Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ
КАФЕДРА СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

Анализ данных в веб-сервисе когнитивного тренажёра

Студент ИУ5-32М _____ **А.Е. Чиваризин**
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель НИР _____ **А.М. Балашов**
(Подпись, дата) (И.О.Фамилия)

Консультант _____ **А.И. Канев**
(Подпись, дата) (И.О.Фамилия)

20 24 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой ___ИУ5____
(Индекс)

_____ _____
(И.О.Фамилия)

« _____ » _____ 20 _24_ г.

ЗАДАНИЕ
на выполнение научно-исследовательской работы

по теме _анализ данных в веб-сервисе когнитивного тренажёра_____

_____

_____

Студент группы ____ИУ5-32М____

_____Чиварзин_Александр_Евгеньевич_____
(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

___Учебная_____

Источник тематики (кафедра, предприятие, НИР) _____

График выполнения НИР: 25% к _10 нед., 50% к _13 нед., 75% к 16 нед., 100% к _17 нед.

Техническое задание *_разработать_автоматизированную_информационную_систему__*
_«Веб-сервис когнитивного тренажёра» и проанализтровать данные_____

_____

_____

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на _____ листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

_____

_____

_____

Дата выдачи задания « _10 » __февраля___ 2024 г.

Руководитель НИР _____ _____А.И._Канев_____
(Подпись, дата) (И.О.Фамилия)

Студент _____ ___**А.Е._Чиварзин**___
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Оглавление

ВВЕДЕНИЕ

В настоящее время, для активного участия в жизни общества очень важно уметь говорить и слышать других. Но не у всех людей имеется хорошо развитый слух и, как следствие, речевой аппарат, что сильно затрудняет общение слабослышащих между собой и другими людьми. Устная речь часто применяется в образовании для передачи знаний от преподавателей к студентам, что требует, как правило, хорошего слуха.

Проблемы коммуникации в настоящее время решают в том числе с помощью сурдопереводчиков, но их применение не улучшает слух и речь слабослышащих и являются дорогой услугой. Другим альтернативным методом является обучение людей, желающих пообщаться со слабослышащими – выучить язык жестов, который редко используется при общении обычных людей.

Всё, перечисленное выше, обосновывает необходимость и актуальность слухового тренажёра, который позволит тренировать слух и речь слабослышащих студентов, поскольку до устройства на работу желательно максимально натренировать слух и речь. Кроме того, тренажёр должен быть доступным и удобным в использовании слабослышащими, поскольку в современной обстановке часто применяются дистанционные технологии.

При использовании технологий дистанционных технологий перспективным вариантом будет использование веб-приложения, доступного через браузер по определённому URL-адресу. Такие приложения как правило обращаются к веб-сервисам за данными, которые обычно хранятся в базах данных.

Разработке такого веб-сервиса и базы данных и посвящена данная выпускная квалификационная работа.

В этом НИР будет рассмотрена обработка данных, полученных от ЭУ.

3. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ

3.1. Автоматическая проверка ответов

Веб-сервис позволяет создавать задания, ответы на которые проверяются автоматически. Алгоритм автоматической проверки различается в зависимости от вида задания.

- При чтении учитывается только факт отправки ответа;
- При аудировании — сравнение ответа с правильным ответом на основе расстояния Левен штейна.

Преимущество автоматической проверки перед ручной состоит в том, что она экономит время преподавателя.

Ранее была разработана миварная экспертная система (МЭС) для автоматической проверки ответов и рекомендации вида задания, но она требует доработки, т.к. не выполняет полностью свою задачу. МЭС разрабатывалась в ПО КЭСМИ «Наука», скриншот которой представлен на рисунке 1.

Автоматическая проверка заданий на аудирование проводится следующим образом: вначале сравнивается ответ на задание с правильным ответом, если он совпадает 1:1, с правильным, то выставляется максимальная оценка, если ответ студента полностью не совпадает, то минимальная. Для определения оценки при отсутствии полного совпадения используется расстояние Левенштейна.

Расстояние Левенштейна — это минимальное число односимвольных преобразований (удаления, вставки или замены), необходимых, чтобы превратить одну последовательность в другую. Для двух одинаковых последовательностей символов оно равно нулю. Например, расстояние Левенштейна между словами «Гора» и «Горка» равно единице.

