МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра математического моделирования

Допустить к защите
Заведующий кафедрой
акад. РАН, профессор,
д-р физ.-мат. наук.
_____ В.А. Бабешко
27. 06 2022г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

РАЗРАБОТКА МОДЕЛЕЙ И АЛГОРИТМОВ СИСТЕМ СПРАВОЧНИКОВ ПРОФЕССИОНАЛЬНОЙ ДЕЯТЕЛЬНОСТИ

Работу выполнил _____ А.В. Зейлеш

Направление подготовки 02.03.03 – «математическое обеспечение и администрирование информационных систем»

Направленность (профиль) Технология программирования

Научный руководитель
канд. физ.– мат. наук, доц. _____ К.И. Костенко

Нормоконтролер
канд. физ.-мат. наук, доц. _____ С.Е. Рубцов

Краснодар
2022

РЕФЕРАТ

Дипломная работа 44 с., 30 рисунков, 13 источников, 1 приложение.

PYTHON, WEB РАЗРАБОТКА, СПРАВОЧНИК, ФРЕЙМВОРК DJANGO, БАЗЫ ДАННЫХ, ORACLE, HTML.

Объектом исследования дипломной работы является система справочников профессиональной деятельности, в ходе которой необходимо разработать пользовательский интерфейс, ориентированный на обычного пользователя, который не является специалистом, что позволит сократить время и силы обычных работников на внесение изменений в систему справочников без помощи специалистов в области взаимодействия с базами данных.

Взаимодействие с приложением подразумевается по средствам web-ресурсов, что позволит осуществлять простой доступ с любого устройства подключенного к интернету.

Так же были рассмотрены классы задач пользователей, решаемые на основе разработанного приложения. Приведены примеры взаимодействия пользователя с приложением и виды задач, которые пользователь сможет решить, используя приложение. Построены диаграммы пользовательского интерфейса в которых описан полный функционал взаимодействия пользователя с интерфейсов программы и описание принципов разработки данного приложения на языке программирования Python, при использовании фреймворка Django. Рассмотрены используемые функции и библиотеки, а также принцип работы некоторых ключевых элементов программы.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В данной курсовой работе рассмотрен принцип создания web-приложения для работы с системами справочников профессиональной деятельности по средствам такого языка программирования, как Python, с использованием фреймворка Django.

Приложение ориентировано на обычных пользователей, которым необходимо взаимодействовать с данными которые хранятся в справочниках базы данных, без помощи специалистов, но также приложением может воспользоваться и специалист, так как приложение достаточно удобно и просто в использовании. Разработанное приложение позволяет использовать базовые функции работы с информацией в справочниках, доступен функционал редактирования, добавления и поиска данных в каждом из имеющихся справочников, что позволит взаимодействовать только с необходимой информацией, а не просматривать всю информацию, которая имеется в базе данных.

Функционал приложения не ограничен использованием лишь одной базы данных, имеется возможность добавления и подключения новых баз данных по средствам пользовательского интерфейса, представленного в приложении, но для полноценной работы с приложением требуется вмешательство специалиста, который сможет добавить необходимые справочники в программу.

1 Постановка задачи

1.1 Область применения

В ходе работы в крупной компании, в которой есть огромная база данных всегда есть множество форм, которые необходимо заполнять и в этих формах имеются данные, которые не задаются динамически, а выбираются уже готовые варианты, например, это будет выпадающий список на сайте с вариантами выбора вариантов ответа, в котором все варианты ответа хранятся в базе данных, с которой может взаимодействовать только специалист в данной области или человек, деятельность которого направлена на взаимодействие с базой данных, а обычный пользователь не имеет доступа к этой базе данных и для того, что бы добавлять свои варианты в справочники, необходимо обращаться к специалисту, что займет достаточно много времени и нагрузит работой не только обычного пользователя, которому необходимы дополнительные значения в справочниках, но и специалиста, который имеет доступ к базе данных и может редактировать справочники.

На сайте кубанского государственного университета используется та же система работы со справочниками, которая предполагает выбор уже имеющегося варианта, для этого рассмотрим простой пример выбора научного направления в разделе публикаций на сайте infoneeds, данный справочник можно увидеть на рисунке 1.

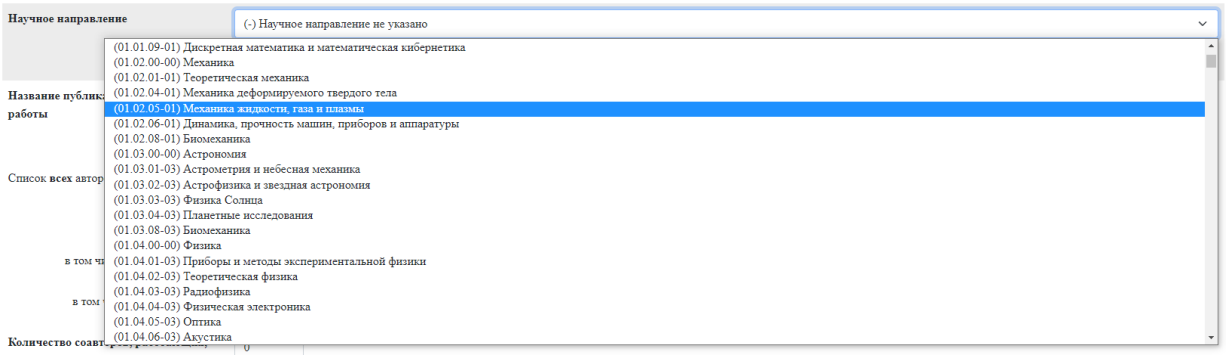


Рисунок 1 – Выбор научного направления

На рисунке 1 изображен справочник выбора научного направления в разделе публикаций. Здесь представлен список научных направлений, с указанием названия научного направления и кодом этого научного направления. Если возникнет необходимость изменить код конкретного научного направления или его названия, придется обращаться к специалисту, который взаимодействует непосредственно с базой данных, в которой хранятся эти данные и запрашивать разрешение на внесение изменений в конкретное направление, которые возможно могут быть связаны с простой ошибкой, когда был введен неверный код направления или допущена опечатка в заполнении направления в справочнике.

1.2 Решение задачи

Для того, чтобы избавиться от потребности обычному пользователю обращаться к специалисту для внесения не критичных поправок в справочники, которые хранятся в базе данных, можно предусмотреть взаимодействие пользователя со справочниками по средствам графического интерфейса, в котором будет представлена возможность просмотра, редактирования, поиска и удаления информации, которая хранится в справочниках.

Самым доступным вариантом разработки такого пользовательского интерфейса для взаимодействия со справочниками, хранящимися в базе

данных будет разработка web-сайта, в котором будет предусмотрена авторизация пользователя и удобный интерфейс для поиска и работы с данными, которые хранятся в справочниках.

1.3 Иерархическая структура

1.3.1 Виды иерархических структур

При помощи иерархических структур можно рассматривать всю систему, как иерархическую структуру, в которой программа разбивается на логические модули или подсистемы, располагающиеся на разных уровнях иерархии. Такой подход часто используется для создания программного обеспечения.

При проектировании иерархии системного программного обеспечения подсистема нижнего уровня предоставляет управление подсистеме верхнего уровня, той, которая вызывает системы нижнего уровня. Нижний уровень имеет более конкретную информацию, например, службы ввода и вывода транзакций. На среднем уровне предоставляются функции предметной области, такие как основные службы обработки. А верхний уровень используется для представления абстрактных функций в виде пользовательского интерфейса.

Основная подпрограмма

Данный стиль был создан для повторного исследования модулей и свободной разработки новых модулей. Суть этого стиля заключается в том, что система делится на подпрограммы с использованием уточнения сверху вниз в соответствии с необходимой функциональностью.

Данные уточнения идут вертикально до тех пор, пока декомпозированные модули не станут достаточно просты, для того, чтобы являться независимыми. Диаграмму описанной иерархии можно увидеть на рисунке 2.

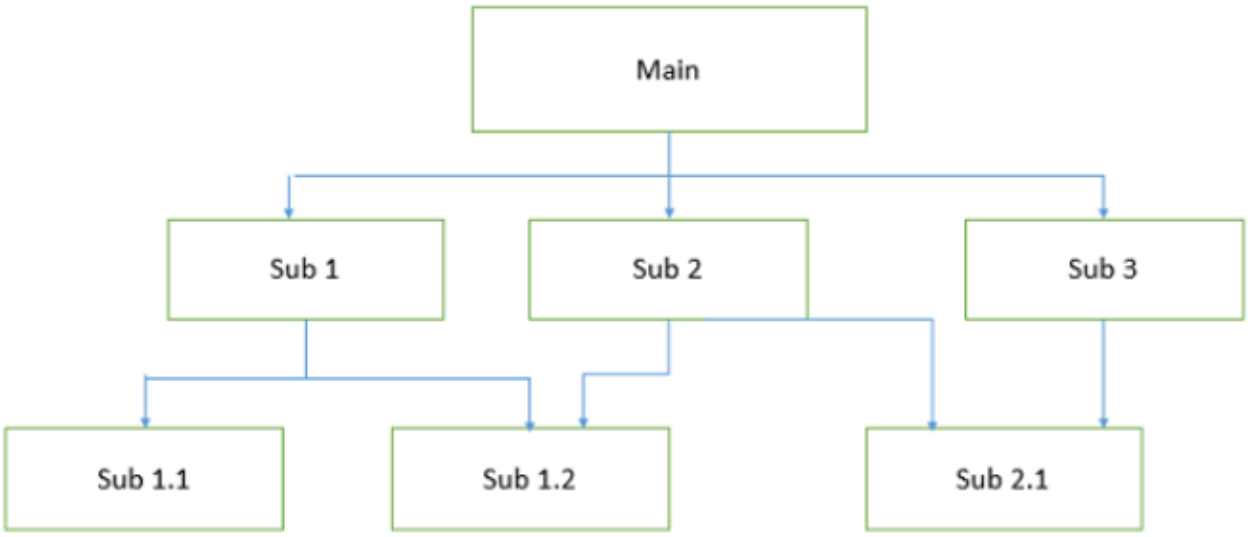


Рисунок 2 – Основная подпрограмма

Плюсами данной иерархии можно считать легкость декомпозирования системы на основе уточнения иерархии и возможность использовать эту систему в объектно-ориентированном программировании.

Но у данной системы есть и недостатки, связанные с уязвимостью, из-за того, что в данной системе содержится много глобально общих данных, так же при создании достаточно тесных связей между модулями может возникнуть потребность в изменении нескольких модулей, после изменения одного.

Мастер

Суть этого подхода заключается в том, что один управляет многими. Такая модификация иерархии обеспечивает надежность и отказоустойчивость.

В этой архитектуре модули, которые являются наследниками, предоставляют головному модулю свой функционал, а головной модуль выбирает нужный результат при помощи определенной системы выбора. Подчиненные модули могут выполнять одну и ту же задачу, но использовать разные алгоритмы и методы решения. В такой системе все ведомые модули

могут работать параллельно и выполнять поставленные задачи. Диаграмма иерархии данной системы представлена на рисунке 3.

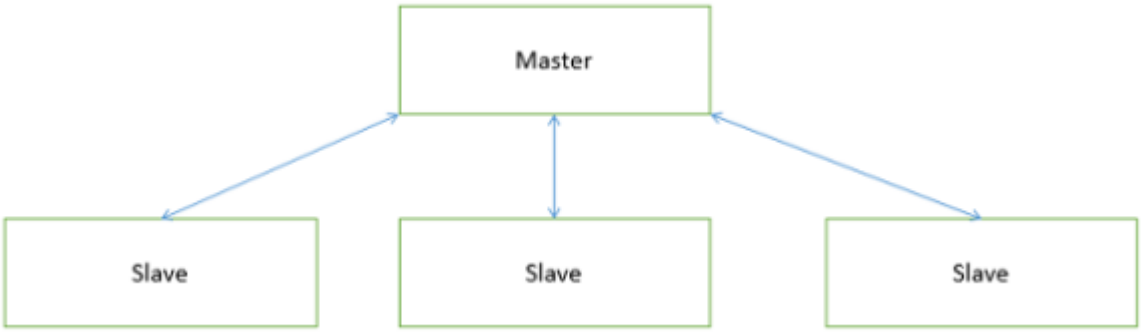


Рисунок 3 – Мастер

Для создания такой системы необходимо разделить выполняемые задачи на набор разных подзадач, указать как можно вычислить конечный результат после получения результатов работы всех модулей, разработать интерфейс для одновременного взаимодействия со всеми модулями, при помощи которого будет происходить делегирование на подзадачи.

