

Министерство просвещения РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Глазовский государственный инженерно–педагогический университет имени
В.Г. Короленко»

Факультет информатики, физики и математики

Кафедра математики и информатики

Среднее профессиональное образование

09.02.03 Программирование в компьютерных системах

Курсовой проект

МДК 03.01 Технология разработки программного обеспечения

РАЗРАБОТКА БАЗЫ ДАННЫХ ДЛЯ МОБИЛЬНОГО РОБОТА ДОСТАВЩИКА

Выполнил(а):
студент(ка) 4 курса, группа 43
очной формы обучения
Куртеева Евгения Алексеевна
(Ф. И. О. студента)

Руководитель:
Старший Преподаватель
Касаткин К.А.
(ученая степень, ученое
звание/должность, Ф.И.О.)

оценка

дата, подпись руководителя

Глазов, 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1. Анализ предметной области и проектирование базы данных для мобильного робота–доставщика	4
2. Разработка структуры и схемы базы данных	19
3. Разработка графического интерфейса.....	21
ЗАКЛЮЧЕНИЕ	25
СПИСОК ЛИТЕРАТУРЫ.....	26
Приложение 1	31
Приложение 2	37
Приложение 3	38

ВВЕДЕНИЕ

С развитием информационных технологий развивалась и сфера доставки. Сначала появились онлайн заказы, которые доставляют курьеры, а теперь заказы может доставлять и робот. Роботы поставщики предоставляют возможность повышения эффективности доставки, уменьшения затрат и сокращение времени на выполнение задач. Эффективное функционирование таких роботов требует хорошо организованной базы данных, которая хранит и обрабатывает поступающие заказы, а также хранит историю доставок.

В данной курсовой работе рассматривается процесс разработки базы данных для мобильного робота доставщика. При создании базы данных учитываются особенности функционала робота.

Цель: разработать базу данных для мобильного робота доставщика, обеспечивающую хранение, управление и быстрый доступ к данным для оптимизации процесса доставки.

Задачи:

1. Изучить СУБД и их возможности, достоинства и недостатки.
2. Разработать свою базу данных, создать связи.
3. Изучить возможности подключения и взаимодействия с базой данных используя сторонние средства. Изучить возможности создания графического интерфейса.
4. Создать графический интерфейс для работы с базой данных. Добавить возможность создавать SQL запросы.
5. Заняться администрированием базы данных. Добавить роли. Создать резервную копию.

1. Анализ предметной области и проектирование базы данных для мобильного робота–доставщика

Предметная область включает в себя различные области и компоненты, связанные с созданием и управлением базы данных для мобильного робота доставщика.

В данной предметной области можно выделить следующее:

База данных должна содержать информацию о заказах, их статусах, доставочных адресах и времени выполнения. Это позволит роботу эффективно выполнять доставку, а заказчику отслеживать статус заказа.

Также база данных должна содержать информацию о доступных товарах, их количестве, а также о месте их хранения, что позволит роботу определить наличие товара и место его хранения.

В базе должна содержаться информация о маршрутах доставки, а также ограничениях и правилах движения. Робот будет использовать эту информацию для планирования оптимального маршрута доставки [13].

База данных содержит информацию о клиентах, их контактных данных, истории заказов. Это позволит роботу более эффективно обслуживать клиентов и осуществлять персонализированную доставку.

База данных может использоваться для сбора и анализа данных о производительности робота, эффективности маршрутов доставки, уровне удовлетворенности клиентов и других показателях. Это позволит компании получить ценную информацию о работе робота и оптимизировать процессы доставки [24].

Основной целью разработки базы данных для мобильного робота–доставщика является обеспечение эффективной и надежной работы робота, оптимизация процесса доставки и повышение уровня обслуживания.

Разработка базы данных для мобильного робота–доставщика включает в себя создание структуры базы данных и определение сущностей, атрибутов и связей, необходимых для эффективного управления и отслеживания доставки.

Сущности, которые могут быть включены в базу данных:

Сущность робота содержит информацию о каждом мобильном роботе–доставщике, такую как id робота, модель, статус, местоположение.

Заказы представляют собой информацию о запросах на доставку от клиентов. Она включает в себя номер заказа, дату и время заказа, дата и время получения заказа, адрес отправителя и получателя, список товаров, статус заказа.

По сущности клиенты можно отслеживать информацию о каждом клиенте, такую как id клиента, ФИО, контактные данные, историю заказов, оценку доставки.

В товарах содержатся данные о каждом товаре, такие как его название, наличие, стоимость и описание, id товара, количество.

Сущность история доставок содержит информацию о каждой доставке, включая номер заказа, время доставки, id робота, отметку о доставке, состояние доставленного заказа.

Для связей между сущностями можно использовать следующие связи: Один к одному (Например, между заказами и клиентами. У одного клиента может быть только один заказ за раз).

Один ко многим (например, у одного клиента может быть несколько заказов) [5].

Некоторые популярные инструменты для разработки базы данных для мобильного робота–доставщика:

1. **SQLite** – легкая и встраиваемая СУБД, которая обеспечивает высокую производительность и простоту использования для мобильных приложений.

Достоинства SQLite:

Высокая скорость: благодаря особенностям архитектуры SQLite работает быстро, особенно на чтение. Компоненты СУБД встроены в приложение и вызываются в том же процессе. Поэтому доступ к ним быстрее, чем при взаимодействии между разными процессами.

