ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет информатики, математики и компьютерных наук

Программа подготовки бакалавров по направлению

38.03.05 Бизнес-информатика

Пилипенко Глеб Игоревич

Курсовая работа

по теме “Продуктизация и доработка навыка голосового помощника Алиса для работы с расписанием и новостями ВШЭ”

Выполнил:

Пилипенко Глеб Игоревич 22БИ-3

Принял: Дыдычкин Александр Алексеевич

**Нижний Новгород, 2024**

**Оглавление**

[Пояснительная записка 3](#_heading=h.gjdgxs)

[Техническое задание 7](#_heading=h.1fob9te)

[Анализ проблематики 10](#_heading=h.3znysh7)

[Анализ существующих решений, технологий и аналогов 16](#_heading=h.2et92p0)

[Полученные результаты, методика испытаний и итоги анализа 23](#_heading=h.tyjcwt)

[Список использованных источников 31](#_heading=h.3dy6vkm)

**Пояснительная записка**

**Введение**

Наименование: Написание модуля по внесению информации о студентах, преподавателях Высшей школы экономики, их расписании и работа с этими данными в Yandex Data Base, проектирования подпрограммы по взаимодействию с пользователем во время сессии навыка..

Разработка ведется на основании данный документов:

1. Официальная документация языка программирования Python.
2. Официальная документация по YDB.
3. Официальная документация YDB Python SDK.

Современные голосовые помощники становятся неотъемлемой частью повседневной жизни благодаря своей простоте и функциональности. Среди множества виртуальных ассистентов, представленных на российском рынке, особой популярностью пользуется «Алиса» от компании Яндекс. Cогласно [данным](https://sprut.ai/news/chto-proishodit-na-rynke-golosovyh-assistentov-v-rossii)[1] доля, занимаемая данным продуктом на рынке выросла с 44% в октябре 2020 года до 83% в августе 2024.

Российский рынок умных колонок в 2024 году показал значительный рост. По данным группы «М.Видео-Эльдорадо», опубликованным 17 января 2025 года в TAdviser, объем продаж таких устройств увеличился на 20% в штучном выражении и на 26% в денежном по сравнению с предыдущим годом.

Цель данной работы – оптимизировать взаимодействие пользователя с голосовым помощником при получении информации о расписании, сделав этот процесс максимально быстрым и комфортным. Также необходимо пересмотреть выполненное ранее решение, оптимизировать его, сделать более универсальным и доступным для пользователей не только филиала НИУ ВШЭ в Нижнем Новгороде, но и в остальных кампусах университета.

Проблематика данной курсовой работы заключается в использовании баз данных компании Яндекс для удовлетворения потребности в универсальности и добавления возможности пользователю вносить данные расписания самостоятельно. Разрабатываемая подпрограмма в рамках командного проекта выполняется функцию по внесению, сбору, изменению и хранению информации о пользователях. Также выполнена работа по составлению прототипа модуля взаимодействия с пользователем.

**Назначение и область применения**

Цель реализуемого программного модуля заключается во внесении информации о пользователе, его расписании, поиску внесенного расписания в базе данных и приведении его в читаемый для пользователя вид.

Данная система предназначена преимущественно для студентов и преподавателей Высшей школы экономики, независимо от их территориального расположения (кампуса). Основной способ взаимодействия с системой осуществляется через голосового помощника Алиса, однако возможна интеграция и с другими платформами, где требуется функционал управления учебным расписанием и работы с базой данных.

**Технические характеристики**

Главной задачей являлось анализ и изменение кода, который был выполнен в прошлой работе, а именно: изменение логики модуля взаимодействия с пользователем, добавление преподавательского режима, возможности вносить и получать пары майноров, английского и факультативов, расширение области применимости, которая бы включала все кампусы НИУ ВШЭ, написание подпрограммы, которая будет по полученным данным из модуля взаимодействия с пользователем осуществлять внесение информации о нем в базу данных, а также о предметах, которые необходимо добавить и осуществлять поиск по базе данных выбранных предметов и на выходе формируется перечень объектов, содержащих полную информацию о учебных занятиях: временные параметры (дата и время), название курса, место проведения (аудитория), ответственного преподавателя, а также вид учебной работы (лекционный формат, практическое занятие или исследовательский семинар). К основным ограничением данного способа хранения и заполнения расписания можно отнести: сложность в составлении и унификации списка предметов, т.е. заранее необходимо знать список дисциплин всех образовательных программ НИУ ВШЭ, для их обработки, необходимость постоянно добавлять информацию о дисциплинах вручную, что может быть не удобно пользователю.

Методы, которые применялись для разработки навыка включали использование официальных библиотек YDB и самого сервиса для создания заранее спроектированной базы данных, заполнения необходимой информации о пользователе: фамилия, имя, отчество и номер группы, если это студент, информации о группе, расписании. Используемый способ подразумевает в независимости от кампуса, вида предмета хранить расписание, что делает его более гибким, чем решение с парсингом гугл таблиц, а также в виду того, что все предметы хранятся в заранее установленным виде, универсальность данного решения также выше. Подпрограмма состоит из пяти модулей:

1. Модуль models, который содержит классы данных: базовый класс Lesson для занятий, содержащий атрибуты name, type\_l, building, auditorium, id\_lecturer, time, is\_weekly, is\_upper, lesson\_date. Два класса наследника PersonalLesson**,** GroupLesson, которые наследуют Lesson, добавляя соответственно атрибуты  id\_student или id\_group. Также присутствуют вспомогательные классы для таблиц: PersonalLessonStudent, GroupLessonGroup. Помимо этого в данном модуле содержатся классы Student, Lecturer для пользователей, содержащие данные о ФИО и номер группы, если это студент, объект группа, который хранит атрибуты name, edu\_year, edu\_program, faculty, edu\_format, edu\_level и вспомогательный класс LecturerGroup.
2. Модуль obj\_queries, содержащий функции для каждого из объектов: Group, Student, Lecturer по получению записей в базе данных объекта, идентификатора и проверке регистрации.
3. Модуль Registration\_ydb, хранящий в себе общую функцию регистрации registration\_user и методы, которые используются ей в зависимости от параметров для регистрации студента, лектора или группы.
4. Модуль schedule\_queries, осуществляющий внесение расписания в таблицы: insert\_lesson, insert\_lesson\_data insert\_help\_tables\_data, его поиск и изменение данных пользователя в БД. В силу особенностей хранения данных и выполнения запросов функции для поиска предмета по названию, лектору, дате разделены для студентов и преподавателей и также присутствуют отдельные методы для нахождения предмета по дню недели: find\_by\_week\_day\_lesson\_student, find\_by\_week\_day\_lesson\_lecturer.
5. Модуль utils содержит функцию make\_readable для вывода данных пользователю в заданном формате.

Рассматриваемый программный модуль функционирует как связующее звено между базой данных YDB и компонентом обработки голосовых запросов ассистента. В процессе работы система выполняет преобразование структурированных данных из хранилища в массив объектов типа row, принадлежащих классу ydb.ResultSet.

Входные данные содержат объект ydb.QuerySessionPool, который необходим для каждой функции по взаимодействию с базой данных и варьируются в зависимости от задачи. Так методы поиска требуют указания параметров, по которым будет осуществляться поиск, связан ли предмет с данной образовательной программой, идентификаторов студента, группы. В функции добавления предмета требуются данные пользователя в виде списка. Также некоторые другие функции требуют передачи объектов Lesson, Student или Lecturer в зависимости от задачи.

Написанная подпрограмма выполнена на языке программирования Python. Можно выделить следующие основные моменты, которые повлияли на данное решение:

* Использование остальными участниками данного языка программирования, поэтому для интеграции рассматриваемого решения необходимо применение единственного решения.
* Высокая популярность рассматриваемого языка программирования.
* YDB обладает обширной документацией и множеством готовых библиотек, что значительно упрощает работу с этим инструментом. Кроме того, выбранный язык программирования обеспечивает высокую скорость разработки благодаря интуитивному синтаксису, активной поддержке комьюнити и постоянному развитию экосистемы.
* Важно и то, что этот язык широко применяется для обработки и анализа данных, включая взаимодействие с YDB, а также с другими источниками информации, что делает его универсальным решением для подобных задач.

Разработанный программный модуль размещается в облачной инфраструктуре Yandex Cloud. Эта платформа предлагает готовые решения для хранения и выполнения кода, специально адаптированные для работы с Яндекс Диалогами. Особенно стоит отметить возможность бесплатного тестирования голосовых навыков и их размещение через сервис Yandex Cloud Functions. Помимо этого, применяемые технологии включают в себя использование сервиса YDB – распределенная реляционная система управления базами данных с открытым исходным кодом на основе языка SQL. Данное информационное решение является ключевым в разрабатываемой программе, потому что оно поддерживают удобную и понятную интеграцию с сервисом Yandex Cloud Functions, в котором размещен сам навык и выполняет функцию по хранению и обработке информации о пользователях и их расписании. Также одним из преимуществ данного продукта является возможность бесплатного использования в рамках ограниченных лимитов.

Применение указанных инструментов существенно оптимизирует работу, сокращая время на тестирование, разработку и внедрение модуля по хранению, извлечению и работе с данными студентов и преподавателей и их расписания.

**Техническое задание**

Модуль по работе с данными пользователей - программа, которая позволяет осуществлять заполнение, хранение, поиск и дальнейшую обработку информации, полученной от студентов и преподавателей в процессе взаимодействия с голосовым ассистентом.

Цель в рамках описанного продукта является создание программного модуля, который будет осуществлять вышеописанные действия с информацией о пользователях всех кампусов НИУ ВШЭ, обеспечивая быстрый и точный доступ к информации через голосовые и текстовые запросы.

Разрабатываемая подпрограмма представляет собой ключевой компонент системы, реализованной в виде навыка для голосового помощника Алиса. Данное программное решение призвано усовершенствовать взаимодействие пользователей с системой, предоставив новый пользовательских опыт.

Основное преимущество реализации заключается в создании интуитивно понятного интерфейса голосового управления, который обеспечивает быстрый и комфортный доступ к запрашиваемой информации. Такой подход существенно сокращает временные затраты пользователей по сравнению с традиционными методами поиска данных, одновременно повышая точность и релевантность получаемых результатов.

**Функциональные требования:**

1. Разрабатываемый программный модуль должен взаимодействовать с продуктом YDB для работы с информацией о студентах и преподавателях различных программ и форматов обучения Высшей школы экономики, их расписании.
2. Программный модуль должен вносить информацию о пользователях в БД, а именно фамилию, имя, отчество и номер группы, если пользователь студент. Также должна фиксироваться информация о группе: название, курс, образовательная программа, факультет, формат обучения, уровень образования. Помимо того, фиксируются данные о предмете, которые включают название, тип предмета (семинар или лекция), корпус, аудитория, преподаватель, время, дата и группа, если предмет проводится для конкретной образовательной программы, или студент, если данный предмет является индивидуальным для студента, например, майнор, факультатив.
3. Программный модуль должен извлекать необходимые данные о расписании пользователей по атрибутам: название, лектор, дата, которая включает, как конкретное число и месяц, так и определенный день недели или пары на сегодня и завтра.
4. Собранные сведения требуется записать в стандартизированном формате, обеспечивающем их дальнейшее использование.
5. При необходимости пользователь должен иметь возможность изменить хранящуюся в базе данных информацию.
6. Разрабатываемый программный модуль должен осуществлять регистрацию преподавателей или групп, если они указываются в атрибутах предмета, но не содержатся в базе данных.