Преимуществом данной системы является быстрота вычислений, простота масштабирования, обеспечение надежности из-за возможности дублирования ведомых систем и каждая система может быть реализована по-разному, что позволит свести к минимуму семантические ошибки.

Недостатками данной системы является сложность в создании связи между модулями, невозможность разделить некоторые проблемы на разные модули, сложность реализации и проблема переносимости.

Архитектура виртуальной машины

Архитектура виртуальной машины разрабатывается для использования функциональности, которую уже имеет система, в которую она внедряется. Она строится на базе существующей системы и должна представлять виртуальную абстракцию, набор атрибутов и операций.

В архитектуре этой системы мастер использует тот же функционал, что и ведомые подсистемы и выполняет те же функции, как разделение работы,

вызов других методов и объединение других результатов. Такая система позволяет разработчикам тестировать системы, которые еще не созданы, а также моделировать возможные ошибки, решение которых было бы слишком сложным, дорогостоящим или опасным на реальной системе.

Обычно виртуальная машина разделяет язык программирования или среду приложения с платформой выполнения. Основной целью является возможность переноса. На рисунке 4 изображена модель виртуальной машины.

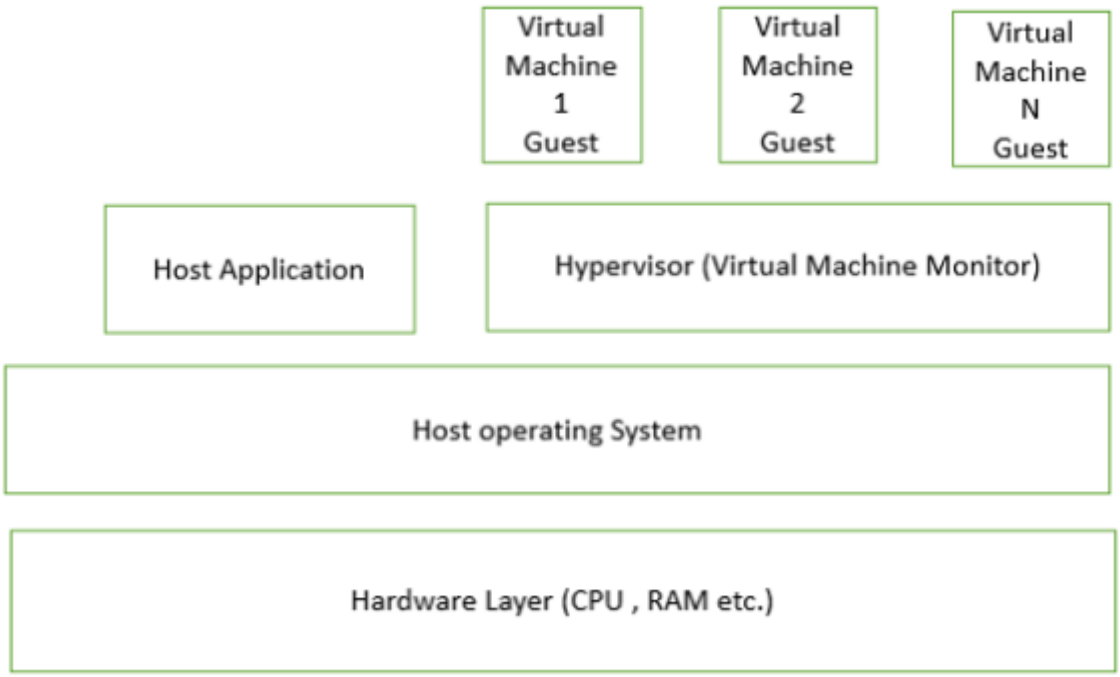


Рисунок 4 – Виртуальная машина

Преимуществами данной системы являются портативность и независимость от машинной платформы, простота разработки программного обеспечения, обеспечение гибкости за счет прерывания и запроса программ, возможность моделирования аварийной ситуации и внесение изменений во время выполнения программы.

Недостатки, это медленное выполнение интерпретатора и большое количество затрат, связанных с дополнительными вычислениями.

Слоистый стиль

При использование данного подхода разрабатывается ряд более глубоких и более поверхностных уровней иерархии, где каждый уровень несет свою исключительную роль в системе.

Каждый уровень состоит из группы связанных классов, инкапсулированных в пакет, в развернутый компонент или в виде группы подпрограмм в формате библиотеки методов или заголовочного файла.

Каждый уровень предоставляет услуги вышестоящему уровню и служит клиентом нижележащего уровня, то есть запрос к уровню i+1 вызывает услуги, предоставляемые уровнем i, через интерфейс уровня i. Ответ может вернуться на уровень i+1, если задача выполнена; в противном случае уровень i постоянно вызывает службы уровня i-1 ниже.

Преимуществами этой системы являются пошаговое проектирование на основе абстракций, обеспечение независимости от улучшений, поскольку изменение функции одного уровня влияет максимум на два других уровня, возможность разделения стандартного интерфейса и его реализации, реализация с использование технологии основанной на компонентах, которые значительно упрощают систему, позволяя подключать новые компоненты, каждый уровень может быть независимо развернутой абстрактной машиной, которая поддерживает переносимость, простота декомпозиции системы на основе определения задач нисходящим уточнением, различные реализации которые могут быть взаимозаменяемы.

К недостаткам данной системы можно отнести то, что многие приложения или системы нелегко структурировать многоуровневым образом, пониженная производительность во время выполнения, потому, что пользовательский запрос или ответ пользователю должен пройти несколько уровней, также существуют проблемы с производительностью, связанные с накладными расходами на маршалинг и буферизацию данных на каждом уровне. Открытие межуровневой связи может вызвать тупиковые ситуации, а большое количество связей может вызвать тесную связь. Наличие

исключений и обработчика ошибок может оказаться проблемой в многоуровневой системе иерархии, поскольку ошибка на одном уровне может распростереться на все уровни выше.[1]

1.3.2 Построение иерархии

Для данной задачи была разработана простая иерархическая схема представления справочников и их информации, в виде двух уровней, где на первом уровне иерархии будет храниться информация о имеющихся справочниках, а на втором информация, которую содержит в себе каждый справочник, графическое представление данной иерархии можно увидеть на рисунке 5.

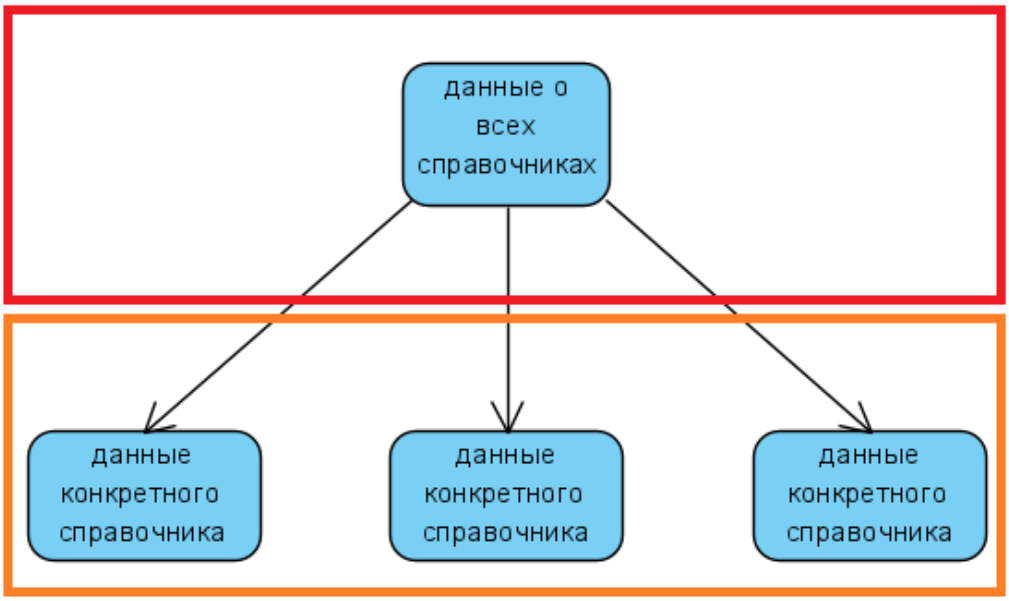


Рисунок 5 – Иерархия представления справочников

На данном рисунке изображено примерное представление того, как будет выглядеть иерархия представления справочников в пользовательском интерфейсе, на первом уровне представлен единый блок с информацией о всех справочниках при помощи которого можно перейти в подробной информации любого из представленных справочников, в примере

информация, которая хранится в конкретном справочнике изображена в виде трех разных классов, что является лишь примером, в реальной ситуации количество справочников будет гораздо больше.[2]

2 Python

2.1 Что такое язык программирования

Для начала определим, что такое язык программирования. Языком программирования можно считать набор формальных правил, используя которые пишут программы. Обычный язык нужен для общения людей между собой, а язык программирования используется для общения человека с компьютером. Поэтому, как и в любом языке, тут есть свои правила лексики, такие, как слова, функции и операторы, при использовании которых по правилам синтаксиса создается выражение. Составленные выражения имеют четкий и определенный смысл, который будет понятен компьютеру.

Так же стоит отличать языки программирования от алгоритмов, поскольку в разных языках программирования один и тот же алгоритм, выполняющий те же действия может выглядеть совсем иначе.

Для того, чтобы запустить любую программу для начала нужно перевести ее в машинный код, который представлен в виде ноликов и единичек. Для этого используется два инструмента, таких, как компилятор и интерпретатор. Компилятор используется для того, чтобы собрать весь текст программы и превратить в исполняемый код, набор команд для процессора. Интерпретатор работает немного иначе, выполнение команд происходит сразу после их получения, а не выполняет огромное количество команд, как компилятор. Но их объединяет одно, это сложный набор правил, при помощи которых язык программирования превращается в машинный код. Например, интерпретатор Python написан на языке C, который в свою очередь написан на языке программирования Ассемблер, который практически является языком машинного кода.[3]

Каждый язык программирования использует готовые шаблоны, которые называются библиотеками. Библиотеки являются набором функций

или готовых шаблонов, которые были написаны почти на любом другом языке программирования.[4]

2.2 Почему именно python

Python, это язык программирования, который в сегодняшнем технологическом ландшафте использует большинство разработчиков, представленный язык программирования существует с начала девяностых годов. Его оптимизированные функции и многочисленные преимущества позволяют создавать конкуренцию в разработке программного обеспечения с другими языками программирования.[5]

Он является бесплатным языком программирования и всегда им будет. Набор инструментов расширения и вспомогательных функций, библиотек и модулей является абсолютно бесплатным. Большинство популярных интегрированных средств разработки под названием IDE, учитывая такие, как PTVS, Pydev, Eclipse и Spyder Python, загружаются бесплатно и находятся в открытом доступе. Это языки с открытым исходным кодом, они являются такими благодаря компании Python Software Foundation.

Данный язык программирования используют крупнейшие технологические компании. Python является приоритетным языком программирования для большинства больших компаний в мире информационных технологий. К таким компаниям можно отнести Google, Dropbox, Instagram и Spotify, и это лишь малая часть компаний, которые используют данный язык программирования. Даже за пределами ИТ индустрии многие компании, такие как, Disney, NASA и Electronic Arts, так же используют в приоритете язык программирования Python, который они используют в технологических стартапах.

С Python легко работать, что сокращает время разработки и введение персонала в курс дела. Написанный в формате удобном для чтения Python делает процесс разработки программного обеспечения быстрым, удобным и

максимально эффективным. Если разработчик имеет хотя бы базовые знания любого другого языка программирования, он сможет достаточно легко изучить принцип работы языка программирования Python и использовать его в своих проектах.[6]

В сравнении с другими языками программирования Python в 5 – 10 раз быстрее по времени разработки, однако такой же скоростью выполнения программ он не сможет похвастаться. Он позволяет достаточно широко управлять процессами и объектно-ориентированным дизайном проекта, это помогает как в скорости, так и в производительности. Упрощенный контекст и структура данных удобная для пользователя, позволяет легко писать и читать код. Сокращенное время разработки программ на языке Python так же сокращает затраты пользователей и разработчиков на проекты.

