействительных ключей.
* Проверка поведения при отсутствии ключа.
* Проверка некорректных данных (например, несуществующие ID).
* Сохранение ответов в JSON или Excel.

Для запуска тестов:

1. Убедитесь, что сервер запущен (python run.py).
2. Запустите тесты:

tests\_student.py

test\_teachers.py

tests\_admin.py

1. Результаты сохраняются в папках student\_test\_results, teacher\_test\_results, test\_results.

**Рекомендации по использованию API**

1. **Безопасность**:
   * Храните API-ключи в безопасном месте и не передавайте их третьим лицам.
   * Используйте HTTPS в продакшене для защиты данных.
2. **Ограничения**:
   * Убедитесь, что запросы не превышают допустимую частоту (если есть лимиты на сервере).
   * Проверяйте корректность входных данных перед отправкой запросов.
3. **Логирование**:
   * API логирует запросы и ошибки в файл api\_access.log. Используйте его для отладки.
4. **Расширение функционала**:
   * Добавьте эндпоинты для редактирования и удаления записей (например, PUT и DELETE для домашних заданий).
   * Реализуйте пагинацию для эндпоинтов с большими списками (например, /api/admin/students).
   * Добавьте фильтры для эндпоинтов, таких как /api/student/homework (например, по предмету или дате).