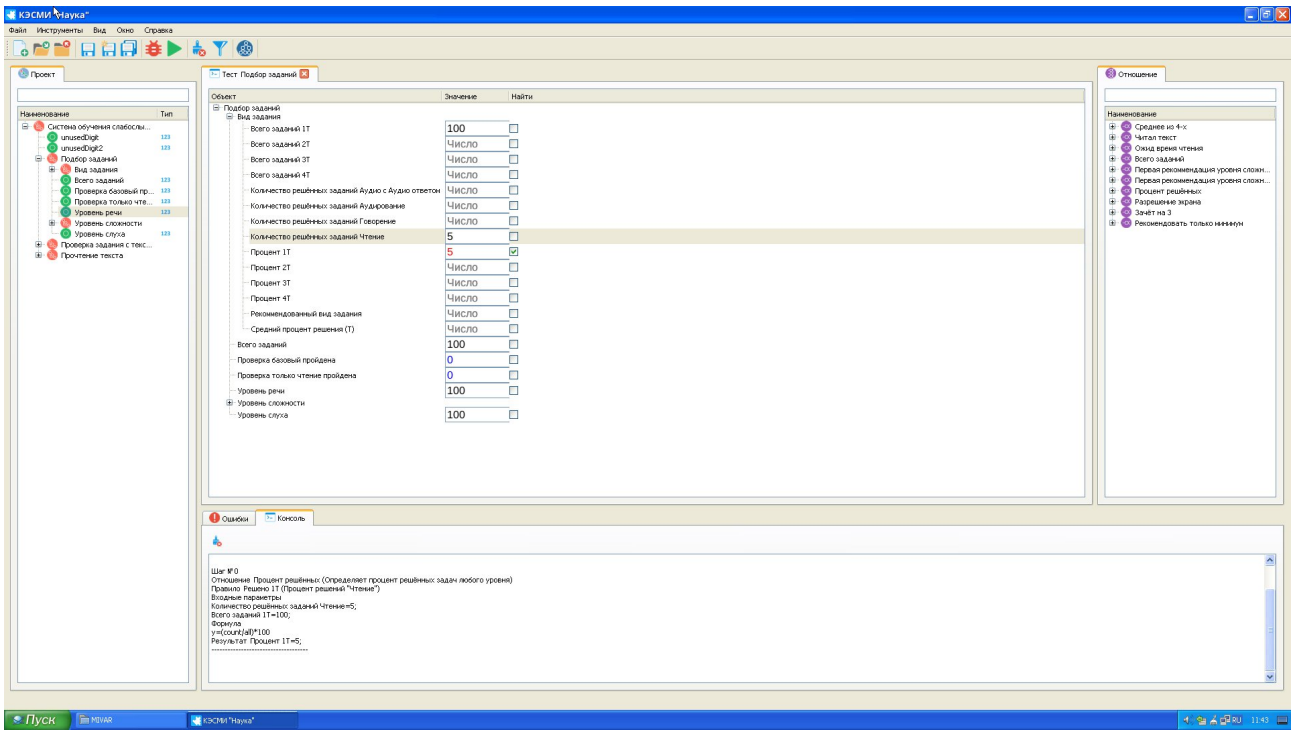

Рисунок 1 — тестирование МЭС в ПО КЭСМИ «Наука».

Для того, чтобы улучшить качество автоматической проверки заданий на аудирование, производится очистка ответа студента и правильного ответа от лишних символов.

Ниже приведён код автоматической проверки заданий на аудирование.

```
case 'Аудирование':
    ответ_правильный = задание.task_answer.lower().replace('\n', '
').replace('  ', ' ')
    ответ_студента = self.answer_text.lower().replace('\n', '
').replace('  ', ' ')
    if ответ_правильный == ответ_студента:
        self.automark_id = Grade.objects.get(grade_number=5).id  #
Отлично
    else:
        len_l = lev(ответ_правильный, ответ_студента)
        max_l = len(ответ_правильный) / 2
        if len_l <= 2:
            self.automark_id =
Grade.objects.get(grade_number=5).id  # Отлично
        elif len_l <= 4 and len_l < max_l:
            self.automark_id =
Grade.objects.get(grade_number=4).id  # Хорошо
        elif len_l <= 6 and len_l < max_l:
            self.automark_id =
Grade.objects.get(grade_number=3).id  # Удовлетворительно
        else:
```

```
            self.automark_id =
Grade.objects.get(grade_number=2).id  # Неудовлетворительно
```

Однако, для проверки заданий на говорение понадобится нейросеть, которая не реализована в веб-сервисе. Кроме реализации нейросети также возможно доработать веб-сервис и подключить стороннюю (в том числе облачную).