Хранение данных в одном файле: база данных состоит из табличных записей, связей между ними, индексов и других компонентов. В SQLite они хранятся в едином файле (database file), который находится на том же устройстве, что и программа. Чтобы при работе не возникало ошибок, файл блокируется для сторонних процессов перед записью. Раньше это приводило к тому, что записывать данные в базу мог только один процесс одновременно. Но в новых версиях это решается перенастройкой режима работы СУБД.

Минимализм: создатели SQLite пользуются принципом «минимального полного набора». Из всех возможностей SQL в ней есть наиболее нужные. Поэтому SQLite отличают малый размер, простота решений и легкость администрирования. Для повышения базовой функциональности можно использовать стороннее программное обеспечение и расширения.

Надежность: код на 100% покрыт тестами. Это означает, что протестирован каждый компонент ПО. Поэтому SQLite считается надежной СУБД с минимальным риском непредсказуемого поведения.

Нулевая конфигурация: перед использованием СУБД не нужна сложная настройка или длительная установка. Для решения большинства задач ей можно пользоваться «из коробки», без установки дополнительных компонентов.

Малый размер: полностью сконфигурированный SQLite со всеми настройками занимает меньше 400 Кб. Если использовать СУБД без дополнительных компонентов, размер можно уменьшить до 250 Кб. Он зависит только от количества загруженной информации. Несмотря на малый размер, SQLite поддерживает большинство функций стандарта SQL2 и имеет ряд собственных.

Доступность: SQLite находится в публичном доступе. На ее использование нет правовых ограничений, а владельцем считается общество. Можно открывать, просматривать и изменять исходный код установленного ПО.

Кроссплатформенность: СУБД подходит для UNIX-подобных систем, MacOS и Windows.

Автономность: система независима от стороннего ПО, библиотек или фреймворков. Чтобы приложение с базой на SQLite работало, дополнительные компоненты не требуются. Также не обязателен доступ в интернет: вся база хранится на устройстве, получить данные можно локально.

Недостатки SQLite:

Ограниченная поддержка типов данных. SQLite поддерживает только четыре типа данных, которые реализованы в SQL:

INTEGER – целое число;

REAL – дробное число;

TEXT – текст;

BLOB – двоичные данные.

Также существует особое значение NULL – отсутствие данных.

Отсутствие хранимых процедур: так называются блоки кода на SQL, которые сохраняются в базу данных. Хранимые процедуры можно вызывать как отдельные функции, и это удобно, если нужно последовательно выполнить несколько однотипных действий. Но SQLite их не поддерживает из-за особенностей архитектуры.

Ограничения в применении: отсутствие сервера – преимущество и недостаток одновременно. Без сервера возможности СУБД меньше. Например, к одной базе не смогут обращаться несколько разных устройств.

В SQLite ограничена многопоточность – единовременное выполнение нескольких процессов. Одновременно читать из базы могут несколько процессов, а писать в нее по умолчанию – только один. В версии 3.7.0 в SQLite внедрили возможность записи разными приложениями, но даже так она

уступает клиент–серверным СУБД по возможностям работы с потоками. Поэтому SQLite не подойдет для многопользовательских приложений или программ, записывающих большой объем данных.

Отсутствие бесплатной техподдержки: стоимость профессиональной технической поддержки от разработчиков – от \$1500 в год.

Отсутствие встроенной поддержки Unicode. Unicode – это популярный стандарт кодирования символов. Он включает практически все существующие знаки и буквы, поэтому считается самым распространенным в мире. Без его поддержки приложение не сможет корректно работать с кириллицей, иероглифами и многими другими символами. SQLite «из коробки» не поддерживает Unicode, его нужно настраивать отдельно. Это может вызвать сложности с локализацией [39].

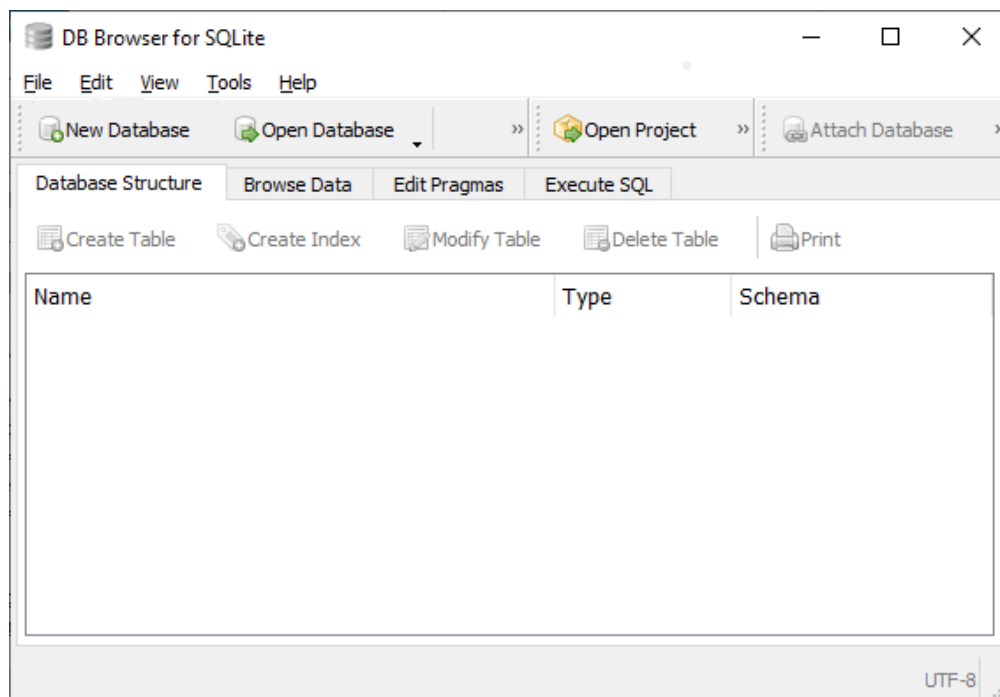


Рисунок 1 – Интерфейс SQLite

2. **Oracle Database** – мощная СУБД, предлагающая широкий спектр функций и возможностей для разработки баз данных (Рисунок 2).