**Требования к надежности:**

1. Программный компонент обязан предусматривать корректную реакцию на ситуации разрыва соединения с СУБД, включая механизмы повторного подключения и обработки исключительных состояний
2. Компонент работы с расписанием должен демонстрировать эффективное использование ресурсов при получении данных минимальное время отклика и достаточную производительность для соответствия требованиям по времени ответа голосового ассистента.

**Требования к информационной и программной совместимости:**

1. Язык программирования: Python (минимальная поддерживаемая версия — 3.12). Независимость от платформы: функционирование должно быть обеспечено вне зависимости от операционной системы или аппаратной конфигурации устройства, на котором работает голосовой помощник.
2. Модуль должен обеспечивать корректную работу с библиотекой по взаимодействию с СУБД YDB.
3. Итоговое программное решение должно быть совместимо с другими модулями, предусмотренными в проекте, а именно получать на вход ссылку и возвращать на выходе данные для модуля поиска необходимых элементов расписания.
4. Программа должна быть реализована с использованием принципов ООП.

**Требования к маркировке и упаковке**

Финальная версия приложения подлежит публикации в официальном каталоге навыков голосового помощника Яндекса. Что касается модуля обработки и хранению данных в YDB, специальные требования к его маркировке и способу упаковки не устанавливаются.

**Требования к транспортированию и хранению.**

Подпрограмма по работе с информацией и пользователях и их расписании может располагаться на виртуальной машине, поддерживающей использование и развертывание Webhook сервера и взаимодействие с YDB, бессерверной платформе Yandex Cloud Funtions или в другом месте, которое соответствуют требованиям выше как часть общего навыка голосового ассистента до момента внесения изменений разработчиками или работниками платформы. К транспортировке никакие требования не предъявляются.

**Тестирование**

1. Нужно проверить разработанную программу при различных входных значениях, включающих ситуации, когда пользователь является не только студентом, но и преподавателем, отсутствует регистрация группы или преподавателя при вводе информации о предмете, вводятся различные данные предмета.
2. Необходимо сделать проверку выполнения CRUD-операций, т.е. создание новых записей, их получение и передача в нужном формате, обновление строк таблицы и их удаление.
3. Протестировать корректную передачу данных подмодулю по обработке запросов пользователя и скорость выполнения запросов, чтобы она не превышала заданного времени для ответа навыка.

**Стадии и этапы разработки**

1. Формирование технического задания:

-Анализ и систематизация функциональных характеристик подпрограммы

-Определение нефункциональных параметров реализации

1. Исследование исходных данных:

-Анализ необходимой структуры данных для хранения

-Проектирование базы данных, включающее создание таблиц и связей между ними

1. Создание архитектуры:

-Разработка структурной схемы модуля

-Декомпозиция на ключевые компоненты и интерфейсы взаимодействия

1. Реализация базового функционала:

-Программная интеграция с сервисом YDB.

-Создание алгоритмов внесения, извлечения и трансформации информации.

1. Верификация работоспособности:

-Тестирование корректности обработки данных

-Проверка устойчивости к вариациям входных форматов

1. Отладка и оптимизация:

-Устранение выявленных ошибок

-Повышение надежности работы модуля

1. Обеспечение совместимости:

-Адаптация интерфейсов для интеграции с другими компонентами навыка

-Тестирование кроссплатформенной работы

1. Финальная интеграция:

-Встраивание модуля в общую систему

-Комплексное тестирование в составе проекта

-Корректировка взаимодействующих компонентов

1. Документирование:

-Подготовка технического описания архитектуры

-Составление руководства по использованию, справочных функций

**Анализ проблематики**

Анализ проблематики предполагает комплексное исследование ключевых аспектов разработки программного решения. В первую очередь необходимо детально изучить доступные источники информации, уделяя особое внимание их функциональным возможностям, форматам хранения и структурным особенностям организации данных о расписании. Параллельно следует проанализировать потенциальные трудности, которые могут возникнуть в процессе разработки и последующей интеграции модуля в общий проект.

Важным направлением исследования становится сравнительный анализ преимуществ и ограничений выбранного способа организации и получения данных с точки зрения его применимости в контексте создаваемого навыка. Особое внимание необходимо уделить изучению условий и технических ограничений, в рамках которых ведется разработка программного модуля.

Значительная часть анализа будет посвящена изучению существующих подходов к решению аналогичных задач, включая обзор реализованных навыков обработки расписания. При этом особый интерес представляют решения с открытым исходным кодом, позволяющие проанализировать применяемые архитектурные и алгоритмические подходы.

Во время поиска потенциальных мест, откуда возможно извлечение информации о расписании студентов и анализа источников были найдены и проработаны некоторые варианты получения необходимой информации. Также были рассмотрены источники, упомянутые в предыдущей работе. Наиболее перспективными и возможными к внедрению стали варианты самостоятельного фиксирования расписания или получения его с сайта РУЗ. К остальным источникам получения расписания можно отнести разработанный модуль по парсингу Google таблиц, расположенных на сайте расписания студентов Нижегородского кампуса НИУ ВШЭ, и извлечение данных из Единого личного кабинета студента.

В рамках прошлой курсовой работы был разработан модуль по парсингу Google Таблиц и получения данных из этого источника информации. В силу отсутствия других вариантов данный способ представлялся как практически единственно возможный, но обладающий рядом недостатков. Рассматриваемый подход обладает рядом существенных технических ограничений, требующих особого внимания. Прежде всего, система не предусматривает автоматизированного сбора данных о занятиях дополнительного профиля, включая программы Minor и курсы английского языка, поскольку эти дисциплины не входят в базовое расписание академических групп.