Автоматическая проверка заданий на чтение не реализована, однако заготовка под неё имеется в миварной ЭС. Её реализовать практически невозможно, поскольку программное обеспечение не способно контролировать прочтение текста человеком, однако можно сделать «таймер» и выставлять оценку в зависимости от него. У автоматической проверки по таймеру есть недостаток, который заключается в том, что один человек может читать с одной скоростью, а другой — с другой. Ещё важно обратить внимание на то, что человек при экране разного размера и/или разном масштабировании скорость чтения может отличаться.

По результатам автоматической проверки задания любого вида, поддерживающего такую проверку, оценка выставляется как automark. Окончательное рещение об оценке принимает преподаватель.

3.2. Работа с электронным университетом МГТУ им. Н.Э. Баумана

У студентов и сотрудников МГТУ им. Баумана есть электронная почта на домене МГТУ. Теоретически, по электронной почте можно отправлять студентам результаты проверки ответов, но этот функционал ещё не реализован. E-mail сотрудника можно получить по API ЭУ. Про то, как получить e-mail студента по API мне ещё неизвестно.

Кроме того, в веб-сервисе имеются дисциплины и учебные группы, которые также можно получить как объекты из API ЭУ.

Для интеграции с электронным университетом и SSO МГТУ им. Баумана в бэкэнд была внедрена модификация django_eu_auth, за работу с ЭУ в которой отвечает пакет django_eu_auth.eu.

Рассмотрим процесс авторизации пользователя.

После того, как веб-сервис получит GET запрос /api/auth/login с токеном django_eu_auth передаёт токен по API SSO и в ответ получает JSON с информацией о пользователе. Этот JSON выглядит примерно так:


```
{
   "id": 43768,
   "guid": "6c5d2b64-7add-11e8-9b28-005056962143",
   "name": "Чиварзин Александр Евгеньевич",
   "displayName": "Чиварзин Александр Евгеньевич",
   "firstname": "Александр",
   "lastname": "Чиварзин",
   "middlename": "Евгеньевич",
   "birthdate": "1998-08-01",
   "email": "chivarzin-aleksandr-evgenevich@bmstu.ru",
   "username": "chivarzin-aleksandr-evgenevich"
}
```


В библиотеке django_eu_auth есть проверка на существование ключа alias. Если он имеется, то из него извлекается учебная группа студента. Это позволяет использовать авторизацию без доступа к API ЭУ. Как можно заметить, JSON содержит неправильный e-mail, т.к. мои email это chivarzinae@student.bmstu.ru и chivarzin@bmstu.ru. У этого JSON есть ещё один недостаток: если два аккаунта на одно и тоже ФИО, то происходит конфликт по username из-за того, что оно должно быть уникальным.

Для достижения уникальности username была реализована возможность его подмены. Таким образом, в таблицу БД auth_user может быть записано не то username, которое получено от SSO.

Библиотека django_eu_auth поддерживает записать в БД username в следующих форматах:

- USERNAME — оставить username без изменений, т.е. как в JSON от SSO.
- USERNAME-GUID — дописать GUID в конец username и вставить между username и GUID символ «-».
- GUID — заменить username на GUID.

В django_eu_auth есть поддержка аккаунтов VK, Yandex, но там защиты от конфликта по username с имеющимися в БД пользователями.

Таким образом, библиотека обеспечивает уникальность username во втором и третьем случае, но при условии использовании только аккаунтов МГТУ.

После добавления SSO пользователя в БД, веб-сервис записывает данные в таблицу surdoapi_userattrs. Эта таблица была создана в связи с тем, что мне по проекту было дано указание не изменять таблицу пользователей.

Затем веб-сервис запрашивает информацию из контингента ЭУ, чтобы получить группу, в которой обучается этот студент. Если пользователя с этим guid (uuid) нет в контингенте, то идёт запрос в «Сотрудники», если переменной среды STAFF_EMAIL_FROM_EU присвоено значение «True». Адрес электронной почты из ЭУ сохраняется в поле email таблицы auth_user.

API ЭУ, которое было использовано в библиотеке возвращает ответы в формате XML. Библиотека преобразовывает XML в типизированные объекты. В ходе преобразования XML конвертируется в dict при помощи xmltodict, затем создаются объекты на основании содержимого этого dict.