Достоинства Oracle Database:

Переносимость: Oracle Database можно экспортировать на разные

платформы – что отличает её от конкурентов. Она может работать на почти 20 протоколах сетевого взаимодействия и более чем на 100 аппаратных платформах. Это дает возможность без проблем разрабатывать приложения для Oracle, не заботясь о изменениях в операционной системе и оборудовании.

Резервное копирование и восстановление: Oracle Database идеально подходит для создания надёжных резервных копий базы данных и восстановления данных. Благодаря Oracle Database можно легко восстановить базу данных к нужному моменту времени. Для этого понадобятся дополнительное место для хранения и архивационные механизмы.

Высокая производительность: Oracle Database обеспечивает высокую скорость работы и способна обрабатывать большие базы данных. Она увеличивает производительность и скорость обработки транзакций за счёт контроля и блокировки.

Поддержка нескольких баз данных: одно из лучших преимуществ Oracle Database – возможность управления несколькими базами данных в рамках одной транзакции. Это наиболее успешно реализовано в версии V7.

Присутствие на рынке: что касается доли рынка СУБД, Oracle имеет наибольшую долю в сегментах серверов VMS, UNIX и OS/2. Это говорит о том, что шансы остаться без поддержки Oracle минимальны, ведь поддерживаются множество сторонних интерфейсов. Более того, вы можете получить дополнительную помощь, так как опытный персонал легко доступен.

Обновление версий: Oracle своевременно информирует вас о предстоящем большом релизе и потенциальных изменениях, чтобы вы могли подготовиться. Oracle предлагает хорошую обратную совместимость, так что вам не придется переписывать приложение при обновлении СУБД. Многие работают с Oracle начиная с версии V4 Beta и никогда не сталкивались с проблемами синтаксиса.

Недостатки Oracle Database:

Сложность: одним из крупных недостатков Oracle Database является ее сложность. Oracle не рекомендуется для использования без достаточных

технических знаний и навыков работы с Oracle Database. Oracle также не подойдет для тех, кто ищет простую в использовании и основных функциях базу данных. Установка Oracle и начало работы с ним требуют специализированных навыков, ведь это чрезвычайно сложный движок.

Стоимость Oracle Database: Oracle может быть в десять раз дороже, чем среднерыночное решение MS SQL Server Database. Поэтому многие предпочитают более доступные варианты, например, установить MySQL бесплатно или использовать один из движков в решении, например, AWS, инвестируя минимальные средства. К тому же, стоимость лицензии Oracle может изменяться со временем в зависимости от различных факторов, таких как изменения в политике компании, патчи и обновления.

Сложность управления: Oracle обычно требует больше усилий для управления некоторыми операциями. Совет: начните с установки базовой версии и минимальной настройки. Oracle Database оправдана только при работе с большими базами данных. Для малых и средних компаний, где требуются небольшие базы данных, Oracle не рекомендуется. В таком случае лучшим вариантом будет MySQL, который является экономически выгодным [37].

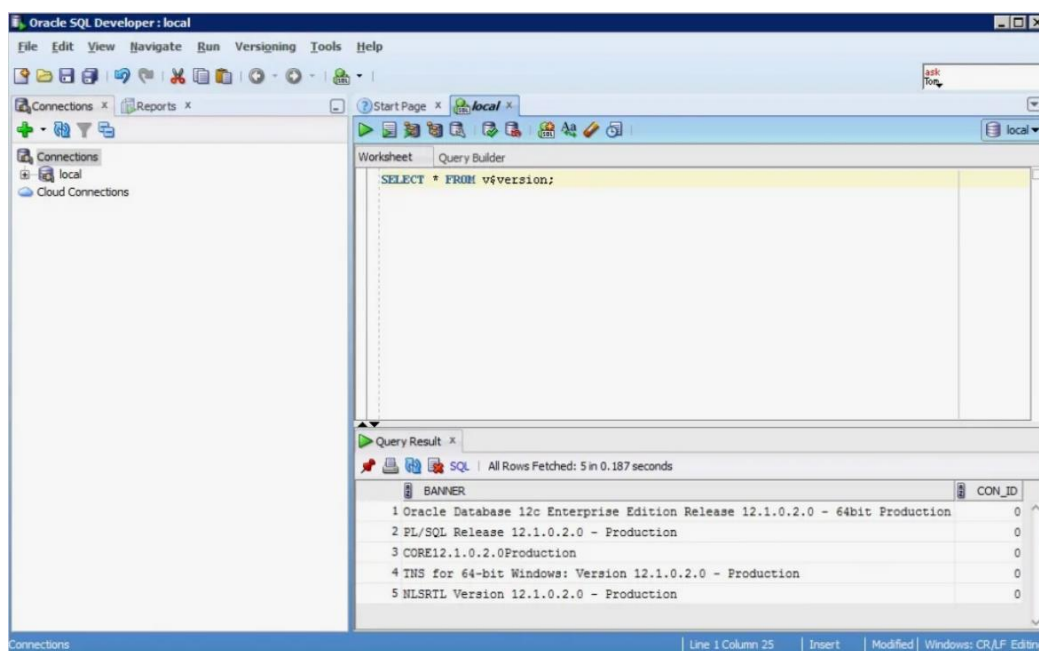


Рисунок 2 – Интерфейс Oracle

3. **Access** – это реляционная СУБД, представляющая собой совокупность связанных между собой реляционных таблиц.