Скорее всего, если разработчик столкнется с проблемой в своем проекте или обнаружит ошибку, другой разработчик легко найдет решение этой проблемы. Python поддерживается большим онлайн-сообществом людей, которые являются сторонниками или программистами на других языках программирования, они постоянно улучшают функционал данного языка/ Кроме того при использовании Python можно легко получить быструю поддержку со стороны других пользователей о возникшей проблеме.

Этот язык программирования является очень гибким и масштабируемым. Он позволяет разработчикам оптимизировать высокоуровневую логику программ без изменения требований или внесения изменений в логику компонентов, которые являются базовыми. Это позволяет легко взаимодействовать с расширением сложных проектов по мере необходимости.[7]

Использовать Python можно так же для машинного обучения, он является одним из лучших языков программирования для использования поэтому направлению. Можно не тратить силы и время на изучение сложных языков программирования, а создавать свои проекты по искусственному интеллекту именно в языке программирования Python.

3 Oracle

3.1 Базы данных

Базы данных Oracle можно разделить на 4 разные версии. Каждая из которых имеет отличительный характер по отношению друг к другу, например, такие как доступность, производительность, масштабируемость и безопасность.[8]

Рассмотри несколько версий Oracle. Oracle database Standard Edition One является версией для платформы windows и поддерживает все функции Enterprise Editions, кроме возможности использования кластерной технологии.

Oracle database Standard Edition является экземпляром базы данных, который использует возможности критически важных бизнес приложений. Такая функция, как SE1 обеспечивает программе отказоустойчивость, производительность и предоставляет возможность использования web приложений. Для использования функций данного экземпляра базы данных необходимо лишь разместить данный экземпляр базы данных на сеовере под управлением Windows или Unix систем, он используется для малых бизнес предприятий.[9]

Oracle database Enterprise Edition имеет ряд преимуществ, которые позволяют повысить уровень защиты от сбоев, которые может допустить пользователь и уменьшает время простоя программы при выполнении плановых работ на экземпляре базы данных. Так же повышена защита базы данных от несанкционированного доступа по средствам прозрачного аудита и шифрования данных. Обеспечивается высокая производительность хранилища данных, возможность онлайн обработки и анализа данных. Легкость администрирования больших объемов данных, которые хранятся в базе данных.

Oracle database Express Edition является базой данных начального уровня, которая используется для работы с небольшими объемами информации. Основа данной версии построена на коде более продвинуты версий баз данных и является бесплатной для разработки, распространения и развертывания, достаточно проста в установке и администрировании.[10]

3.2 Использование в организациях.

Oracle Database Enterprise Edition может предоставить надежное, эффективное и безопасное управление данными критически важных приложений для бизнеса, например, онлайн среды, выполняющие масштабную обработку транзакций, под названием OLTP, это хранилища данных с высокой интенсивностью потока запросов, а так же интернет-приложения, которые являются ресурсоемкими. Редакция Oracle Database Enterprise Edition обеспечивает инструментальными средствами и функциями, которые позволяют соблюдать современные требования корпоративных приложений в области доступности и масштабируемости. Данная редакция включает в себя все компоненты Oraclr Database и допускает расширение используя средства приобретения дополнительных модулей и приложений.

По средствам использования Oracke Database, у компоний появляется возможность управлять полностью всей корпоративной информацией и лучше понимать собвственый бизнес, а также быстро и просто адаптироваться к конкуренции, которая становится все более изменчивой. Для предоставления таких возможностей были расширены возможности, которые являюьтся уникальными механизмами Oracle, они обеспечивают кластеризацию базы данных, управление рабочими нагрузками и автоматизацию центров обработки данных. При использовании защищенных, масштабируемых grif-инфраструктур, на базе серверов имеющих малую стоимость, пользователи Oracle могут создавать OLTP приложения, это

хранилище данных и система, которая управляет контентом с наивысшими требованиями.[11]

Real Application Testing является модулем, который сокращает время, затраты и риски для внесения правок.

Oracle Database предоставляет новейший функционал, который позволяет самому управлять и автоматизировать процессы, для помощи организациям, которые соблюдают соглашения об уровне обслуживания. Такими организациями, которые регулярно обновляют операционные системы, версии СУБД и вносят изменения в системные и аппаратные конфигурации. Так же Oracle предоставляет систему Oracle Real Application Testing, которая позволяет клиентам проводить процедуры тестирования и управления в своих ИТ-средах быстро и точно.

4 Справочники

4.1 Что такое справочник

Это прикладные объекты конфигурации. Они позволяют хранить в информационной базе данных информацию, которая имеет схожую структуру и вид списка. Это может быть почти любая информация, например, перечень товаров или список сотрудников.[12]

4.2 Структура справочников

Все элементы справочника должны характеризоваться кодом и наименованием. В базах данных используется система автоматической нумерации объектов справочника, таким образом система сама может указывать код для нового элемента справочника. Кроме этого должен осуществляться контроль уникальных наименований в справочнике и запрет на создание однотипных кодов элемента справочника.

Каждый элемент справочника содержит еще и дополнительную информацию, кроме кодов и наименований, эта информация должна подробно описывать этот элемент справочника. Это может быть информация о названии кафедры или факультета к которую привязан конкретный студент, эта информация служит реквизитом для справочника.

Справочник обязательно имеет одинаковую структуру заполнения, но информация, которая хранится в полях справочника может отличатся, как например номера телефонов сотрудников или электронные адреса почты. Для хранения такой информации используются определенные табличные ячейки, в которые записывается данная информация.

В справочниках может присутствовать так же иерархическая структура построения записей, например, по названию факультетов можно получить всех студентов, которые связаны с этим факультетом, таким образом можно

формировать определенные группы в справочниках. Так же в группы справочников могут включаться и другие группы, тем самым создавая многоуровневую структуру хранения информации.

Возможен и другой вид хранения информации в справочниках по средствам иерархических структур, этот вид представлен в форме того, что конкретная запись может относиться не к конкретной группе записи, а к точно указывать на одну из записей. Таким образом можно построить запись с названием задачи, которая так же будет включать в себя и подзадачи.

Справочники могут находиться в подчинении, когда одни элементы справочника подчиняются другим элементам другого справочника или конкретным группа. В пример можно привести выбор преподавателя, когда одного преподавателя можно найти не из всего списка преподавателей, а просмотреть список преподавателей, привязанных к конкретной кафедре и найти нужного.[13]

При создании базы данных могут быть созданы преопределенные элементы, которые создает разработчик при разработке прикладного решения, и они никак не зависят от действий пользователя, то есть не могут быть редактированы или удалены. В пример можно привести справочник стран, если справочник разрабатывается для России, там обязательно должна быть страна Россия, это предопределенный элемент, который обязательно понадобится для дальнейшей работы.[14]

5 Библиотеки используемы при разработке

5.1 Cx_Oracle

Для разработки программы были использованы некоторые основные вспомогательные библиотеки, без которых невозможно было бы построить имеющуюся структуру программы.

В качестве базы данных используется Oracle, поэтому для работы с этой базой данных и для того что бы иметь возможность обращаться к ней, была выбрана самая популярная библиотека для работы с базами данных под управлением СУБД Oracle, она называется cx_Oracle и метод подключения изображен на рисунке 6.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.oracle',
        'NAME': 'VEDA',
        'USER': 'ineeds',
        'PASSWORD': '            ',
        'HOST': '10.15.0.22',
        'PORT': '1521',
    }
}
```

Рисунок 6– Подключение к базе данных

5.2 Fuzzywuzzy

Для реализации поиска по базе данных были использованы не стандартные средства базы данных, а язык программирования, который обрабатывает полученные данные, для этого была использована библиотека для анализа совпадения текста, которая называется fuzzywuzzy, фрагмент кода, написанный на сервере изображен на рисунке 7.

```
for row1 in cursor.execute('select * from ' + str(data.split(":")[1])):
    for i in range(0, len(row1)):
        if fuzz.WRatio(str(data.split(":")[2].replace("_", " ")), str(row1[i])) > 86 and len(
                str(data.split(":")[2]).replace("_", " ")) <= len(str(row1[i])):
            s += "<tr>"
            tochnost = 1
            for i in range(0, len(row1)):
                s += "<td>" + str(row1[i]) + "</td>"
```

Рисунок 7 – Анализ поискового запроса

5.3 JsonResponse

Для получения ответов и запросов на сервере используется библиотека JsonResponse, которая позволяет получать данные с клиента на сервер и отправлять данные с сервера клиенту.

Функция, которая находится на серверной части приложения должна получать, обрабатывать запросы и выдавать ответ клиенту, запросы разделены на 2 вида, они называются «POST» и «GET», если к серверу приходит запрос на получение информации с определенной ссылки по запросу «GET», сервер обрисовывает страницу, для пользователя, по средствам которой он может отправить «POST» запрос на обработку введенных данных и дальнейший ответ сервера, фрагмент кода с сервера с «GET» и «POST» запросами изображен на рисунке 8.

```
def index(request):
    if request.method == "GET":
        return render(request, 'login.html')
    if request.method == "POST":
        if request.is_ajax():
            search = request.POST
            s = search.dict()
            search = s['data']
```

Рисунок 8 – Обработка ajax запроса с сервера

Для того, чтобы после отображения страницы пользователь мог отправить «POST» запрос на сервер для дальнейшей обработки данных на сервере и отображения актуальной информации на странице с которой взаимодействует пользователь, на стороне клиента написан ajax запрос, при помощи которого с клиентской части сервера будет отправляться и приниматься информация с сервера, фрагмент кода ajax запроса на стороне клиента представлен на рисунке 9.