Ниже представлен фрагмент XML (часть заменена на ...).

```
<person guid_kdr="b9e30883-5ae6-11eb-b819-005056b15581" lastname="Чиварзин" firstname="Александр" middlename="Евгеньевич" birthdate="01.08.1998" gender="gender.male" file_num="0000232270" email="chivarzin@bmstu.ru" phone="79854788198" totalseniority_date="06.12.2024" totalseniority_years="1" totalseniority_months="4" pedagseniority_date="06.12.2024" ...>
```

Объект сотрудника создаётся на основе приведённого ниже класса.

```
class StaffPerson:
    guid_kdr: str = ""
    lastname: str = "UNKNOWN"
    firstname: str = "UNKNOWN"
    middlename: str = "UNKNOWN"
    birthdate: str = '01.01.1970'  # Не Python формат даты
    gender: str = 'gender.not_set_:)'
    file_num: int = -1
    email: str = ""
    phone: str = '00000000000'
    totalseniority_date: str = '01.01.1970'
    totalseniority_years: int = -1
    totalseniority_months: int = -1
    pedagseniority_date: str = '01.01.1970'
    part_time_date: str = ""
    part_time_organization: str = 'ООО "Неизвестная Компания" :)'
    part_time_position: str = 'Неизвестный разработчик'
    employment: Employment | None = None
```

Этот объект содержит переменную employment, которая либо None либо объект класса Employment.

Для студента в библиотеке используется другой класс ContingentPerson. Однако, он не содержит e-mail студента в связи с тем, что в XML ответе API ЭУ я не нашёл его.

У студента в зависимости от того, обучался ли он в перый или непервый раз в МГТУ им. Баумана, может быть один или несколько stage в ответе API ЭУ при запросе информации о нём по UUID. Актуальная учебная группа в библиотеке определяется по первому в списке Stage.

Ниже приведён фрагмент XML (частично заменённый на ...)

```
<person uuid="6c5d2b64-7add-11e8-9b28-005056962143" fio="Чиварзин Александр Евгеньевич" lastname="Чиварзин"
```

```
firstname="Александр" middlename="Евгеньевич" birthdate="1998-08-01">
          <citizenship id="a115db60-d635-11dc-bb0c-003048351d16" shortname="Россия" name="Гражданин РФ" alias="citizenship.russian"/>
          <country code="643" shortname="Россия"/>
          <dormitory id="685e5538-d499-11dc-a344-003048351d16" name="Не проживает" shortname="не проживает" alias="dormitory.false" start="" end=""/>
          <gender id="d0475c5a-d596-11dc-ae45-003048351d16" name="Мужской" alias="gender.male"/>
          <stages>
            <stage id="836d88c8-36b0-11ee-bcaa-005056aa673b" ... name="Магистратура" alias="stage.magister" card_number="23УМ282" start="2023-09-01" end="2025-08-31" target="0" network="false" permission="1">
              ...
              <group ... multiplicity="first" studyform="fulltime" course="2" abbr="ИУ5-32М" index="2" semester="3" term_in_year="1" diplom="1" students_budget="14" students_pay="3" ... planned_end_date="31.08.2025" ...>
                ...
              </group>
              ...
            </stage>
            <stage id="c42735e6-7ade-11e8-ad73-005056962143" ... name="Бакалавриат" alias="stage.bachelor" card_number="18Ц026" start="2018-09-01" end="2023-08-31" target="0" network="false" permission="1">
              ...
              <group ... multiplicity="first" studyform="fulltime" course="5" abbr="ИУ5Ц-102Б" ...>
                ...
              </group>
              ...
```

```
      </stage>
    </stages>
  </person>
```

Как видно из XML, информация о студенте включает не только его ФИО и учебную группу. Однако при сохранении данных в БД большинство данных отбрасывается, т.к. в БД хранится намного меньше данных, чем отдаёт API ЭУ.

В веб-сервисе имеется функционал по импорту учебных групп, который реализован с помощью Python скрипта. Этот скрипт вначале запрашивает список факультетов через django_eu_auth.eu, затем через туже библиотеку получает список групп по каждому факультету. Блок-схема этого скрипта показана на рисунке 1.