Достоинства Oracle Database:

Его очень легко установить и использовать: установка этого Microsoft Access займет всего несколько минут. Взамен пользователи получают полнофункциональную базу данных. Кроме того, для использования не требуется никакого сложного программирования, поэтому средний пользователь компьютера может воспользоваться информацией, которую может собирать эта система.

Также легко интегрировать: практически все, что основано на Windows, может быть интегрировано с Microsoft Access. Сюда входят SQL, Sybase и Oracle для интерфейсных или внутренних таблиц. Кроме того, ее легче обслуживать, чем другие системы, которые могут предлагать более крупные приложения для работы с базами данных.

Предлагает большую емкость для хранения: вы можете хранить несколько ГБ данных с помощью Microsoft Access. Он также доступен многим пользователям в одном приложении Access. Некоторые пользователи сообщают, что к приложению Access одновременно обращаются 10 пользователей в сети [30].

Импортировать данные просто: вы можете импортировать все собранные данные в Microsoft Access примерно за то же время, которое требуется для первоначальной установки. Существует также простое решение для резервного копирования ваших данных, поскольку Access хранит все в одном месте.

Стоимость – реальное преимущество: Microsoft Access не только зачастую более доступен по цене, чем более крупные системы баз данных, он также может быть практически бесплатным для некоторых предприятий. Поскольку он включен в профессиональный набор продуктов Office, многие предприятия малого бизнеса уже имеют его и в настоящее время просто не используют. Рентабельность также распространяется на плату за консультации, когда могут возникнуть проблемы с программой.

Может быть размещен на веб-сайте для удаленного доступа пользователей. Несмотря на то, что у вас по-прежнему будут ограничения одновременного подключения, когда у вас будет доступ удаленного пользователя, вы сможете обеспечить полный контроль и функциональность, чтобы сделать удаленную работу жизнеспособным вариантом для администраторов баз данных.

Недостатки Access:

Это конечная система баз данных: когда дело доходит до ввода информации в эту базу данных, в конечном итоге может быть достигнута стена. Для пользователей с большими данными Microsoft Access может вызвать ограничения, которые могут отсутствовать в других программах баз данных, просто потому, что существует ограничение на размер файла.

Все данные сохраняются в одном файле: для файлов, которые закрыты до максимального размера, разрешенного этой программой базы данных, единый формат файла снижает производительность программы. Создание и отправка отчетов может занять несколько минут. Запросы и формы могут даже вызвать сбой на некоторых компьютерах.

Мультимедийные данные сложно включить в Microsoft Access: это связано с проблемой сохранения файла. Мультимедийные данные обычно занимают много места, а это означает, что всего пара файлов может снизить производительность базы данных.

Операции, критичные ко времени, трудно перехватить в Microsoft Access: если вы генерируете данные, которые необходимо использовать или распространять немедленно, эта база данных может вам не подойти. Думайте о Access как о долгосрочной базе данных для интеллектуального анализа данных и оценки показателей.

Могут быть проблемы с безопасностью: если ваша база данных должна быть защищена расширенными протоколами безопасности базы данных, тогда Access может быть не лучшим вариантом. Вы можете избежать этой проблемы, используя Access на внешнем интерфейсе SQL-сервера, но не всем компаниям

нужен такой уровень инфраструктуры, и они по-прежнему создают конфиденциальные данные для ее защиты.

Различные операционные системы Windows могут повредить вашу базу данных. У вас может быть несколько различных операционных систем Windows, используемых в компании, от Windows 98 до текущей Windows 10. Если вы делитесь базой данных Access со смешанными пользователями операционных систем, вы можете испортить собранную информацию.

Еще предстоит научиться: Microsoft Access имеет относительно плохую реляционную структуру, что может затруднить обучение некоторых пользователей использованию форм базы данных или доступ к ним. Элементы управления и формы также могут быть трудными для настройки или адаптации к определенным потребностям, и, как правило, необходимо иметь базовое понимание языка программирования SQL, чтобы иметь наибольшие шансы на успех [1].