```
$.ajax({
        type: "POST",
          data: {
              "data": data,
              csrfmiddlewaretoken: '{{ csrf_token }}',
            }, // получаем данные формы
          url: "http://127.0.0.1:8000/calculator/",
          // если успешно, то
          success: function (response) {
```

Рисунок 9 – Ajax запрос на стороне клиента

5.4 Connection

Для динамического подключения к другим базам данных была использована библиотека Connection. При помощи которой был реализован переход между основной базой данных и одной из подключенных баз данных. В Django по стандарту можно подключить лишь фиксированное количество баз данных и использовать их, а этот метод позволит при желании использовать неограниченное количество подключений.

6 Схема пользовательского интерфейса

Для лучшего понимания работы программы рассмотрим схему пользовательского интерфейса, где показаны все возможны варианты взаимодействия с программой, схема представлена на рисунке 10.

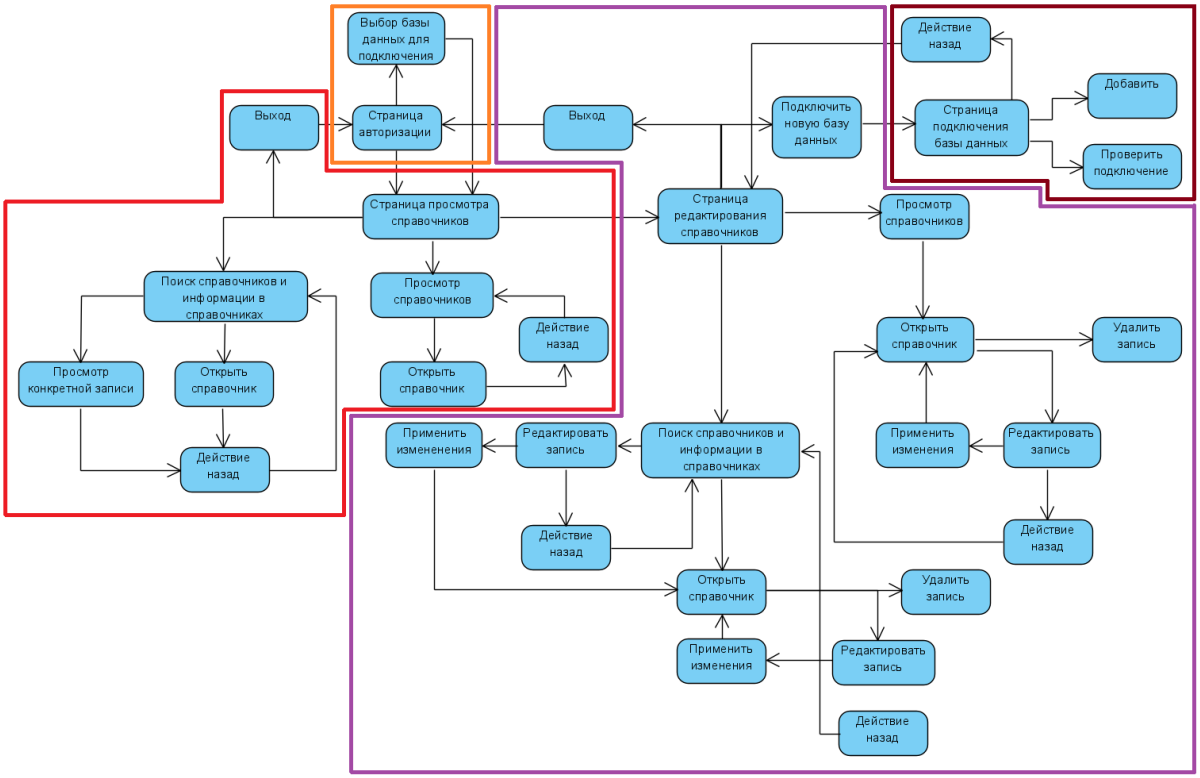


Рисунок 10 – Диаграмма пользовательского интерфейса

Весь интерфейс разделен на 4 страницы, которые на диаграмме разделены линиями.

6.1 Страница авторизации

На рисунке 11 представлена часть диаграммы пользовательское интерфейса, которая содержит в себе функционал страницы входа.

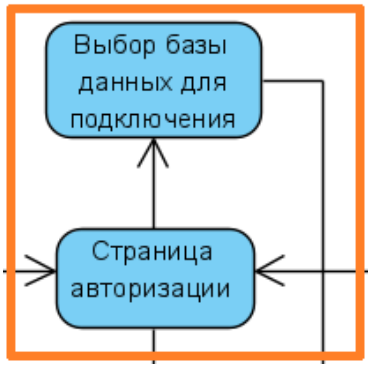


Рисунок 11 – Функционал страницы авторизации

Страница авторизации используется лишь для выбора базы данных, к которой можно подключиться и входа в систему по средствам авторизации.

6.2 Страница просмотра справочников

После авторизации пользователь попадает на главную страницу, которая называется «Страница для просмотра справочников», диаграмма пользовательского интерфейса для этой страницы показан на рисунке 12.

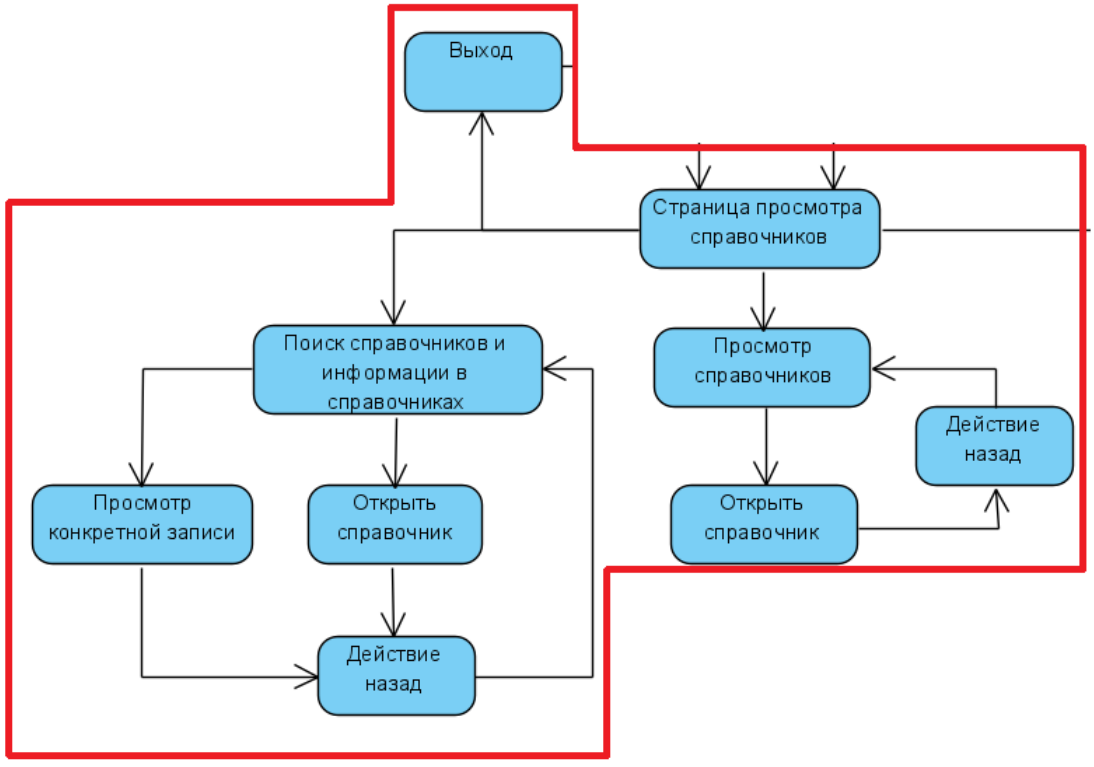


Рисунок 12 – Диаграмма пользовательского интерфейса главной страницы

Страница для просмотра справочников имеет функционал для просмотра справочников без возможности дальнейшего взаимодействия с ними. На этой странице реализован поиск по базе данных и просмотр всех имеющихся справочников, а также записей, находящихся в этих справочниках.

6.3 Страница редактирования справочников

На странице для просмотра справочников реализована кнопка выхода из системы, для перехода на страницу авторизации и кнопка перехода на страницу для редактирования справочников, под названием «К редактированию», диаграмму пользовательского интерфейса страницы для редактирования справочников можно увидеть на рисунке 13.

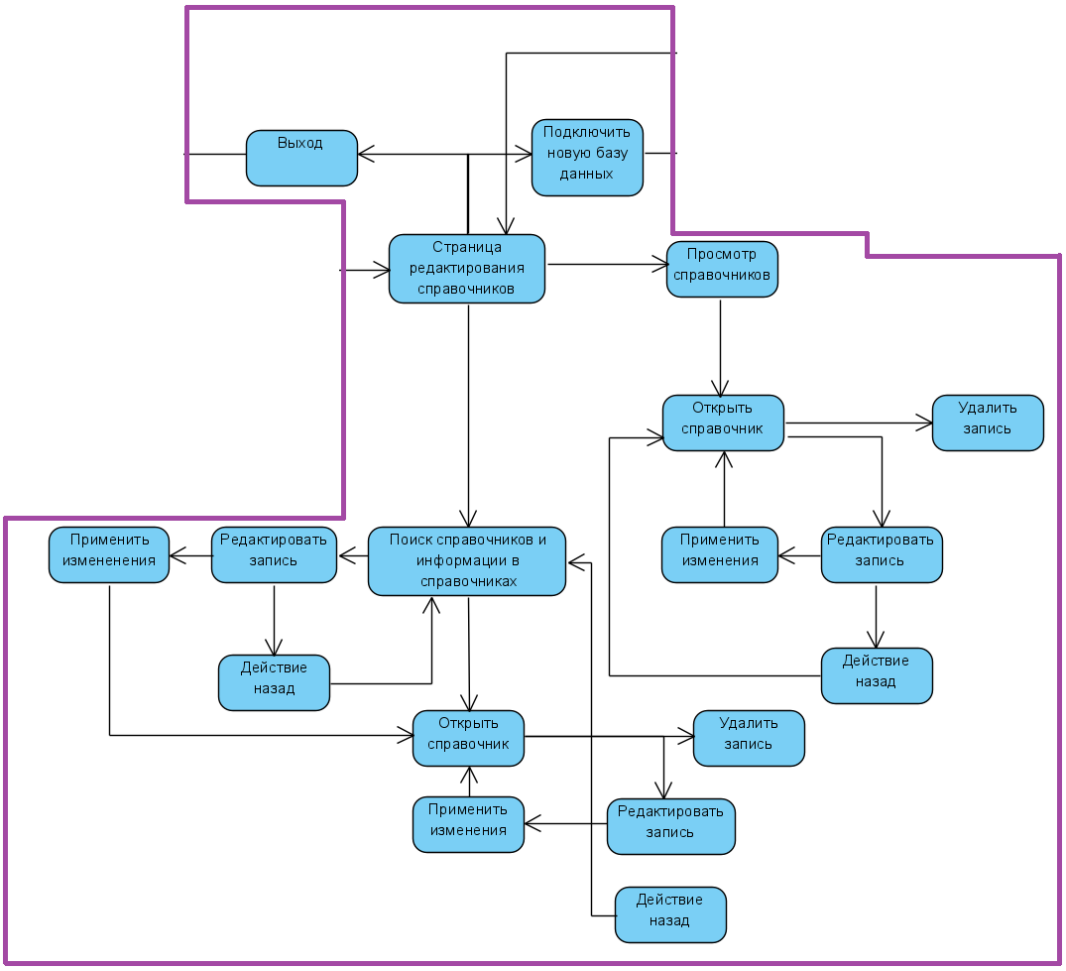


Рисунок 13 – Диаграмма страницы для редактирования справочников

Страница для редактирования справочников обладает достаточно большим функционалом и возможностями, по средствам взаимодействия с интерфейсов данной страницы реализована возможность полного управления записями, находящимися в справочниках. Так же реализован поиске данных находящихся в справочниках и самих справочников при помощи поисковой строки, которая предоставляет расширенный функционал в отличие от страницы просмотра справочников в виде возможности редактирования найденных записей сразу из окна поиска.

6.4 Страница для подключения БД

На страницы для редактирования справочников так же есть функционал для выхода из системы, но еще добавлена кнопка для создания новых подключений к базам данных, при помощи которой можно попасть на страницу с формой для ввода параметров к новой базе данных, диаграмма пользовательского интерфейса страницы для подключения дополнительных баз данных показана на рисунке 14.

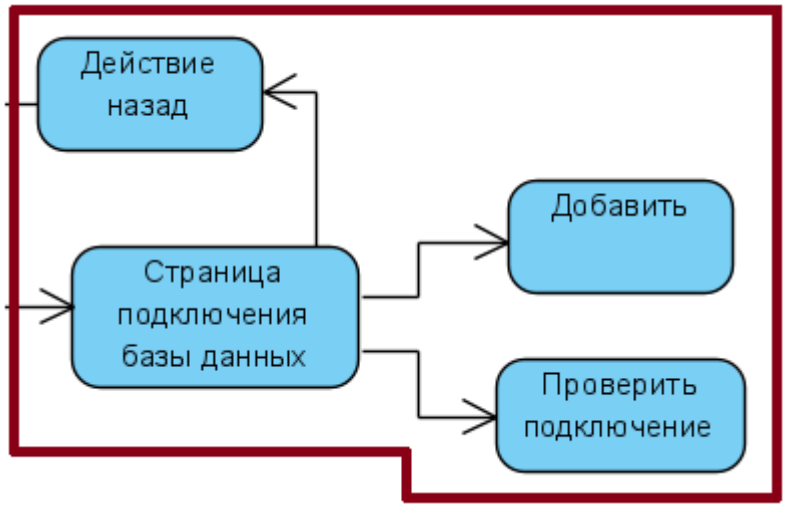


Рисунок 14 – Диаграмма ПИ для страницы подключения новых БД

Страница для добавления новых справочников содержит в себе форму, по средствам заполнения которой можно проверить подключение к базе данных

и добавить данное подключение, после чего оно будет добавлено в интерфейс страницы авторизации для дальнейшего использования подключенной базы данных. Интерфейс так же содержит кнопку для возвращения к предыдущей странице, которая называется «Страница для редактирования справочников».

7 Обзор интерфейса программы

7.1 Страница авторизации

Данная программа рассчитана на работу с базами данных в которых может храниться конфиденциальная информация, поэтому для защиты данных пользователей обязательно должна присутствовать возможность защитить данные от посторонних. Внешний вид страницы авторизации представлен на рисунке 15.

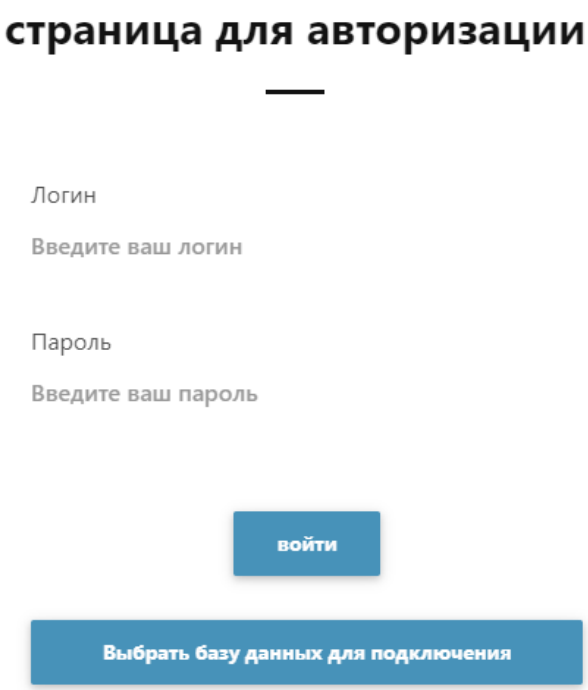


Рисунок 15 – Страница авторизации без введенных данных

По средствам браузера Google Chrome можно воспользоваться сохранением введенных данных для более быстрого входа в приложение. Данные будут автоматически подставляться в поля, если вы разрешите браузеру их сохранить для дальнейшего использования, это представлено на рисунке 16.

Рисунок 16 – Страница авторизации с заполненными данными

Страница для авторизации содержит в себе несколько элементов. В самом верху страницы отображается информация о том, что вы сейчас находитесь на странице авторизации и вам нужно ввести логин или имя пользователя, а также пароль, которые записаны в базе данных, для того что бы вы имели возможность получить доступ к сайту.

Возможность регистрации в данной программе не предусмотрена. Данные для входа должны заполняться специалистом непосредственно в самой базе данных, что бы никто не мог просто так получить доступ к данным, которые хранятся в базе данных.

В нижней части страницы расположена кнопка «Выбрать базу данных для подключения», после ее нажатия появится окно с доступными подключениями, которые пользователь сможет добавить сам в пользовательском интерфейсе программы после авторизации, после нажатия на кнопку на экране авторизации будет отображена таблица с возможными вариантами для подключения, это можно увидеть на рисунке 17.

Рисунок 17 – Выбор базы данных для подключения

По стандарту без выбора базы данных для подключения будет выбрана основная база данных, которая введена в программе, а если выбрать базу данных для подключения по средствам взаимодействия с представленной таблицей, в которой показаны возможные варианты подключения к базам данных, можно подключиться к конкретным выбранным базам данных. В интерфейсе представляется таблица, которая имеет две колонки под названием «Название базы данных» и «Действие», в первом столбце находится название базы данных как в подключении, а во втором столбце находится кнопка для выбора этой базы данных, у каждой строки имеется своя кнопка взаимодействия.

7.2 Главная страница

Когда пользователь проходит авторизацию на странице для входа, он попадает на главную страницу, которая предназначена для просмотра справочников и поиска, как самих справочников, так и их содержимого, без возможности редактирования. Интерфейс главной страницы представлен на рисунке 18.

Рисунок 18 – Страница для просмотра справочников

Рассмотрим более подробно функционал данной страницы. Для начала можно заметить кнопку «Выйти», расположенную в правом верхнем углу страницы, она позволяет выйти из аккаунта и закончить сеанс взаимодействия с программой. Так же слева сверху расположена кнопка «К редактированию», которая позволяет перейти на другую страницу, предназначенную для редактирования информации, которая находится в справочниках. Основным элементом для взаимодействия можно считать кнопку «Показать добавленные», при ее нажатии отобразится список справочников, доступных пользователю для взаимодействия, это можно увидеть на рисунке 19.

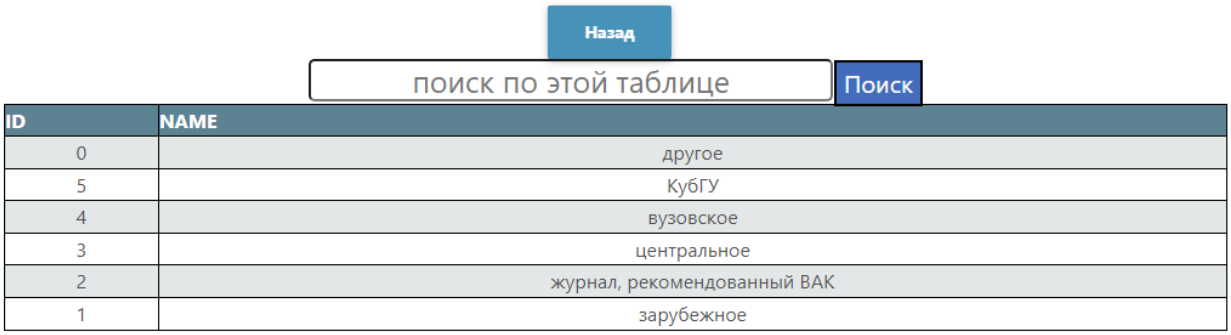
NAME	DESCRIPTION	Action
P_C_TYPE	Вид конференции	открыть
P_PUBL_DIRS	Научное направление	открыть
P_PUBL_TERRS	Уровень издания	открыть
P_CONF_TYPES	Уровень проведения	открыть
P_PUBL_TYPES	Вид издательства	открыть
P_UCH_STYPES	Вид издания	открыть
P_UCH_TERRS	Место издания	открыть
P_UCH_GRIFS	Гриф	открыть
P_TYPES	Вид публикации	открыть
COUNTRY	Страна	открыть
SPYFACULTIES	проверка	открыть

Рисунок 19 – Доступные справочники

Добавление новых справочников осуществляется специалистом, по средствам прямого взаимодействия с базой данных, в интерфейсе не предусмотрена возможность свободного добавления новых справочников обычным пользователем.

Когда отображаются доступные справочники, мы видим таблицу, состоящую из трех столбцов, где в первом столбце под названием «NAME», написано название таблицы как в базе данных, далее идет столбец с названием «DESCRIPTION», в котором отображается описание таблицы, созданное специалистом при добавлении справочников доступных пользователю и в третьем столбце под названием «Action» находятся возможные действия с конкретной строкой, в данном случае взаимодействие

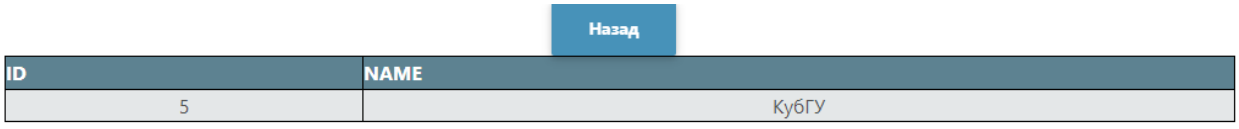
представлено в виде кнопки «Открыть», по средствам которой можно получить доступ к содержимому данного справочника, но это не единственный способ для открытия содержимого справочника, также можно взаимодействовать с названием справочника, которое представлено в первом столбце таблицы, оно оформлено в виде ссылки, на которую можно нажать и точно также получить доступ к содержимому справочника, как и при взаимодействии с кнопкой «Открыть», результат можно увидеть на рисунке 20.



ID	NAME
0	другое
5	КубГУ
4	вузовское
3	центральное
2	журнал, рекомендованный ВАК
1	зарубежное

Рисунок 20 – Информация в справочнике

Рассмотрим рисунок 20 более подробно. Для начала стоит обратить внимание на возможность вернуться назад к списку всех справочников, по средствам взаимодействия с кнопкой, расположенной сверху посередине, которая называется «Назад». В представленной таблице находится пример данных, которые могут храниться в справочнике, в данном случае это импровизированный справочник студентов в котором всего 2 столбца под названиями «ID» и «NAME». Так же есть строка поиска, при помощи которой можно найти информацию в этой таблице, пример работы поиска по точному запросу фамилии студента можно увидеть на рисунке 21.