Существенные сложности связаны с особенностями работы с Google Таблицами как основным источником данных. Любые изменения в структуре или формате исходных таблиц требуют существенной модификации алгоритмов обработки информации. Это обусловлено жесткой зависимостью системы от конкретного способа организации и представления данных в используемом источнике.

Отдельную проблему представляет механизм авторизации через Google API. Поскольку модуль использует персональные учетные данные разработчика, периодически возникает необходимость их обновления в связи с истечением срока действия токенов доступа. Это приводит к временной потере функциональности до момента обновления аутентификационных данных.

Несмотря на все эти ограничения, в прошлой курсовой работе выбор пал на данный способ, так как он все равно позволял получать в среднем данные о расписании, которые соответствуют действительности, но в рамках данной курсовой работы по причине изменения организации хранения расписания и нахождении потенциально более универсального способа хранения и получения информации, было принято решение отказаться от использования ранее описанного подхода и полностью переписать навык.

Что касается парсинга данных с Единого личного кабинета студента, то ограничения, которые были связаны с необходимостью предоставлять конфиденциальные данные в виде пароля и логина от корпоративной почты НИУ ВШЭ остались неразрешенными, поэтому данный способ возможен только в рамках личного применения и локального запуска на устройстве пользователя.

В качестве альтернативного варианта рассматривался способ получения данных через подключение к системе РУЗ. Данный подход был признан достаточно универсальным и эффективным для разработки модуля сбора информации о расписании. В отличие от общих источников, таких как Google Таблицы, РУЗ предоставляет персонифицированные данные, включая индивидуальные занятия (например, майноры, курсы английского языка), которые отсутствуют в групповых расписаниях. Кроме того, информация в этой системе регулярно обновляется, что гарантирует её актуальность.

Применение данного метода также открывало возможность реализации функционала для преподавателей, поскольку система содержит все необходимые для этого сведения. Дополнительным преимуществом являлось наличие готовых решений в данной области, что могло сократить время разработки и минимизировать возможные ошибки за счёт анализа существующего опыта.

Однако ключевым ограничением оказался закрытый доступ к РУЗ: подключение возможно исключительно через внутренние сети НИУ ВШЭ или университетский VPN, причём только для сотрудников вуза. Однако, в рамках майнора один из участников данной курсовой работы получил доступ к файлу конфигурации VPN, что сделало возможным бы использование данного способа, но по неизвестным причинам доступ к сайту РУЗа при использовании данного файла прекратился, а с помощью заявок по получению нового файла результата достичь не удалось.

Таким образом, также, как и в прошлой работе по причине недоступности большинства способов получения расписания или ограничений, связанных с их использованием было принято решение об использовании нового подхода, а именно – сервиса YDB. Данный способ позволяет фиксировать расписание, включая английский язык, майноры и схожие предметы пользователям с различных кампусов Высшей школы экономики. Единственным ограничением является необходимость пользователю самому вносить данные о занятиях, что может быть выполнено не всегда удобно, особенно, в случае голосового управления навыком.

YDB (Yandex Database) — это распределённая отказоустойчивая СУБД, разработанная компанией Yandex для работы с большими объёмами данных и высоконагруженными сервисами. Система поддерживает горизонтальное масштабирование, что позволяет эффективно обрабатывать запросы даже при значительном росте нагрузки. YDB обеспечивает совместимость с SQL-запросами, что упрощает интеграцию с существующими решениями, а также предлагает инструменты для работы как с структурированными, так и с полуструктурированными данными (например, JSON).

Одним из ключевых преимуществ YDB является её встроенная отказоустойчивость: данные автоматически реплицируются между узлами, что минимизирует риск потери информации при сбоях. Кроме того, сервис предоставляет низкие задержки при выполнении запросов, что критически важно для голосовых помощников, где скорость ответа напрямую влияет на пользовательский опыт.

Сервис также предоставляет возможность бесплатного использования в рамках обозначенных ограничений и интегрирован в платформу Yandex Cloud, что позволяет ускорить разработку и тестирования навыка голосового ассистента. Также YDB поддерживает бессерверные вычисления в рамках работы с Yandex Cloud Functions, что избавляет от необходимости аренды, развертывания и поддержки виртуальной машины и значительно облегчает процесс выполнения работы.

Для интеграции YDB с Python используется официальный YDB Python SDK, предоставляемый Yandex Cloud. Этот SDK поддерживает все основные операции с базой данных, включая выполнение SQL-запросов, транзакционную обработку данных и работу с prepared statements для оптимизации повторяющихся запросов.

Ключевые возможности SDK:

-Подключение к YDB через gRPC-протокол, обеспечивающий высокую скорость передачи данных.

-Асинхронная работа через библиотеку asyncio, что критически важно для навыков голосовых помощников, где задержки должны быть минимальными.

-Поддержка сессионного управления, позволяющая контролировать время жизни соединения и избегать перегрузки сервера.

Данный способ хранения информации позволяет самостоятельно выбирать формат данных и сделать хранение информации более универсальным. В отличие от Google Таблиц, где данные жестко привязаны к формату ячеек, YDB позволяет хранить информацию в структурированном виде (например, в формате JSON или реляционных таблицах с четкой схемой). Это исключает проблемы, связанные с изменением оформления данных, так как система автоматически адаптируется под заданную структуру.

В Google Таблицах отсутствует возможность автоматически связывать разрозненные данные (например, название предмета с временем и аудиторией), если они не имеют строгой привязки по расположению. YDB, благодаря поддержке связей между таблицами и индексации, позволяет мгновенно находить сопутствующую информацию, даже если исходные данные меняют формат.