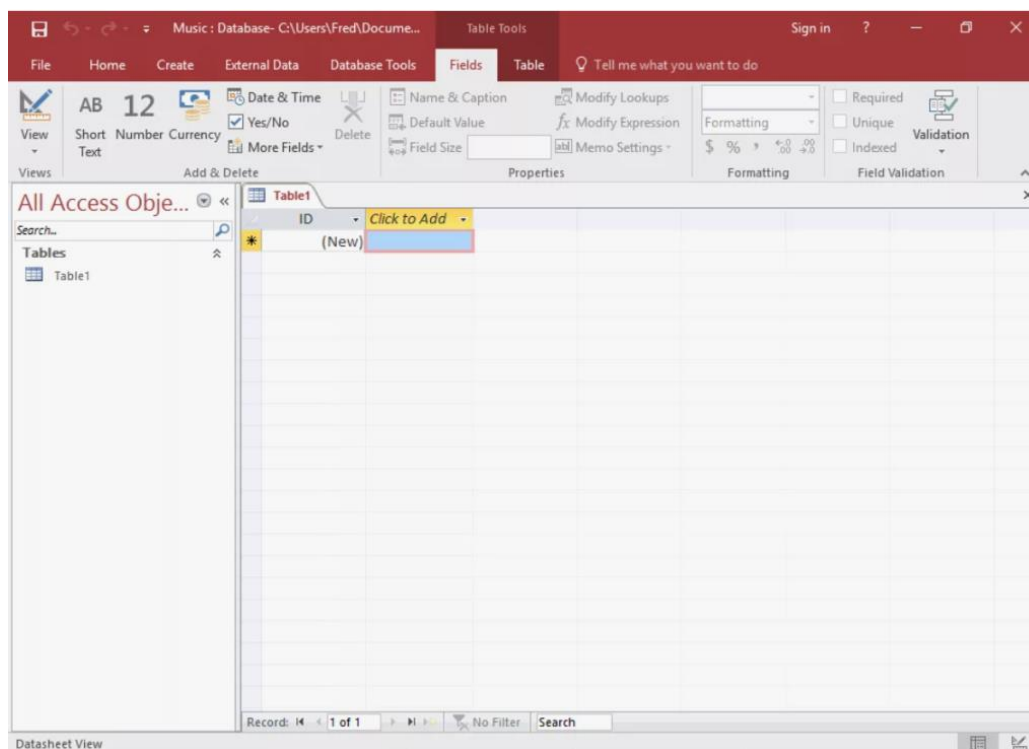


Рисунок 3 – Интерфейс Microsoft Access

После сравнения нескольких СУБД мой выбор пал на PostgreSQL.

PostgreSQL – еще одна популярная СУБД, которая предлагает расширенные возможности для хранения и обработки данных.

Достоинства PostgreSQL:

Поддержка множества типов данных: Еще одна особенность PostgreSQL – поддержка большого количества типов записи информации. Это не только стандартные целочисленные значения, числа с плавающей точкой, строки и булевы значения («да/нет»), но и денежный, геометрический, перечисляемый, бинарный и другие типы. PostgreSQL «из коробки» поддерживает битовые строки и сетевые адреса, массивы данных, в том числе многомерные, композитные типы и другие сложные структуры. В ней есть поддержка XML, JSON и NoSQL-баз.

При необходимости к СУБД можно подключить поддержку типов данных, которые нужны в конкретном проекте. В PostgreSQL есть несколько внутренних форматов, которые используются только в ней.

Работа с большими объемами: в большинстве СУБД, рассчитанных на средние и небольшие проекты, есть ограничения по объему базы и количеству записей в ней. В PostgreSQL ограничений нет.

Ограничения касаются только конкретных записей: одна таблица может занимать не больше 32 Тб, а одна запись – 1,6 Тб. В одном поле записи может быть не больше 1 Гб данных, а максимальное количество полей зависит от типа и составляет от 250 до 1600 штук. Максимальных значений хватает, чтобы хранить в БД любые данные.

Поддержка сложных запросов: PostgreSQL работает со сложными, составными запросами. Система справляется с задачами разбора и выполнения трудоемких операций, которые подразумевают и чтение, и запись, и вариацию одновременно. Она медленнее аналогов, если речь заходит только о чтении, но в других аспектах превосходит конкурентов.

Написание функций на нескольких языках: В PostgreSQL можно писать собственные функции – пользовательские блоки кода, которые выполняют те

или иные действия. Эта возможность есть практически в любых СУБД, но PostgreSQL поддерживает больше языков, чем аналоги. Кроме стандартного SQL, в PostgreSQL можно писать на C и C++, Java, Python, PHP, Lua и Ruby. Он поддерживает V8 – один из движков JavaScript, поэтому JS тоже можно использовать совместно с PostgreSQL. Реализована поддержка Delphi, Lisp и прочих редких языков. При необходимости можно расширить систему под другие ЯП.

Модификация SQL, которая используется в PostgreSQL, называется PL/pgSQL. Это процедурное расширение, которое поддерживает сложные вычисления и дополняет «классический» SQL новыми возможностями.

Одновременная модификация базы: важная особенность PostgreSQL – возможность одновременного доступа к базе с нескольких устройств. В СУБД реализована клиент–серверная архитектура, когда база данных хранится на сервере, а доступ к ней осуществляется с клиентских компьютеров. Так, например, реализуются разнообразные сайты. Одна из возможных сложностей – ситуация, когда несколько человек одновременно модифицируют базу и нужно избежать конфликтов.

Соответствие ACID: ACID – это набор принципов для обеспечения целостности данных. Аббревиатура расшифровывается как Atomicity, Consistency, Isolation, Durability – атомарность, согласованность, изолированность, прочность. Если база данных соответствует этим принципам, она ведет себя максимально предсказуемо и надежно. В ней низок риск конфликта или непредвиденного поведения системы. PostgreSQL соблюдает требования ACID благодаря технологии MVCC. Это делает систему надежной и безопасной в использовании, а данные – защищенными от возможных сбоев, ошибок и потерь.

Возможность расширения: разработчик может написать для СУБД собственные типы и их преобразования, операции и функции, ограничения и индексы, собственный процедурный язык для запросов. PostgreSQL можно модифицировать практически под любую нестандартную задачу.

Высокая мощность и широкая функциональность: PostgreSQL – возможно, единственная бесплатная СУБД с открытым исходным кодом, которая рассчитана на работу с объемными и сложными проектами. Она мощная, производительная, способна эффективно работать с большими массивами данных. Есть примеры реального использования СУБД для баз данных в несколько петабайт с сотнями тысяч запросов в секунду. На главной странице официального сайта PostgreSQL называют «самой продвинутой бесплатной СУБД». Система действительно имеет высокую функциональность и не уступает платным продуктам.

Открытость: PostgreSQL – ПО с открытым исходным кодом, которое распространяется по свободной лицензии. Это означает, что любой разработчик может посмотреть, как написана система, или предложить для нее свои правки. СУБД разрабатывается сообществом энтузиастов и в определенной степени никому не принадлежит, а значит, ее можно свободно и без ограничений использовать в своих проектах.