ID	NAME
5	КубГУ

Рисунок 21 – Результат поиска по слову «кубгу»

Если поисковый запрос не выдает однозначный ответ, то будет представлен список возможно подходящих вариантов совпадений под поисковый запрос, это можно увидеть на рисунке 22.

Назад

ID	FLAG	NAME	LOCAL	MEDIAN_IMPACT	AGG_IMPACT	IMPACT_ORIG
19	100219	Теория языка	1	0.487	0.844	0
80001	080001	Экономическая теория	1	0.778	1.148	0
90001	090001	Онтология и теория познания	1	0.421	0.668	1
100108	100108	Теория литературы. Текстология	1	0.487	0.844	0
130008	130008	Теория и методика профессионального образования	1	0.807	1.275	0

Рисунок 22 – Результат поиска по слову «теория»

В результате поиска по таблице будет показан один или несколько результатов, подходящих под поисковый запрос, а также будет представлена кнопка «Назад» для возвращения обратно к данным хранящимся в справочнике.

Так же стоит рассмотреть функционал поиска по всем справочникам и их содержимому. Что бы им воспользоваться нужно ввести ключевое слово в строку для ввода текста и нажать кнопку с названием «Поиск», после чего будут показаны результаты поиска, это можно увидеть на рисунке 23.

Рисунок 23 – Результат глобального поиска

Результатом поиска является таблица, которая содержит в себе информацию о том в каком справочнике найдена информация и сама информация. В данном случае, по запросу «кубгу» было найдено 4 совпадения в таких справочниках, как «уровень издания», «Уровень проведения», «Вид издательства» и «Вид награды» а ниже представлена информация, которую удалось найти в этих справочниках. Например, в справочнике «Уровень издания» есть лишь одна запись с поисковым запросом, при помощи кнопки «открыть», которая находится в синем поле во втором столбце таблицы, сразу после названия справочника, можно осуществить переход к этому справочнику и просмотреть всю информацию, находящуюся в нем, а по средствам кнопки «назад» можно вернуться обратно к поиску, но также можно просмотреть и конкретную запись, которая была найдена в ходе поиска по ключевому слову, при помощи взаимодействия с кнопкой «Показать», тогда будет открыта запись полностью и можно просмотреть всю информацию в найденной строке, это можно увидеть на рисунке 24.

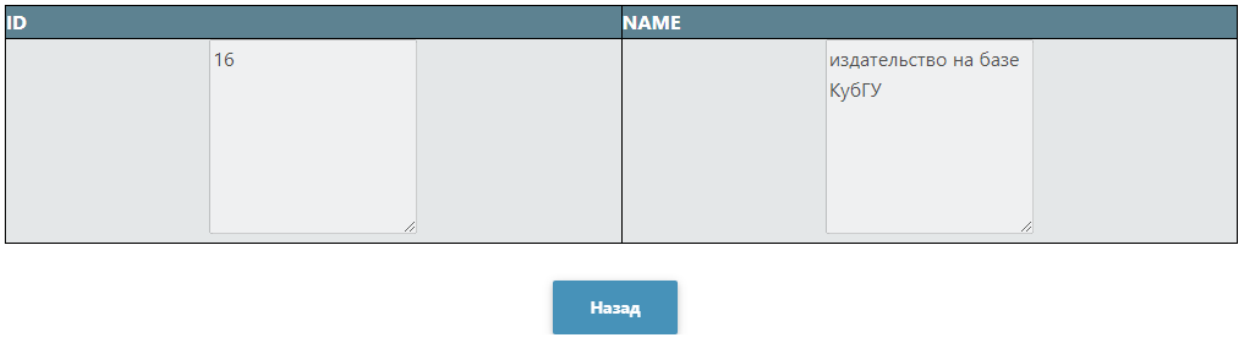


Рисунок 24 – Просмотр найденной записи

В появившейся таблице представлены названия полей справочника и информация, которая содержится в каждом поле для найденной строки. Так же реализована возможность возврата назад после перехода к этому окну, по средствам взаимодействия с кнопкой «Назад».

7.3 Страница для редактирования

На главной странице была кнопка перехода на страницу для редактирования, которая по внешнему виду практически ничем не отличается от страницы для просмотра справочников, это можно увидеть на рисунке 25, но привносит новый функционал в работу со справочниками.

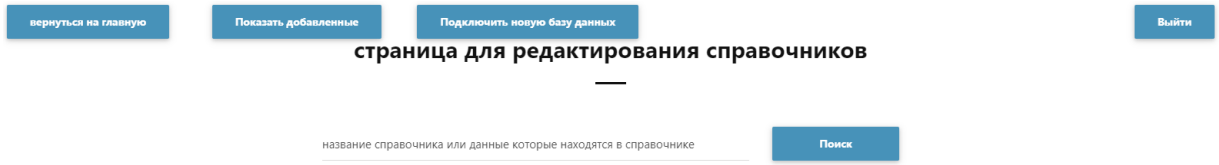


Рисунок 25 – Внешний вид страницы для редактирования справочников

Основным элементом управления опять же можно назвать кнопку, расположенную на странице слева сверху, под названием «Показать добавленные», отображается тот же самый список справочников доступных для взаимодействия и их так же можно открыть, это можно увидеть на рисунке 26.

Рисунок 26 – Информация в справочнике с дополнительным функционалом

Рассмотрим рисунок 26 более подробно. Для начала стоит обратить внимание на возможность вернуться назад к списку всех справочников, по средствам взаимодействия с кнопкой, расположенной сверху посередине,

которая называется «Назад». В представленной таблице находится та же информация, что и в таблице, отображаемой на странице для просмотра справочников, но появляется дополнительный столбец названием «Действие», который показывает какие действия можно осуществлять с каждой строкой, находящейся в таблице, на первой строке представлены пустые поля, по средствам заполнения которых и нажатием кнопки «добавить» можно пополнять имеющийся список данных, добавленные данные будут сразу же отображены в таблице. После строки с добавлением данных сразу же идет информация о имеющихся данных с которой так же можно взаимодействовать и на каждой строке в столбце «Действие» представлены две кнопки, под названием «удалить» и «редактировать», кнопка удаления позволяет удалить выбранную запись, но после нажатия на эту кнопку появится дополнительное окно, в котором нужно подтвердить выполнение действия, это можно увидеть на рисунке 27.

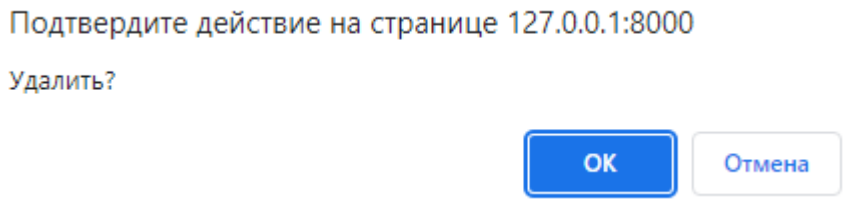


Рисунок 27 – Подтверждение удаления

Подтверждение удаления необходимый аспект в работе, чтобы случаем не нажать на эту кнопку и не удалить нужную информацию, это действие можно отменить, нажав кнопку «Отмена» или выполнить, нажав кнопку «ОК», после согласия с удалением строки, запись будет сразу же удалена, а в таблице отобразится актуальная информация, находящаяся в справочнике.

Рядом с кнопкой «Удалить», находится кнопка «Редактировать», которая позволяет открыть строку в новой таблице и редактировать все поля, это видно на рисунке 28.

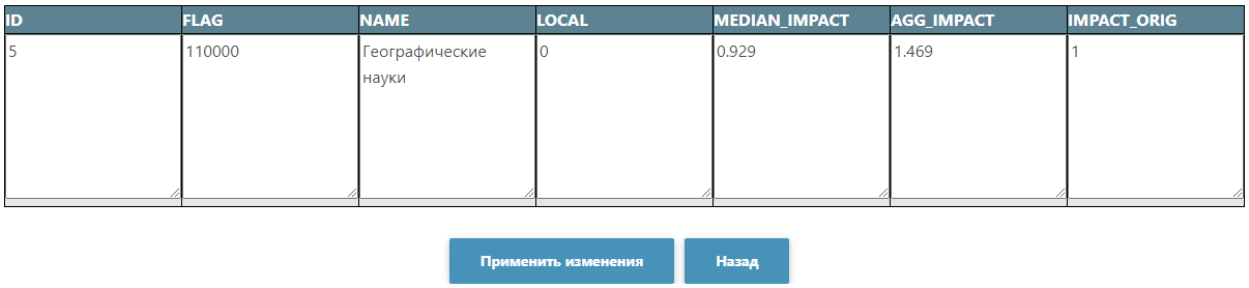


Рисунок 28 – Редактирование записей

Данный интерфейс позволяет применить изменения, внесенные в запись справочника и сразу же применить изменения после нажатия кнопки «Применить изменения», но что бы применить изменения, нужно после нажатия кнопки так же подтвердить действие, нажатием кнопки «ОК» в сплывающем окне или же отменить его, нажатием кнопки «Отмена», это показано на рисунке 29. Так же в интерфейсе есть кнопка возвращения обратно к списку всех данных находящихся в справочнике, которая называется «Назад».

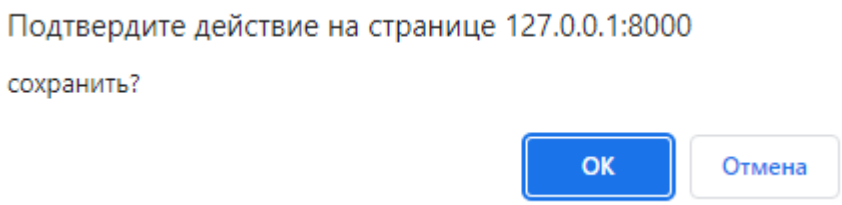


Рисунок 29 – Подтверждение изменений

В остальном страница для редактирования справочников ничем не отличается от страницы для просмотра справочников, кроме как тем, что можно взаимодействовать с данными находящимися в справочнике, глобальный поиск и поиск по таблицам работает точно так же, только с возможность. редактировать найденные данные.

7.4 Страница для добавления подключения к базе данных

Для того что бы попасть на страницу добавления новых подключений к базам данных, нужно нажать кнопку на странице редактирования в верхней части экрана, которая называется «Подключить новую базу данных». Внешний вид страницы для подключения новых баз данных можно увидеть на рисунке 30.

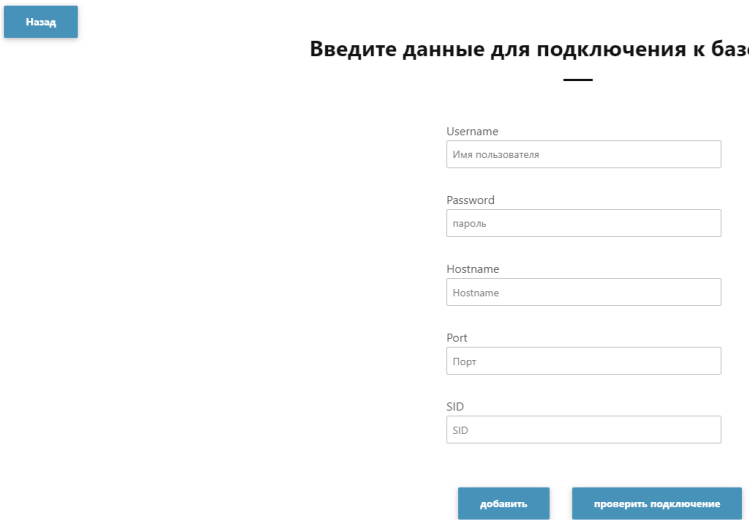


Рисунок 30 – Страница для добавления подключения к базе данных

Интерфейс страницы для подключения новой базы данных представлен в виде заголовка, который указывает на то, что пользователь должен ввести данные для подключения к новой базе данных, текстовых полей, в которые нужна заполнить необходимую информацию для подключения и кнопок для проверки подключения, добавления базы данных и возвращения на страницу для редактирования справочников.

ЗАКЛЮЧЕНИЕ

Разработанная программа направлена на использование обычным пользователем, который не является специалистом в области работы с базами данных. Но также может быть использована и специалистами в качестве ознакомительного инструмента для более лучшего понимания структуры базы данных или быстрого доступа к конкретным элементам справочников без необходимости разбираться в огромной базе данных. Данная программа предоставляет базовый функционал взаимодействия уже с существующими базами данных под управлением Oracle.

Представленная программа имеет ряд недоработок, которые необходимо исправить перед тем, как использовать ее в профессиональной деятельности, поскольку в программе используется достаточно слабая система безопасности, что не позволит обезопасить все данные должным образом.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1 Hierarchical Architecture [Электронный ресурс] – URL: https://www.tutorialspoint.com/software_architecture_design/hierarchical_architecture.htm (Дата обращения 14.04.22)

2 Гурьянова К.Н. Математический анализ / К.Н. Гурьякова, У.А. Алексеева, В.В. Бояршинов. – Екатеринбург: Издательство Уральского университета, 2014. – 332 с. – ISBN 978-5-7996-1340-2

3 Handcalcs: Python calculations in Jupyter, as though you wrote them by hand [Электронный ресурс] – URL: https://github.com/connorferster/handcalcs (Дата обращения 16.04.22)

4 Django введение [Электронный ресурс] – URL: https://developer.mozilla.org/ru/docs/Learn/Server-side/Djan4go/Introduction (Дата обращения 16.04.22)

5 Writing your first Django app [Электронный ресурс] – URL: https://docs.djangoproject.com/en/4.0/intro/tutorial01/ (Дата обращения 20.04.22)

6 Database PL/SQL User's Guide and Reference [Электронный ресурс] – URL: https://docs.oracle.com/ (Дата обращения 20.04.22)

7 Руководство администратора [Электронный ресурс] – URL: http://oracle.bulldogss.com/ (Дата обращения 01.05.22)

8 HTML Tutorial [Электронный ресурс] – URL: https://w3schools.com/html/default.asp (Дата обращения 01.05.22)

9 CSS Tutorial [Электронный ресурс] – URL: https://www.w3schools.com/css/default.asp (Дата обращения 01.05.22)

10 Python for Web Development [Электронный ресурс] – URL: https://www.stxnext.com/what-is-python-used-for/ (Дата обращения 5.05.22)

11 Why Is Python So Popular? An Introduction to The World's Favorite Programming Language [Электронный ресурс] – URL: https://learnpython.com/blog/why-is-python-so-popular/ (Дата обращения 10.05.22)

12 The Hitchhiker’s Guide to Hierarchical Classification [Электронный ресурс] – URL: https://towardsdatascience.com/https-medium-com-noa-weiss-the-hitchhikers-guide-to-hierarchical-classification-f8428ea1e076 (Дата обращения 10.05.22)

13 Modeling Structure with Blocks [Электронный ресурс] – URL: https://www.sciencedirect.com/topics/computer-science/classification-hierarchy (Дата обращения 15.05.22)

ПРИЛОЖЕНИЕ А

Основная программа