Также одним из главнейших преимуществ является возможность обработки расписания пользователей различных кампусов университета и корректно работать с дополнительными курсами: Minor, английский язык, различные факультативы и др. В предыдущий раз алгоритм был заложником формата и не позволял выполнять данные действия.

Хранение информации организовано в рамках реляционной базы данных[Приложение1], которая включает в себя следующие таблицы:  
1. Студент – содержит ФИО студентов, их id, id группы для связки с соответствующей таблицей.

2. Группа включает в себя id группы, название, курс, образовательную программу, факультет, формат обучения: очный, заочный или иной, уровень образования.

3. Преподаватель - ФИО, id.

4. Пара(групповая) содержит атрибуты: id, название, тип предмета(семинар, лекция), корпус, аудитория, id преподавателя, время, поля которые отражают каждую неделю проходит пара или нет и по верхним неделям или нет проходят занятия, дата и id группы. Данная таблица служит для хранения занятий, которые проводятся для групп конкретной образовательной программы.

5. Пара(индивидуальная) хранит все атрибуты групповой пары с единственным отличием, вместо id группы, хранится id студента. Соответственно, рассматриваемая таблица необходима для фиксации дисциплин, которые могут быть индивидуальны для каждого студента: английский язык, майонор, факультатив и другие.

6. Пара(групповая)-группа – вспомогательная таблица, чтобы обеспечить нормализацию данных. Необходима для связи между таблицами Группа и Пара(групповая), соответственно.

7. Преподаватель-группа – вспомогательная таблица.

8. Студент-пара(индивидуальная) – вспомогательная таблица.

Подводя итог, можно сказать, что YDB является оптимальным решением для осуществления организации хранения, обработки и получения информации о пользователях и их расписании и обладает рядом преимуществ по сравнению с рассмотренными выше способами: универсальностью, гибкостью, быстрой и эффективной возможностью выполнять поиск против одного недостатка - необходимости самостоятельно заполнять данные о расписании.

В рамках исследуемой проблематики было проанализировано несколько существующих решений, однако ни одно из них не может быть использовано в качестве полноценной альтернативы.

Первое рассмотренное решение требует исключительно ввода кода, полученного от бота, не предоставляя пользователю никаких дополнительных сведений или инструкций. Второй вариант – "Помощник Вышки" – оказался неработоспособным: при обращении к нему система выдает сообщение об ошибке, указывающее на невозможность получения ответа. Хотя исходный код данного навыка доступен на GitHub, его функциональность ограничена из-за отсутствия доступа к API Расписания учебных занятий (РУЗ), что приводит к ошибкам при попытке запуска. Таким образом, на текущий момент в открытых источниках не обнаружено готового решения, соответствующего поставленным в проекте задачам. Выполненный в прошлом году программный модуль на данный момент нефункционален в силу изменения способа хранения информации и не обладает рядом существенных недостатков по сравнению с новым выбранным вариантом реализации.

Что касается опыта других учебных заведений, то было выявлено два подхода. Финансовый университет разработал специализированное приложение, интегрированное с внутренним API расписания. Аналогичное решение существует в Томском политехническом университете, где расписание доступно через отдельный веб-сайт. Прочие аналогичные системы, функционирующие в данной предметной области, используют более совершенные и универсальные методы получения данных, что обеспечивает их стабильную работу.

Данный анализ подтверждает актуальность разработки нового решения, поскольку существующие аналоги либо неработоспособны, либо адаптированы под специфические условия других образовательных учреждений.

**Анализ существующих решений, технологий и аналогов**

Для реализации необходимого функционала подпрограммы существует несколько вариантов реализации хранения и работы с базой данных. Во-первых, существуют вариации расположения базы данных, создание собственного сервера, использование уже готовых облачных решений, которые представляют также различные возможности, т.е. некоторые решения предполагают аренду и развертывание облачный серверов или использование бессерверных вычислений. Помимо этого, в качестве базы данных для хранения информации также присутствует большое число вариантов.

Одним из основных решений является использование Yandex Database (YDB) – распределённой отказоустойчивой СУБД, которая обеспечивает высокую производительность и масштабируемость, что критически важно для обработки запросов от множества пользователей. Альтернативными вариантами могут выступать реляционные (PostgreSQL, MySQL) и документоориентированные (MongoDB) базы данных, каждая из которых обладает своими преимуществами и ограничениями в контексте задачи.

Для решения задачи взаимодействия с базой данных можно использовать различные языки программирования, так как все перечисленные СУБД поддерживают многие популярные варианты, также существует большое число библиотек, что увеличивает количество реализаций выбранной задачи в разы.

Выбор оптимальной технологии хранения данных является ключевым аспектом при разработке системы обработки расписания для голосового ассистента. В современных условиях существует выбор между различными классами систем управления базами данных, каждый из которых обладает уникальными характеристиками и особенностями применения.

Традиционные реляционные СУБД, такие как PostgreSQL и MySQL, долгое время остаются стандартом для работы со структурированными данными. Их основное преимущество заключается в строгой организации информации и поддержке принципов ACID, что гарантирует целостность данных при выполнении транзакций. Возможность выполнения сложных запросов с использованием операций JOIN и развитое сообщество пользователей делают эти системы популярными среди разработчиков. Однако применительно к нашему проекту они демонстрируют ряд ограничений, включая сложности горизонтального масштабирования, потенциальное снижение производительности под высокой нагрузкой и жесткую схему данных, затрудняющую модификацию структуры хранимой информации.