На базе PostgreSQL существуют коммерческие продукты с платным доступом – ими обычно пользуются крупные компании, которым нужна дополнительная функциональность. Это, например, связь с Oracle Database или продвинутый веб–интерфейс для администрирования БД.

Минимальное количество багов: PostgreSQL – проект, который известен высоким качеством отладки. Каждая версия системы появляется в доступе только после полной проверки, поэтому СУБД очень стабильна. Частая проблема бесплатных проектов – новые версии с большим количеством багов, но в случае с PostgreSQL такой проблемы нет. Согласно независимым автоматизированным исследованиям, в исходном коде СУБД есть одна ошибка на 39 000 строк кода. Это в пять раз меньше, чем в MySQL, и в пятьдесят раз меньше, чем в ядре операционной системы Linux.

Кроссплатформенность: чаще всего PostgreSQL используют на серверах с операционными системами семейства Linux, но СУБД поддерживает и другие ОС. Ее можно установить в системы на базе Windows, BSD, macOS и Solaris.

Кроме того, у PostgreSQL есть автономный веб-сервер PostgREST, с которым можно работать с помощью REST API. СУБД можно развернуть и в облаке.

Недостатки PostgreSQL:

Сложности при настройке: у PostgreSQL очень обширный набор возможностей. Очевидно, что такое разнообразие функций влечёт за собой множество настроек, что может вызвать затруднения у новичков. Корректная настройка базы данных требует глубокого понимания архитектуры и параметров.

Повышенное потребление ресурсов: в сравнении с некоторыми другими СУБД, PostgreSQL может потреблять больше ресурсов (включая оперативную память и процессорное время). Это особенно заметно при работе с большими объёмами данных и выполнении сложных запросов.

Отсутствие некоторых функций: по сравнению с определенными коммерческими аналогами PostgreSQL может немного уступать в функциональности [38].