```
from django.shortcuts import render
from django.http import JsonResponse
from django.db import connection
from fuzzywuzzy import fuzz

def index(request):
    if request.method == "GET":
        return render(request, 'login.html')
    if request.method == "POST":
        if request.is_ajax():
            search = request.POST
            s = search.dict()
            search = s['data']
            a = str(search).upper()
            splite = a.split()
            global auth
            if str(splite[0]).split(":")[0] == "SELECTDATABASE":
                data = select_database(search)
            if str(splite[0]).split(":")[0] == "EDIT" and:
                data = edit(search)
            if str(splite[0]).split(":")[0] == "EDITSTRING":
                data = edit_string(search)
            if str(splite[0]).split(":")[0] == "STRING":
                data = string(search)
            if str(splite[0]).split(":")[0] == "SEARCHADDTABLE":
                data = search_add_table(search)
            if str(splite[0]).split(":")[0] == "SEARCHTABLE" :
                data = search_table(search)
            if str(splite[0]).split(":")[0] == "SEARCHALLTABLE" :
                data = search_all_table(search)
            if str(splite[0]).split(":")[0] == "SEARCHINTABLE" :
                data = search_in_table(search)
```

```
            print(search)
        if str(splite[0]).split(":")[0] == "SEARCHINTABLEVIEW" :
            data = search_in_table_view(search)
            print(search)
        if str(splite[0]).split(":")[0] == "ADDSTRING" :
            data = addString(search)
        if str(splite[0]).split(":")[0] == "DELETE" :
            data = dell(search)
        if str(splite[0]).split(":")[0] == "CREATETABLE" :
            data = add(search)

        q = dict()

        if splite[0] == "HELP" :
            data = replace(":SPRAV_NAME")
        if str(splite[0]).split(":")[0] == "VIEW" :
            data = replace(search)

        if splite[0] == "HELP1" :
            data = edit_view(":SPRAV_NAME")
        if str(splite[0]).split(":")[0] == "EDIT_VIEW" :
            data = edit_view(search)
        if auth == 0:
            data = {"actionButton": "<h2>авторизация не пройдена</h2>"}
        return JsonResponse(data)

def select_database(data):
    if data != None:
        cursor = connection.cursor()
        send = dict()
        s ="<table><th>Название базы данных</th><th>Действие</th><tr>"
        for row in cursor.execute("""SELECT username FROM CONNECT_TABLE"""):
            s += "<td>" + str(row[0]) + "</td><td><button
class='edit'>выбрать</button></td></tr>"
        s += "</table>"
```