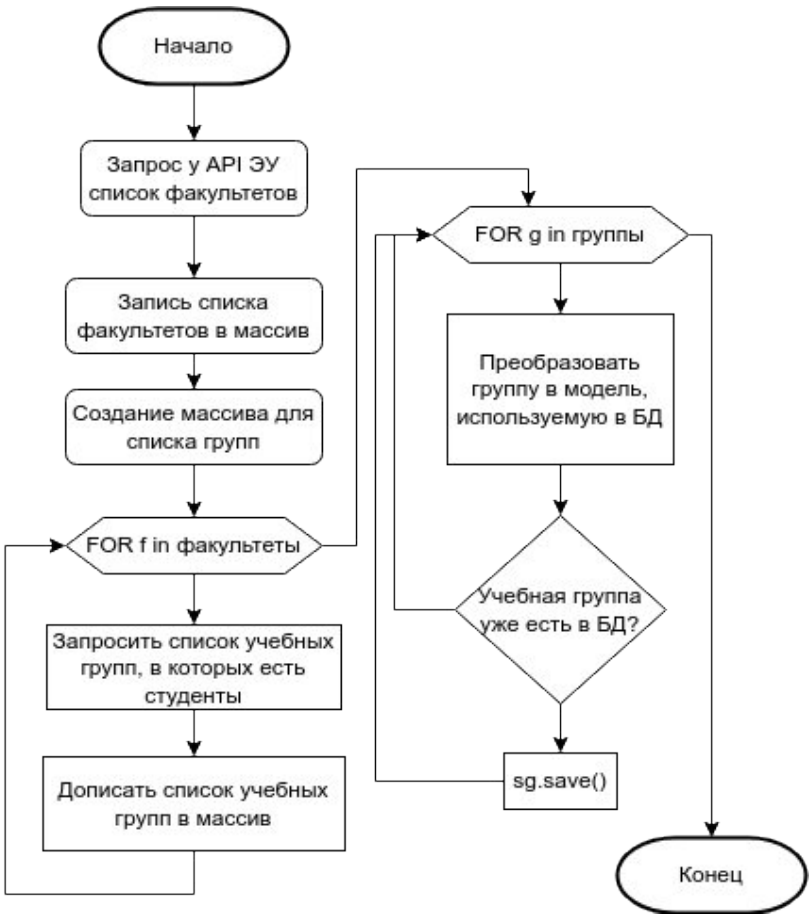

Рисунок 1. Алгоритм сохранения в БД списка учебных групп.

Список факультетов в API ЭУ представлен следующим XML:

```
<?xml version="1.0" encoding="utf-8"?>
<list>
  <item id="a31b5b3d-82fc-1029-96dd-000347adedc6" value="АК"/>
```

```
      <item              id="a30af6b7-82fc-1029-96dd-000347adedc6"
value="ГУИМЦ"/>
      <item id="a22d4cbb-82fc-1029-96dd-000347adedc6" value="ИУ"/>
    </list>
```

В XML выше представлены не все факультеты.

Рассмотрим на примере этого XML конвертацию его в объекты.

После преобразования XML в dict получается примерно такой словарь.

```
{'list': {'item': [
{'@id':   'a31b5b3d-82fc-1029-96dd-000347adedc6',   '@value':
'АК'},
{'@id':   'a30af6b7-82fc-1029-96dd-000347adedc6',   '@value':
'ГУИМЦ'},
{'@id':   'a22d4cbb-82fc-1029-96dd-000347adedc6',   '@value':
'ИУ'}
]
}}
```

В этом dict содержится вложенный словарь «list», в котором один ключ «item», значение которого представляет собой массив.

Как видно, все ключи в словарях, входящих в массив, начинаются с символа «@», однако это связано с тем, что все они были получены из атрибутов XML. Если ключ получен не из атрибута XML, то он не начинается с символа «@».

Элементы этого массива преобразовываются в объекты класса Faculty, который содержит 2 переменных: id и name, где name соответствует value в вышеприведённом dict.

```
self.id = data["@id"]
self.name = data["@value"]
```

У класса Faculty реализован метод groups, который возвращает массив объектов класса StudentGroup. Этот метод возвращает массив объектов класса StudentGroup.

Диаграмма взаимодействия с API электронного университета показана на рисунке 2. Запись в БД не показана на этой диаграмме.