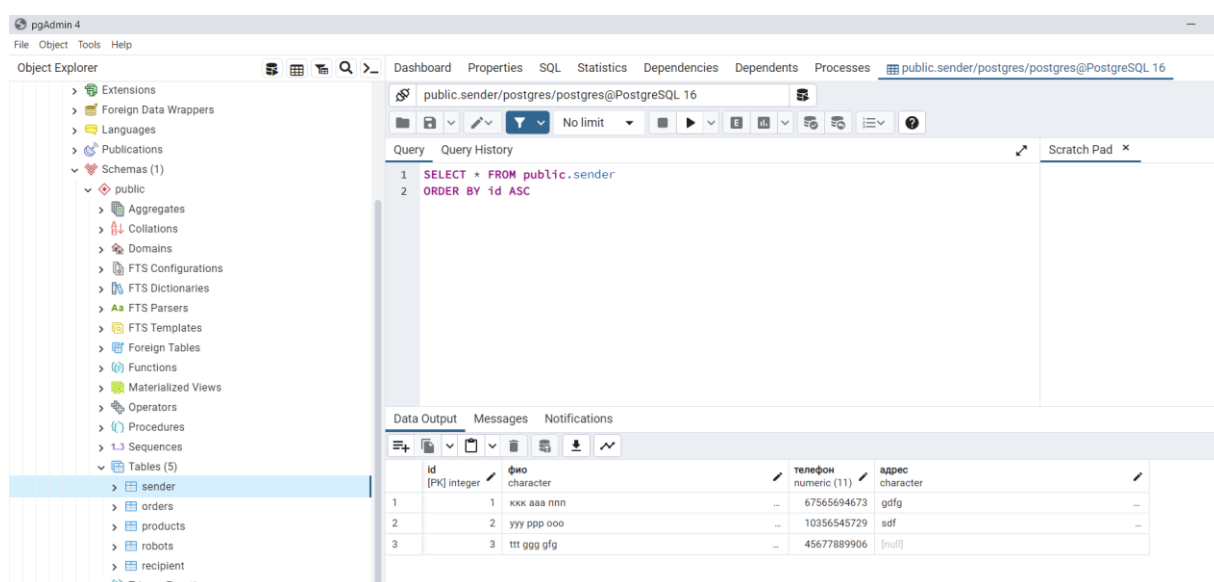


Рисунок 4 – Интерфейс PostgreSQL

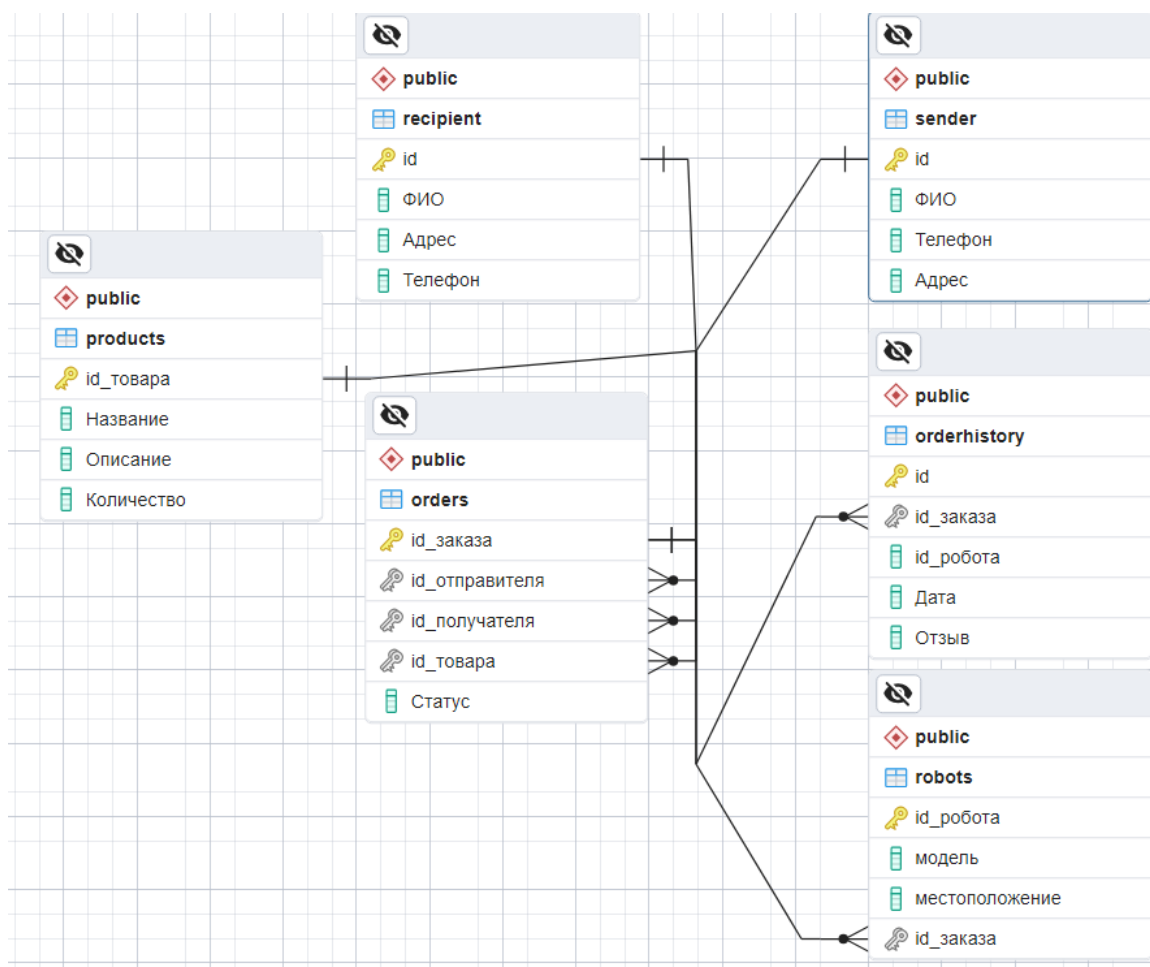


Рисунок 5 – Концептуальная модель базы данных

Концептуальная модель данных, представленная на рисунке 5, включает в себя все основные сущности и связи. Она не содержит подробных сведений об атрибутах и часто используется на начальном этапе планирования [12].

2. Разработка структуры и схемы базы данных

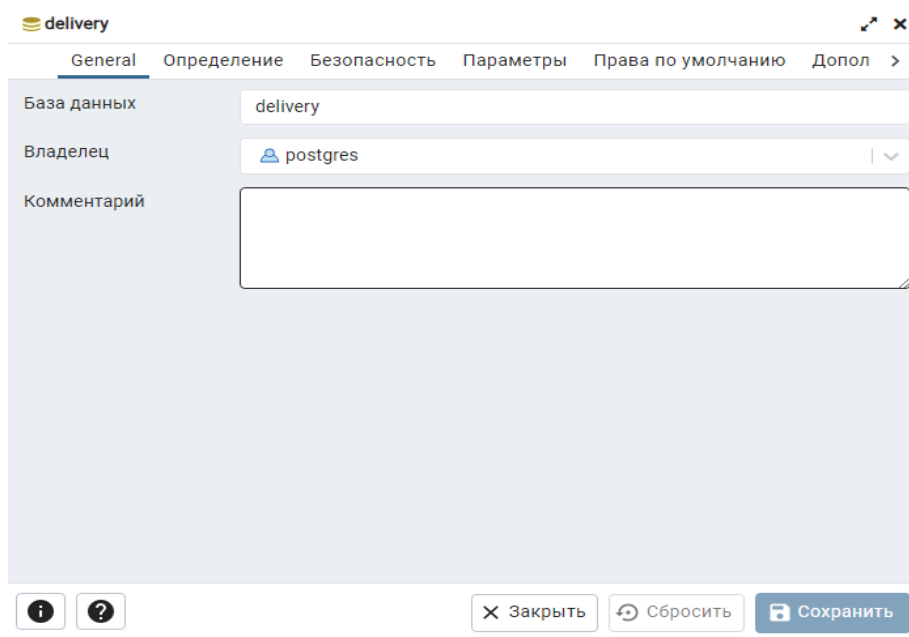


Рисунок 6 – Создание базы данных

На рисунке 6 показано создание базы данных. При её создании необходимо написать её название, а так же определить её владельца.