```
        send.update({"add": s})
        return send
def connect(data):
    if data.method == "GET":
        return render(data, 'connect.html')
    if data.method == "POST":
        if data.is_ajax():
            s = 0
    return data

def edit(data):
    if data != None:
        cursor = connection.cursor()
        s = "UPDATE " + str(data.split(":")[1]) + " SET "
        size = len(data.split(":"))-5
        size = int(size)/2
        s += str(data.split(":")[4+int(size)]) + " = '" + str(data.split(":")[4]) + "'"
        for i in range(int(size)-1):
            s += ", " + str(data.split(":")[5+i+int(size)]) + " = '" + str(data.split(":")[5+i]) + "'"
        s += " WHERE " + str(data.split(":")[3]) + " = '" + str(data.split(":")[2]) + "'"
        cursor.execute(s)
        send = edit_view(":" + str(data.split(":")[1]))
        return send

def edit_string(data):
    if data != None:
        diction = dict()
        cursor = connection.cursor()
        s = ""
        col = 0
        cursor.execute('select * from ' + data.split(":")[1])
        col_names = [row[0] for row in cursor.description]
        for i in col_names:
            s += "<th>" + str(i) + "</th>"
        s += "<tr>"
```

```
        for row in cursor.execute('select * FROM ' + data.split(":")[1].upper()):
            if str(row[0]) == str(data.split(":")[2]):
                dateTable = row[0]
                for i in row:
                    if str(i) == "None":
                        s += "<td><textarea rows=6 id='name" + str(col) + "'></textarea></td>"
                    else:
                        s += "<td><textarea rows=6 id='name" + str(col) + "'  >" + str(i) +
"</textarea></td>"
                    col += 1
        s += "</tr>"
        diction.update({"add": s})

        s = "<button class='btn btn-lpbuilder wave-effect'
onclick=if(confirm('сохранить?')){vala7('" + \
            str(data.split(":")[1]) + ":" + str(dateTable) + ":" + str(col_names[0]) + "')"
        for i in range(0, col):
            s += ",vala7(document.getElementById('name" + str(i) + "').value)"
        for i in range(0, col):
            s += ",vala7('" + str(col_names[i]) + "')"
        s += ",vala6();}else{event.stopPropagation();event.preventDefault();};>Применить
изменения</button>"
        if data.split(":")[int(len(data.split(":"))-1)] == "back":
            s += "<button onclick=vala('EDIT_VIEW:" + str(data.split(":")[1]) + \
                "') class='btn btn-lpbuilder wave-effect'>Назад</button>"
        else:
            s += "<button onclick=vala5('" + str(data.split(":")[int(len(data.split(":"))-1)]) + \
                "') class='btn btn-lpbuilder wave-effect'>Назад</button>"
        diction.update({"addButton": s})
        return diction

def string(data):
    if data != None:
        diction = dict()
        # print(data)
```

```
    cursor = connection.cursor()
    s = ""
    col = 0
    cursor.execute('select * from ' + data.split(":")[1])
    col_names = [row[0] for row in cursor.description]
    for i in col_names:
        s += "<th>" + str(i) + "</th>"
    s += "<tr>"
    for row in cursor.execute('select * FROM ' + data.split(":")[1].upper()):
        if str(row[0]) == str(data.split(":")[2]):
            dateTable = row[0]
            for i in row:
                if str(i) == "None":
                    s += "<td><textarea disabled rows=6 id='name" + str(col) + "' " + "'></textarea></td>"
                else:
                    s += "<td><textarea disabled rows=6 id='name" + str(col) + "'  >" + str(i) + "</textarea></td>"
                col += 1
    s += "</tr>"
    diction.update({"add": s})
    s = ""
    if data.split(":")[int(len(data.split(":")) - 1)] == "back":
        s += "<button onclick=vala('VIEW:" + str(
            data.split(":")[1]) + "') class='btn btn-lpbuilder wave-effect'>Назад</button>"
    else:
        s += "<button onclick=vala5('" + str(data.split(":")[int(len(
            data.split(":")) - 1)]) + "') class='btn btn-lpbuilder wave-effect'>Назад</button>"
    diction.update({"addButton": s})
    return diction

def search_add_table(data):
    if data != None:
        s ="<tr><th colspan=3>справочники с таким названием</th></tr><th>название таблицы</th><th>название справочника</th><th>действие</th>"
```

```
        diction = dict()
        flag = 0
        tochnost = 0
        cursor = connection.cursor()
        cursor1 = connection.cursor()
        for row in cursor.execute('select * from sprav_name'):
            for i in range(0, len(row)):
                if fuzz.WRatio(str(data.split(":")[1]).replace("_", " "), str(row[i])) > 86 and \
                        len(str(data.split(":")[1]).replace("_", " ")) <= len(str(row[i])):
                    flag = 1
                    s += "<tr>"
                    for i in row:
                        s += "<td>" + str(i) + "</td>"
                    s += "<td><button class=edit onclick=vala('EDIT_VIEW:" + row[0] + ":" + \
                         data.split(":")[1].replace(" ", "_") + "') >открыть</button></td>"
                    s += "</tr>"
        if flag == 0:
            s = ""
        s += "<tr><th colspan=3>найдены совпадения в следующих справочниках</th></tr>"
        for row1 in cursor.execute('select * from sprav_name'):
            o = 1
            for row2 in cursor1.execute('select * from ' + str(row1[0])):
                for i in range(0, len(row2)):
                    if fuzz.WRatio(str(data.split(":")[1].replace("_", " ")), str(row2[i])) > 86 and \
                            len(str(data.split(":")[1]).replace("_", " ")) <= len(str(row2[i])):
                        tochnost = 1
                        if o == 1:
                            s += "<tr><th colspan=2 >название справочника: " + row1[
                                1] + "</th><th><button class=edit onclick=vala('EDIT_VIEW:" +
str(row1[0]) + ":" + \
                                 data.split(":")[1].replace(" ", "_") + "') >открыть</button></th></tr>"
                            o = 0
                        s += "<tr>"
                        s += "<td colspan=2>" + str(row2[i]) + "</td>"
```

```
                s += "<td><button class=edit onclick=vala('EDITSTRING:" + str(row1[0]) + ":"
+ str(row2[0]) + ":" \
                     + data.split(":")[1].replace(" ", "_") + "') >показать</button></td>"
                s += "</tr>"
        if tochnost == 0:
            for row1 in cursor.execute('select * from sprav_name'):
                o = 1
                for row2 in cursor1.execute('select * from ' + str(row1[0])):
                    for i in range(0, len(row2)):
                        if fuzz.WRatio(str(data.split(":")[1].replace("_", " ")), str(row2[i])) > 85 and len(
                                str(data.split(":")[1]).replace("_", " ")) <= len(str(row2[i])):
                            if o == 1:
                                s += "<tr><th colspan=2 >название справочника: " + row1[
                                    1] + "</th><th><button class=edit onclick=vala('VIEW:" + str(row1[0]) +
":" + \
                                     data.split(":")[1].replace(" ", "_") + "') >открыть</button></th></tr>"
                                o = 0
                            s += "<tr>"
                            s += "<td colspan=2>" + str(row2[i]) + "</td>"
                            s += "<td><button class=edit onclick=vala('STRING:" + str(row1[0]) + ":" +
str(
                                row2[0]) + ":" + data.split(":")[1].replace(" ", "_") + "')
>показать</button></td>"
                            s += "</tr>"
        diction.update({"add": s})
    return diction

def search_table(data):
    if data != None:
        s ="<tr><th colspan=3>справочники с таким названием</th></tr><th>название
таблицы</th><th>название справочника</th><th>действие</th>"
        diction = dict()
        flag = 0
        tochnost = 0
        cursor = connection.cursor()
```

```
cursor1 = connection.cursor()
for row in cursor.execute('select * from sprav_name'):
    for i in range(0, len(row)):
        if fuzz.WRatio(str(data.split(":")[1]).replace("_", " "), str(row[i])) > 86 and len(str(data.split(":")[1]).replace("_", " ")) <= len(str(row[i])):
            flag = 1
            s += "<tr>"
            for i in row:
                s += "<td>" + str(i) + "</td>"
            s += "<td><button class=edit onclick=vala('VIEW:" + row[0] + ":" + data.split(":")[1].replace("_", " ") + "') >открыть</button></td>"
            s += "</tr>"
    if flag == 0:
        s = ""
    s += "<tr><th colspan=3>найдены совпадения в следующих справочниках</th></tr>"
    for row1 in cursor.execute('select * from sprav_name'):
        o = 1
        for row2 in cursor1.execute('select * from ' + str(row1[0])):
            for i in range(0, len(row2)):
                if fuzz.WRatio(str(data.split(":")[1].replace("_", " ")), str(row2[i])) > 86 and len(str(data.split(":")[1]).replace("_", " ")) <= len(str(row2[i])):
                    tochnost = 1
                    if o == 1:
                        s += "<tr><th colspan=2 >название справочника: " + row1[
                            1] + "</th><th><button class=edit onclick=vala('VIEW:" + str(row1[0]) + ":" + data.split(":")[1].replace(" ", "_") + "') >открыть</button></th></tr>"
                        o = 0
                    s += "<tr>"
                    s += "<td colspan=2>" + str(row2[i]) + "</td>"
                    s += "<td><button class=edit onclick=vala('STRING:" + str(row1[0]) + ":" + str(row2[0]) + ":" + data.split(":")[1].replace(" ", "_") + "') >показать</button></td>"
                    s += "</tr>"
    if tochnost == 0:
        for row1 in cursor.execute('select * from sprav_name'):
            o = 1
```

```
            for row2 in cursor1.execute('select * from ' + str(row1[0])):
                for i in range(0, len(row2)):
                    if fuzz.WRatio(str(data.split(":")[1].replace("_", " ")), str(row2[i])) > 85 and len(
                            str(data.split(":")[1]).replace("_", " ")) <= len(str(row2[i])):
                        if o == 1:
                            s += "<tr><th colspan=2 >название справочника: " + row1[
                                1] + "</th><th><button class=edit onclick=vala('VIEW:" + str(row1[0]) +
":" + \
                                 data.split(":")[1].replace(" ", "_") + "') >открыть</button></th></tr>"
                            o = 0
                        s += "<tr>"
                        s += "<td colspan=2>" + str(row2[i]) + "</td>"
                        s += "<td><button class=edit onclick=vala('STRING:" + str(row1[0]) + ":" +
str(
                            row2[0]) + ":" + data.split(":")[1].replace(" ", "_") + "')
>показать</button></td>"
                        s += "</tr>"
        diction.update({"add": s})
    return diction


def search_in_table_view(data):
    if data != None:
        diction = dict()
        s = ""
        tochnost = 0
        cursor = connection.cursor()
        cursor.execute('select * from ' + data.split(":")[1])
        col_names = [row[0] for row in cursor.description]
        for i in col_names:
            s += "<th>" + str(i) + "</th>"
        for row1 in cursor.execute('select * from ' + str(data.split(":")[1])):
            for i in range(0, len(row1)):
                if fuzz.WRatio(str(data.split(":")[2].replace("_", " ")), str(row1[i])) > 86 and len(
                        str(data.split(":")[2]).replace("_", " ")) <= len(str(row1[i])):
                    s += "<tr>"
```

```
                tochnost = 1
                for i in range(0, len(row1)):
                    s += "<td>" + str(row1[i]) + "</td>"
        if tochnost == 0:
            for row1 in cursor.execute('select * from ' + str(data.split(":")[1])):
                for i in range(0, len(row1)):
                    if fuzz.WRatio(str(data.split(":")[2].replace("_", " ")), str(row1[i])) > 85 and len(
                            str(data.split(":")[2]).replace("_", " ")) <= len(str(row1[i])):
                        s += "<tr>"
                        for i in range(0, len(row1)):
                            s += "<td>" + str(row1[i]) + "</td>"
        diction.update({"add": s})
        s = "<button class='btn btn-lpbuilder wave-effect' onclick=vala('VIEW:" +
str(data.split(":")[1]) + "')>Назад</button>"
        diction.update({"actionButton": s})
    return diction


def search_in_table(data):
    if data != None:
        diction = dict()
        s = ""
        tochnost = 0
        cursor = connection.cursor()
        cursor.execute('select * from ' + data.split(":")[1])
        col_names = [row[0] for row in cursor.description]
        for i in col_names:
            s += "<th>" + str(i) + "</th>"
        s += "<th>Действие</th>"
        for row1 in cursor.execute('select * from ' + str(data.split(":")[1])):
            for i in range(0, len(row1)):
                if fuzz.WRatio(str(data.split(":")[2].replace("_", " ")), str(row1[i])) > 86 and len(
                        str(data.split(":")[2]).replace("_", " ")) <= len(str(row1[i])):
                    s += "<tr>"
                    tochnost = 1
                    for i in range(0, len(row1)):
```