В качестве альтернативы рассматриваются NoSQL-системы, такие как MongoDB и Redis, которые предлагают принципиально иной подход к хранению данных. Их основные преимущества включают гибкую схему данных, высокую производительность операций чтения и записи, а также эффективную работу с большими объемами неструктурированной информации. Упрощенное горизонтальное масштабирование делает эти системы привлекательными для проектов с переменной нагрузкой. Однако для нашей задачи они имеют существенные ограничения, связанные с отсутствием полноценной поддержки операций JOIN, ограниченными возможностями аналитической обработки и неполной реализацией транзакционных механизмов.

Особое место среди современных решений занимает Yandex Database (YDB), которая сочетает преимущества разных подходов к хранению данных. Эта система поддерживает как традиционные SQL-запросы, так и NoSQL-подход, обеспечивая при этом горизонтальную масштабируемость без потери производительности. Встроенные механизмы отказоустойчивости и автоматического восстановления делают YDB надежным решением для критически важных приложений. Для нашего проекта особую ценность представляет глубокая интеграция YDB с экосистемой Яндекса, включая голосового ассистента Алису и облачную платформу Yandex Cloud. Возможности автоматического масштабирования под нагрузку и высокая доступность данных благодаря распределенной архитектуре позволяют создать устойчивую и производительную систему. Кроме того, YDB эффективно работает как со структурированными данными (например, расписанием занятий), так и с полуструктурированной информацией.

Проведенный анализ технологических решений показывает, что YDB наиболее полно соответствует требованиям проекта, объединяя надежность реляционных систем с гибкостью NoSQL-подхода. Дополнительным преимуществом является естественная интеграция с платформой голосового ассистента, что существенно упрощает разработку и развертывание конечного решения. Уникальные характеристики YDB в сочетании с возможностями экосистемы Яндекса делают эту платформу оптимальным выбором для реализации системы работы с расписанием студентов.

Для эффективной интеграции Yandex Database в систему обработки расписания необходимо рассмотреть доступные инструменты разработки. YDB предоставляет несколько вариантов взаимодействия, позволяя выбрать оптимальный подход в зависимости от требований проекта и навыков разработчиков.

Основным языком программирования для работы с YDB является Go, для которого существует официальный SDK с полной поддержкой всех функций базы данных. Go особенно хорошо подходит для создания высоконагруженных распределенных систем благодаря своей производительности и встроенным возможностям конкурентного программирования. Альтернативой выступает Python, для которого также имеется официальная библиотека ydb-python-sdk. Python предпочтителен при необходимости быстрой разработки прототипов или интеграции с системами анализа данных.

Для веб-интерфейсов и серверных приложений может использоваться Node.js с соответствующим JavaScript SDK. Этот вариант особенно актуален при создании единой JavaScript-инфраструктуры для всего проекта. В enterprise-решениях часто применяют Java через JDBC-драйвер YDB, что обеспечивает совместимость с существующими корпоративными системами.

Особого внимания заслуживает возможность работы через HTTP/gRPC API, что позволяет интегрировать YDB с любыми языками программирования, включая C++, PHP или Ruby. Для .NET-разработчиков доступен официальный .NET SDK, поддерживающий все основные функции платформы.

При выборе языка программирования для разработки программного модуля по извлечению расписания были учтены несколько ключевых факторов. Во-первых, Python применяется при разработке других частей навыка, что упрощает интеграцию отдельных подпрограмм в единый проект. Во-вторых, поскольку задача не предполагает обработки значительных объемов данных, высокая производительность C++, Gо и других языков не является критичным требованием. Наконец, решающую роль сыграл имеющийся опыт работы с Python, что позволило ускорить процесс реализации решения.

В качестве библиотек для взаимодействия с YDB было принято решение использовать официальную библиотеку ydb python sdk, которая позволяет использовать и самостоятельно писать запросы к базе данных. Альтернативой было применение библиотеки ydb-sqlalchemy, но использовании данной библиотеки теряется производительность, что могло повлиять на конечный результат, где скорость ответа навыка критически важна, поэтому выбор был остановлен на первом техническом решении.

Для работы программы с использованием вышеописанных технических средств необходимо небольшая настройка платформы Yandex Cloud, в которой будет происходить запуск функций по работе с YDB.

1. В консоли управления Yandex Cloud выбрать сервис YDB.
2. Создать новую базу данных, указав название и тип serverless. Это необходимо для применения с Yandex Cloud Functions, поддерживающих вычисления без необходимости развертывания облачного сервера.
3. После развертывания базы данных нужно сохранить: Endpoint (адрес подключения, например, grpcs://ydb.server.com:2135), путь к БД (например, /ru-central1/b1g8ejq2vl02h2k3d4r5/etn0123456789)
4. Для доступа к YDB из кода необходимо в разделе IAM создать сервисный аккаунт с необходимыми правами для возможности взаимодействия с базой данных.
5. В разделе Yandex Cloud Functions создать файл requirements.txt, где указать зависимость ydb, в качестве сервисного аккаунта выбрать только что созданный и прописать в переменных окружения YDB\_ENDPOINT, YDB\_DATABASE, созданные и сохраненные на предыдущем этапе.

Проведенные исследования и анализ доступных технологических решений убедительно доказали целесообразность использования YDB в качестве основного хранилища данных для системы управления учебным расписанием. Разработанная архитектура интеграции с голосовым ассистентом демонстрирует высокую степень надежности и производительности, полностью соответствующую всем техническим требованиям проекта. Выбор языка программирования Python обусловлен его очевидными преимуществами - простотой использования, высокой скоростью разработки и отсутствием жестких требований к вычислительной мощности в рамках данного проекта. Особенностью предложенного решения стало применение бессерверной архитектуры на базе Yandex Cloud Functions, что позволило достичь оптимального баланса между ключевыми характеристиками системы: производительностью при обработке запросов, возможностью динамического масштабирования под изменяющуюся нагрузку и экономической эффективностью эксплуатации. Такое сочетание технологических решений обеспечивает не только стабильную работу системы в текущий момент, но и создает прочную основу для ее дальнейшего развития и модернизации без необходимости кардинального изменения базовой архитектуры.