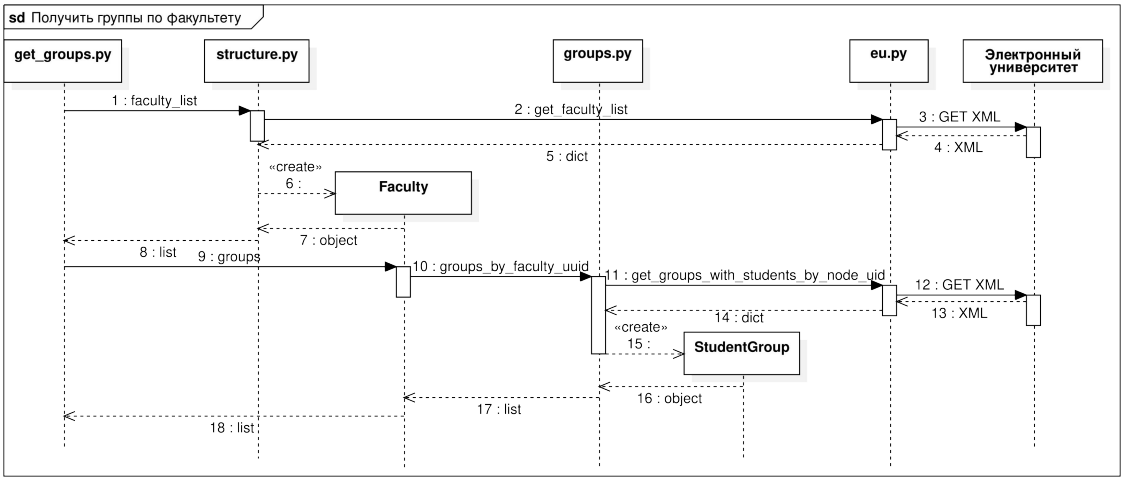

Рисунок 2. Диаграмма взаимодействия с API электронного университета.

Импорт учебных групп позволяет работать с ними до того, как студенты, обучающиеся в соответствующих группах пройдут авторизацию через CAS по API, предоставляемом WEB-сервисом.

3.3. Интеграция django_eu_auth в проект на основе Django

Для того, чтобы интегрировать Django_eu_auth в проект необходимо скопировать каталог django_eu_auth в каталог с проектом Django.

Для работоспособности библиотеки требуются следующие зависимости: Django, Django REST Framework, xmltodict, django-environ.

Для интеграции с основным приложением в django_eu_auth используются переменные среды, которые приведены в таблице 1.

Таблица 1. Переменные среды.

Переменная	Описание	Значение по-умолчанию
SSO_HTTPS_REDIRECT	Если True, то редиректпользователь получитредирект по HTTPS.	False
SSO_HTTPS_DOMAIN	Домен, на которыйперенаправлять приуспешной авторизации	

Переменная	Описание	Значение по-умолчанию
SSO_ENABLE_HEADER_AUTH	Разрешить авторизацию по HTTP заголовку	False
SSO_KEYWORD	Keyword для токена	Bearer
SSO_PREFIX		
REDMINE_ENABLED	Включить интеграцию с Redmine	False
REDMINE_API_KEY	API ключ Redmine	
SSO_BMSTU_USERNAME_FORMAT	Как записывается имя пользователя в БД	USERNAME
SSO_DEBUG	Выводить отладочную информацию в лог если включена отладка Django	False
EU_CONTINGENT_PERSONS_URL	Начало URL запроса к определённому сервису ЭУ.	
EU_STAFF_PERSONS_URL		
EU_FACULTY_LIST_URL		
EU_STRUCTURE_GROUP_LIST_URL		
EU_UCHPLAN_URL		

В ходе разработки WEB-сервиса (включая django_eu_auth) переменные среды были задокументированы в MD файлах.

В переменных среды также включены URL-ы сервисов ЭУ. Это было сделано по заданию. Вынесение этих URL-ов в переменные среды предотвращает случайную отправку запросов к ЭУ на хостах, не имеющих права на отправку соответствующих запросов.

Для отладки библиотеки предусмотрена переменная среды SSO_DEBUG, однако, она не влияет на отладочный вывод части кода, из django_eu_auth.eu. Каждая строка вывода отладочного лога SSO начинается с «[SSO][DEBUG]».

Веб-сервис взаимодействует с библиотекой через методы, которые описаны в файлах README.md.

ЗАКЛЮЧЕНИЕ

При разработке веб-сервиса когнитивного тренажёра были получены следующие результаты:

1. Были обработаны и проанализированы данные, полученные из API электронного университета.
2. Была разработана библиотека, предназначенная для работы с API электронного университета.
3. Был разработан механизм автоматической проверки заданий.
4. Библиотека работы с ЭУ была интегрирована в django_eu_auth.
5. Django_eu_auth, созданная Балашовым Антоном Михайловичем, была улучшена.

Список использованной литературы

1. Ответы API Электронного университета. Хост backend.guimc-dev.bmstu.ru