```
                s += "<td>" + str(row1[i]) + "</td>"
            s += "<td><button onclick=if(confirm('Удалить?')){vala('DELETE:" + str(
                data.split(":")[1].upper()) + ":" + str(row1[0]) + ":" + str(col_names[0]) + \
                 "');}else{event.stopPropagation();event.preventDefault();};
class='dell'>Удалить</button>" \
                 "<button class='edit' onclick=vala('EDITSTRING:" + str(data.split(":")[1]) + ":"
+ str(
                row1[0]) + ":back:" + str(data.split(":")[2].replace(" ", "_")) +
"')>редактировать</button></td></tr>"
    if tochnost == 0:
        for row1 in cursor.execute('select * from ' + str(data.split(":")[1])):
            for i in range(0, len(row1)):
                if fuzz.WRatio(str(data.split(":")[2].replace("_", " ")), str(row1[i])) > 85 and len(
                    str(data.split(":")[2]).replace("_", " ")) <= len(str(row1[i])):
                s += "<tr>"
                for i in range(0, len(row1)):
                    s += "<td>" + str(row1[i]) + "</td>"
                s += "<td><button onclick=if(confirm('Удалить?')){vala('DELETE:" + str(
                    data.split(":")[1].upper()) + ":" + str(row1[0]) + ":" + str(col_names[0]) + \
                     "');}else{event.stopPropagation();event.preventDefault();};
class='dell'>Удалить</button>" \
                     "<button class='edit' onclick=vala('EDITSTRING:" + str(data.split(":")[1]) +
":" + str(
                    row1[0]) + ":back:" + str(data.split(":")[2].replace(" ", "_")) +
"')>редактировать</button></td></tr>"
    diction.update({"add": s})
    s = "<button class='btn btn-lpbuilder wave-effect' onclick=vala('EDIT_VIEW:" +
str(data.split(":")[1]) + "')>Назад</button>"
    diction.update({"actionButton": s})
  return diction

def search_all_table(data):
  if data != None:
    cursor = connection.cursor()
    cursor1 = connection.cursor()
```

```
        diction = dict()
        s = "<tr><th colspan=3>найдены совпадения в следующих справочниках</th></tr>"
        for row1 in cursor.execute('SELECT table_name FROM user_tables'):
            o = 1
            for row2 in cursor1.execute('select * from ' + str(row1[0])):
                for i in range(0, len(row2)):
                    if fuzz.WRatio(str(data.split(":")[1].replace("_", " ")), str(row2[i])) > 86:
                        if o == 1:
                            s += "<tr><th colspan=2 >название справочника: " + row1[
                                0] + "</th><th><button class=edit onclick=vala('EDIT_VIEW:" +
str(row1[0]) + ":" + \
                                 data.split(":")[1].replace(" ", "_") + "') >открыть</button></th></tr>"
                            o = 0
                        s += "<tr>"
                        s += "<td colspan=2>" + str(row2[i]) + "</td>"
                        s += "<td><button class=edit onclick=vala('EDITSTRING:" + str(row1[0]) + ":"
+ str(row2[0]) + ":" + \
                             data.split(":")[1].replace(" ", "_") + "') >показать</button></td>"
                        s += "</tr>"
        diction.update({"add": s})
    return diction


def edit_view(data):
    if data != None:
        cursor = connection.cursor()
        diction = dict()
        cursor.execute('select * from ' + data.split(":")[1])
        col_names = [row[0] for row in cursor.description]
        size = len(col_names)
        s = ""
        for i in col_names:
            s += "<th>" + i + "</th>"
        s += "<th>Действие</th>"
        diction.update({"name0": s})
        if data.split(":")[1] == "SPRAV_NAME":
```

```
            dablic = 0
            s = ""
            for row in cursor.execute('select * from ' + data.split(":")[1]):
                s += "<tr>"
                for i in range(0, size):
                    if dablic == 0:
                        s += "<td><a onclick=vala('EDIT_VIEW:" + str(row[i]) + "')>" + str(row[i]) +
"</a></td>"
                        dablic = 1
                    else:
                        s += "<td>" + str(row[i]) + "</td>"
                        s += "<td><button class='edit' onclick=vala('EDIT_VIEW:" + str(row[0]) +
"')>открыть</button></td>"
                        dablic = 0
                s += "</tr>"
            diction.update({"add": s})


        else:
            s = "<tr>"
            for i in range(0, size, 1):
                s += "<td><input id=name" + str(i) + " placeholder='значение поля' type='text'
class='input'></input></td>"
            s += "<td><button onclick=vala3('" + str(data.split(":")[1].upper()) + "'),"
            for i in range(0, size, 1):
                s += "vala3(document.getElementById('name" + str(i) + "').value),"
            for i in col_names:
                s += "vala3('" + i + "'),"
            s += "vala4() class = 'add' >добавить</button></td></tr>"
            diction.update({"add": s})
            s = ""
            for row in cursor.execute('select * from ' + data.split(":")[1]):
                s += "<tr>"
                for i in range(0, size):
                    s += "<td>" + str(row[i]) + "</td>"
```

```
            s += "<td><button onclick=if(confirm('Удалить?')){vala('DELETE:" + str(data.split(":")[1].upper()) + ":" + str(row[0]) + ":" + str(col_names[0]) + \
                "');}else{event.stopPropagation();event.preventDefault();};
class='dell'>Удалить</button>" \
                "<button class='edit' onclick=vala('EDITSTRING:" + str(data.split(":")[1]) + ":" + str(row[0]) + ":back')>редактировать</button></td></tr>"
            diction.update({"add1": s})
            if len(data.split(":")) == 2:
                s = "<div align=150px><button onclick=vala('HELP1') class='btn btn-lpbuilder wave-effect'>Назад</button></div>"
            else:
                s = "<div align=150px><button onclick=vala5('" + data.split(":")[2] + "') class='btn btn-lpbuilder wave-effect'>Назад</button></div>"
            s += "<input id=search class=inputNT placeholder='поиск по этой таблице' type='text'></input><button onclick=vala('searchintable:'+'" + str(data.split(":")[1]) + ":'+document.getElementById('search').value) class='upbutton'>Поиск</button>"
            diction.update({"actionButton": s})

        return diction

def addString(data):
    if data != None:
        tableName = str(data.split(":")[1])
        dataPoly = list()
        namePoly = list()
        size = len(data.split(":"))
        size = int(((size-3)/2)+2)
        for i in range(2, size):
            dataPoly.append(str(data.split(":")[i]))
        for i in range(size, size+size-2):
            namePoly.append(str(data.split(":")[i]))
        s = "INSERT INTO " + str(tableName) + " ("
        for i in range(0, size-2):
            if i == size-3:
                s += str(namePoly[i]) + ") "
```

```
            else:
                s += str(namePoly[i]) + ","
        s += "VALUES ("
        for i in range(0, size-2):
            if i == size-3:
                s += "'" + str(dataPoly[i]) + "');"
            else:
                s += "'" + str(dataPoly[i]) + "',"
        cursor = connection.cursor()
        cursor.execute(str(s))
        send = edit_view(":" + tableName)
    return send


def dell(data):
    if data != None:
        nameTable = str(data.split(":")[1])
        dateTable = data.split(":")[2]
        nameColumn = data.split(":")[3]
        s = ""
        if nameTable == "SPRAV_NAME":
            cursor = connection.cursor()
            s += "DELETE FROM " + nameTable + " WHERE " + nameColumn + "='" + dateTable
+ "';"
            cursor.execute(str(s))
            s = "drop table " + dateTable[0] + ";"
            cursor.execute(str(s))
            send = edit_view("SPRAV_NAME")
        else:
            s += "DELETE FROM " + nameTable + " WHERE " + str(nameColumn) + "='" +
str(dateTable) + "';"
            cursor = connection.cursor()
            cursor.execute(str(s))
            send = edit_view(":" + nameTable)
    return send
```

```
def add(data):
    if data != None:
        tableNmae = str(data.split(":")[1]).upper()
        description = data.split(":")[2]
        name = list()
        type = list()
        size = list()
        check = list()

        for i in range(3, len(data.split(":"))-3, 4):
            name.append(data.split(":")[i])
        for i in range(4, len(data.split(":"))-3, 4):
            type.append(data.split(":")[i])
        for i in range(5, len(data.split(":"))-2, 4):
            size.append(data.split(":")[i])
        for i in range(6, len(data.split(":"))-1, 4):
            if str(data.split(":")[i]) == "true":
                check.append("NOT NULL")
            else:
                check.append("")
        s = "CREATE TABLE " + str(tableNmae) + " ( "
        for i in range(0, len(name)):
            if i == len(name)-1:
                s += str(name[i]) + " " + str(type[i]) + "(" + str(size[i]) + ") " + str(check[i])
            else:
                s += str(name[i]) + " " + str(type[i]) + "(" + str(size[i]) + ") " + str(check[i]) + ", "
        s += ");"
        cursor = connection.cursor()
        cursor.execute(str(s))
        # print(s)
        s = "INSERT INTO SPRAV_NAME (NAME, DESCRIPTION) VALUES ('" + tableNmae
+ "', '" + description + "');"
        # print(s)
        cursor.execute(str(s))
        send = edit_view(tableNmae)
```

```
        return send

def replace(data):
    if data != None:
        global auth
        print(auth)
        cursor = connection.cursor()
        diction = dict()
        cursor.execute('select * from ' + data.split(":")[1])
        col_names = [row[0] for row in cursor.description]
        size = len(col_names)
        namecolumn = ""
        for i in col_names:
            namecolumn += "<th>" + i + "</th>"
        if data.split(":")[1] == "SPRAV_NAME":
            namecolumn += "<th>Действие</th>"
            diction.update({"name0": namecolumn})
            dablic = 0
            s = ""
            for row in cursor.execute('select * from ' + data.split(":")[1]):
                s += "<tr>"
                for i in range(0, size):
                    if dablic == 0:
                        s += "<td><a onclick=vala('VIEW:" + str(row[i]) + "')>" + str(row[i]) +
"</a></td>"
                        dablic = 1
                    else:
                        s += "<td>" + str(row[i]) + "</td>"
                        s += "<td><button class='edit' onclick=vala('VIEW:" + str(row[0]) +
"')>открыть</button></td>"
                        dablic = 0
                s += "</tr>"
            diction.update({"add": s})

        else:
```

```
            diction.update({"name0": namecolumn})
            s = ""
            for row in cursor.execute('select * from ' + data.split(":")[1]):
                s += "<tr>"
                for i in range(0, size):
                    s += "<td>" + str(row[i]) + "</td>"
            diction.update({"add1": s})
            if len(data.split(":")) == 2:
                s = "<div align=150px><button align=150px onclick=vala('HELP') class='btn btn-lpbuilder wave-effect'>Назад</button></div>"
            else:
                s = "<div align=150px><button onclick=vala5('" + data.split(":")[2] + "') class='btn btn-lpbuilder wave-effect'>Назад</button><div>"
            s += "<input id=search class=inputNT placeholder='поиск по этой таблице' type='text'></input><button onclick=vala('searchintableview:'+'" + str(
                data.split(":")[1]) + ":'+document.getElementById('search').value) class='upbutton'>Поиск</button>"
            diction.update({"actionButton": s})

        return diction


def add_sprav(request):
    if request.method == "GET":
        return render(request, 'add_page.html')
    if request.method == "POST":
        if request.is_ajax:
            name = request.POST
            s = name.dict()
            name = s['add']
            a = str(name)
            data = replace(a)
            print(data)
            return JsonResponse(data)
```

```
def wiev_sprav(request):
    if request.method == "GET":
        return render(request, 'page2.html')
    if request.method == "POST":
        if request.is_ajax:
            name = request.POST
            s = name.dict()
            name = s['add']
            a = str(name)
            data = replace(a)
            print(data)
            return JsonResponse(data)

def start_page(request):
    if request.method == "GET":
        return render(request, 'page1.html')
    if request.method == "POST":
        if request.is_ajax:
            name = request.POST
            s = name.dict()
            name = s['add']
            a = str(name)
            data = replace(a)
            print(data)
            return JsonResponse(data)


def function1(request):
    data = {"massage": ""}
    if request.method == "GET":
        data["massage"] = "uspeh"
    return JsonResponse(data)
```