**Полученные результаты, методика испытаний и итоги анализа**

Итогом выполненной работы оказалось программный модуль, состоящий из нескольких подмодулей и фиксирующий информацию о пользователе, его расписании, обрабатывающий данную информаций и предоставляющий расписание в виде массива. Для работы данного модуля необходимо выполнение всех шагов, указанных выше, добавление файлов с кодом в Yandex Cloud Functions и файла requirements.txt с зависимостью ydb.

Модуль models.py реализует объектную модель данных, предназначенную для хранения и обработки информации о расписании учебных занятий, группах, студентах и преподавателях в рамках интеграции с Yandex Database (YDB). Основу модуля составляет иерархия классов, обеспечивающая структурированное представление сущностей предметной области.

Центральным элементом архитектуры является базовый класс Lesson, который инкапсулирует общие характеристики всех типов занятий. К ним относятся название дисциплины, тип занятия (лекция, семинар и др.), данные о месте проведения (корпус и аудитория), идентификатор преподавателя, время проведения, а также флаги, определяющие периодичность (еженедельное занятие) и принадлежность к верхней или нижней неделе. Дополнительно класс хранит дату проведения занятия.

Наследники класса Lesson расширяют его функциональность, добавляя специализированные атрибуты. Класс PersonalLesson вводит поле id\_student для идентификации студента, с которым связано индивидуальное занятие, тогда как GroupLesson содержит поле id\_group, указывающее на учебную группу. Оба класса-наследника используют механизм наследования, вызывая конструктор родительского класса через super().\_\_init\_\_().

Для представления учебных групп предназначен класс Group, хранящий название группы, учебный год (в строковом формате) и дополнительные опциональные параметры, такие как образовательная программа, факультет, формат и уровень обучения. Персональные данные участников образовательного процесса описываются классами Student и Lecturer. Первый включает фамилию, имя, отчество студента и идентификатор его группы, второй ограничивается ФИО преподавателя.

Особую роль играют вспомогательные классы, обеспечивающие связи между сущностями. Класс PersonalLessonStudent устанавливает соответствие между студентом и индивидуальным занятием, а GroupLessonGroup связывает групповое занятие с учебной группой. Также в модуле присутствует класс LecturerGroup, который реализовывает связь преподавателя с группой.

Представленная объектная модель обеспечивает целостное представление данных в системе, позволяя эффективно работать с расписанием и связанными сущностями. Использование строгой типизации (приведение типов для идентификаторов) и принципов ООП (наследование, инкапсуляция) способствует надежности и поддерживаемости кода при интеграции с YDB в контексте голосового ассистента Алиса.

Модуль obj\_queries.py реализует функционал взаимодействия с Yandex Database (YDB), предоставляя набор методов для выполнения запросов к базе данных. Основная задача модуля - обеспечение CRUD-операций для сущностей, описанных в модуле models.py, с использованием пула сессий YDB.

Модуль содержит ряд унифицированных функций для работы с различными сущностями системы. Каждая сущность (группа, занятие, студент, преподаватель) обрабатывается по схожему шаблону, что обеспечивает целостность архитектуры.

Для работы с учебными группами реализованы три ключевые функции:

* get\_group\_records выполняет поиск записей о группе в базе данных, принимая параметр is\_exist\_check для оптимизации запроса при проверке существования записи
* select\_id\_group возвращает идентификатор группы, используя сортировку по ID для надежного определения актуальной записи
* is\_group\_reg проверяет наличие группы в базе данных, возвращая булево значение

Аналогичный набор функций (get\_lesson\_records, select\_id\_lesson, is\_lesson\_reg) реализован для работы с занятиями. Функции автоматически определяют тип занятия и формируют соответствующий SQL-запрос, учитывая различные условия поиска для разных типов занятий.

Для работы с персональными данными студентов и преподавателей предусмотрены:

* Функции get\_student\_records, select\_id\_student, is\_student\_reg для работы со студентами
* Функции get\_lecturer\_records, select\_id\_lecturer, is\_lecturer\_reg для работы с преподавателями

Функция connect\_lecturer\_with\_group реализует связь между преподавателем и учебной группой. Сначала она проверяет существование связи, и только при ее отсутствии выполняет операцию вставки, что предотвращает дублирование данных.

Все функции модуля используют пул сессий YDB (ydb.QuerySessionPool), что обеспечивает эффективное управление соединениями с базой данных. Запросы выполняются с автоматическим повторением при временных сбоях благодаря методу execute\_with\_retries.

Для параметризованных запросов используется декларация переменных непосредственно в SQL-коде (через DECLARE), что повышает безопасность и предотвращает SQL-инъекции. Типы данных явно указываются при передаче параметров, включая преобразование дат из строкового формата.

Для обеспечения надежности работы с базой данных и предотвращения потенциальных ошибок в модуле используется аннотация типов Python, явное приведения типов для некоторых категорий полей и преобразование строковых дат в объекты date.

Модуль registration\_ydb.py представляет собой ключевой компонент системы управления образовательными данными, обеспечивающий взаимодействие с Yandex Database (YDB). Его основное назначение заключается в организации процесса регистрации пользователей и учебных групп с последующим сохранением.

Центральным элементом модуля является функция registration\_user, которая обрабатывает весь цикл регистрации пользователей. В зависимости от параметра is\_student функция определяет тип регистрируемого пользователя (студент или преподаватель) и выполняет соответствующий набор операций. Для студентов процесс регистрации включает дополнительные этапы работы с учебными группами - система сначала проверяет существование группы через вспомогательную функцию is\_group\_reg, при необходимости регистрирует новую группу с помощью reg\_group, а затем получает ее идентификатор через select\_id\_group. Только после этого выполняется проверка и регистрация самого студента. В случае преподавателей процесс более простой и ограничивается проверкой и добавлением записи в соответствующую таблицу.

Функция reg\_group реализует логику добавления новых учебных групп в систему. Она формирует параметризованный SQL-запрос, включающий все атрибуты класса Group: название группы, учебный год, образовательную программу, факультет, формат и уровень обучения. Аналогичный подход используется в функции reg\_student, которая дополнительно обрабатывает связь студента с учебной группой через параметр id\_group. Все числовые идентификаторы явно преобразуются в соответствующий тип данных YDB.

Для работы с преподавателями предназначена функция reg\_lecturer, которая сохраняет в базе данных полные анкетные сведения (ФИО) преподавателя. В отличие от студентов, регистрация преподавателей не требует дополнительных операций с сопутствующими сущностями. Все функции модуля используют единый интерфейс работы с базой данных через пул сессий YDB, что обеспечивает эффективное управление соединениями и повышает общую производительность системы.

Важной особенностью модуля является реализация механизма предотвращения дублирования данных. Перед каждой операцией добавления новой записи система проверяет ее существование в базе данных, и в случае обнаружения дубликата возвращает специальный код ошибки (-1). Такой подход позволяет внешним компонентам системы корректно обрабатывать попытки повторной регистрации без генерации исключений. Все запросы к базе данных выполняются через метод execute\_with\_retries, что обеспечивает устойчивость системы к временным сбоям и проблемам с соединением.

Архитектура модуля построена с учетом принципов повторного использования кода и единообразия реализации. Все функции следуют общему шаблону работы с YDB, используют согласованный набор параметров и одинаковые подходы к формированию SQL-запросов. Это значительно упрощает сопровождение и развитие кода, а также делает поведение системы более предсказуемым. Модуль органично интегрируется с другими компонентами системы, используя определенные в models.py классы данных и вспомогательные функции из obj\_queries.py, что создает целостную и согласованную архитектуру работы с образовательными данными.

Модуль schedule\_queries.py представляет собой комплексный компонент для работы с расписанием занятий в системе управления образовательным процессом, интегрированной с Yandex Database (YDB). Он обеспечивает полный цикл операций с расписанием - от поиска информации до добавления и изменения данных, поддерживая как групповые, так и индивидуальные занятия.

Модуль начинается с определения вспомогательных структур данных - словарей week\_days\_dict и module\_boundaries, которые используются для преобразования дней недели и определения границ учебных модулей соответственно. Основная логика модуля реализована в нескольких группах функций, каждая из которых отвечает за конкретный аспект работы с расписанием.

Для студентов реализованы функции find\_lesson\_student и find\_by\_week\_day\_lesson\_student, которые позволяют искать занятия по различным критериям. Первая функция поддерживает поиск по названию предмета, дате или идентификатору преподавателя, автоматически обрабатывая специальные значения "today" и "tomorrow". Вторая функция осуществляет поиск по дням недели, используя преобразование дат через функцию DateTime::GetDayOfWeek YDB.

Для преподавателей предусмотрены аналогичные функции find\_lesson\_lecturer и find\_by\_week\_day\_lesson\_lecturer, которые учитывают специфику работы с преподавательским расписанием. Все функции поиска возвращают список найденных записей или пустой список, если соответствий не обнаружено.

Функция insert\_lesson\_data обеспечивает базовую вставку данных о занятии, автоматически определяя тип занятия (групповое или индивидуальное) и формируя соответствующий SQL-запрос. Она обрабатывает все атрибуты занятия, включая название, тип, место проведения, время и периодичность.

Вспомогательная функция insert\_help\_tables\_data отвечает за заполнение связующих таблиц, которые устанавливают отношения между занятиями и студентами или группами. В зависимости от типа занятия она работает с соответствующей таблицей (GroupLessonGroup или PersonalLessonStudent).

Комплексная функция insert\_lesson объединяет всю логику добавления занятий, включая проверку существования преподавателя в системе, регистрацию при необходимости, и последующее создание занятия с установлением всех необходимых связей. Функция поддерживает различные сценарии работы как для студентов, так и для преподавателей.

Функция change\_db\_data реализует механизм обновления информации о студенте в базе данных с использованием операции UPSERT, что обеспечивает атомарность изменения данных. Она обрабатывает полный набор персональных данных студента и его принадлежность к учебной группе.

**Список использованных источников**

1. Какие голосовые ассистенты популярны в России. Результаты опроса Sprut.ai [Электронный ресурс].: официальный сайт- https://sprut.ai/news/chto-proishodit-na-rynke-golosovyh-assistentov-v-rossii (дата обращения: 10.01.2024)
2. Краткое руководство по Python [Электронный ресурс]. — URL: <https://developers.google.com/sheets/api/quickstart/python?hl=ru> (дата обращения: 13.01.2024)
3. Документация языка Python: [Электронный ресурс]. официальный сайт- <https://www.python.org/doc/> (дата обращения: 15.02.2024)
4. Документация Yandex Cloud: [Электронный ресурс] официальный сайт.т - <https://yandex.cloud/ru/docs/functions/tutorials/alice-skill?utm_referrer=https%3A%2F%2Fyandex.ru%2F> (дата обращения: 16.02.2024)
5. Школа Алисы: официальный сайт[Электронный ресурс]. — URL: https://events.yandex.ru/events/webinars/alisa-school/index (дата обращения: 08.03.2024)