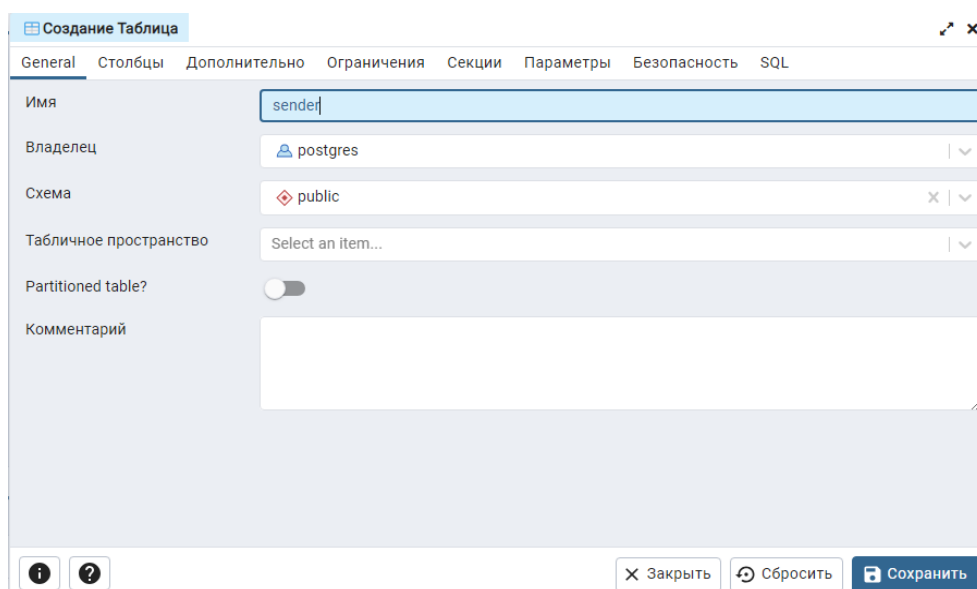


Рисунок 7 – Создание таблицы

При создании таблицы так же задаётся её имя (Рисунок 7).

sender

General **Столбцы** Дополнительно Ограничения Параметры Безопасность SQL

Наследуется из таблиц(ы)

Выберите источник наследования...

Столбцы

	Имя	Тип данных	Length/Precision	Масш...	Не NULL?	Первичный ...	По умолча...
	id	integer			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
	ФИО	character	100		<input type="checkbox"/>	<input type="checkbox"/>	
	Телефон	numeric	11	0	<input type="checkbox"/>	<input type="checkbox"/>	
	Адрес	character	100		<input type="checkbox"/>	<input type="checkbox"/>	

✕ Закрыть
↺ Сбросить
💾 Сохранить

Рисунок 8 – Добавление столбцов в таблицу

Так же необходимо заполнить вкладку столбцы показанную на рисунке 8. Добавить имена столбцов, их тип. У некоторых типов необходимо ввести максимальную длину вводимых символов. Для ключевого поля необходимо установить первичный ключ и Не NULL (данное поле обязательно должно быть заполнено) (Рисунок 8).

	id [PK] integer	ФИО character	Телефон numeric (11)	Адрес character
1	1	Третьякова Екатерина Леонидовна	89513754980	Корпус1, кабинет 10
2	2	Сорокина Татьяна Леонидовна	89509742806	Корпус1, кабинет 10
3	3	Белов Егор Матвеевич	89043686634	Корпус1, кабинет 10
4	4	Цветкова Александра Никитовна	89043654980	Корпус1, кабинет 10
5	5	Лазарева Дарина Викторовна	89503409678	Корпус1, кабинет 10

Рисунок 9 – Заполнение таблицы

При добавлении записи id должно быть уникально для каждой из них. Если id совпадут то запись не сохраниться. Также если после ввода данных тип или длина не совпадут с указанным то данная запись не сохраниться (Рисунок 9).

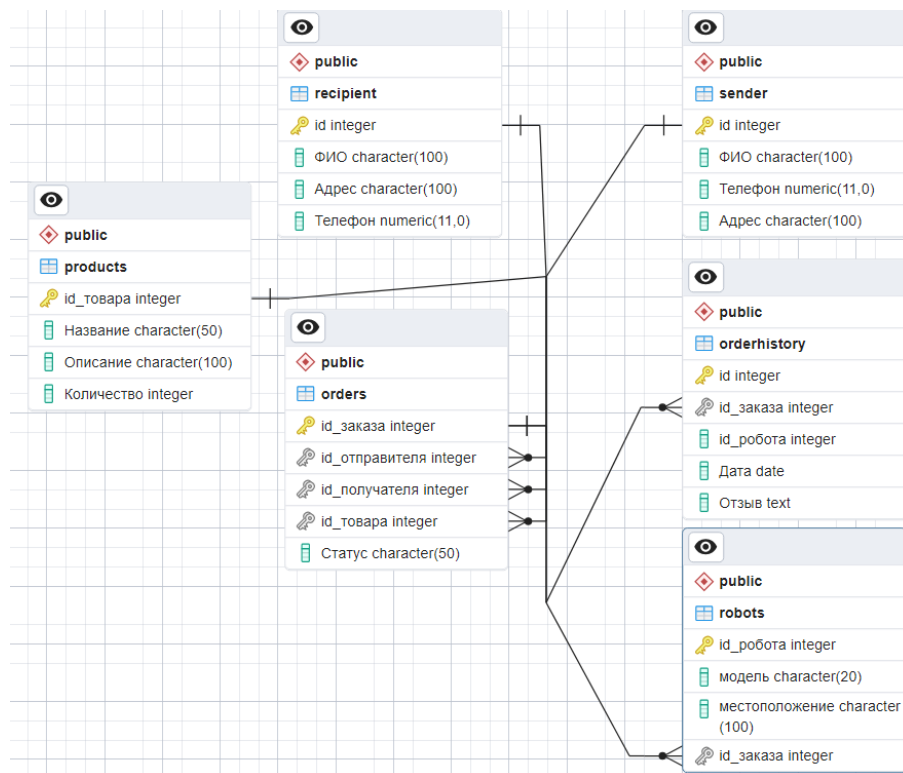


Рисунок 10 – Физическая модель базы данных

На рисунке 10 физическая модель данных, которая включает в себя все необходимые таблицы, столбцы, связи, свойства базы данных для физической реализации баз данных [12]. Руководство программиста приведено в приложении 2.

3. Разработка графического интерфейса

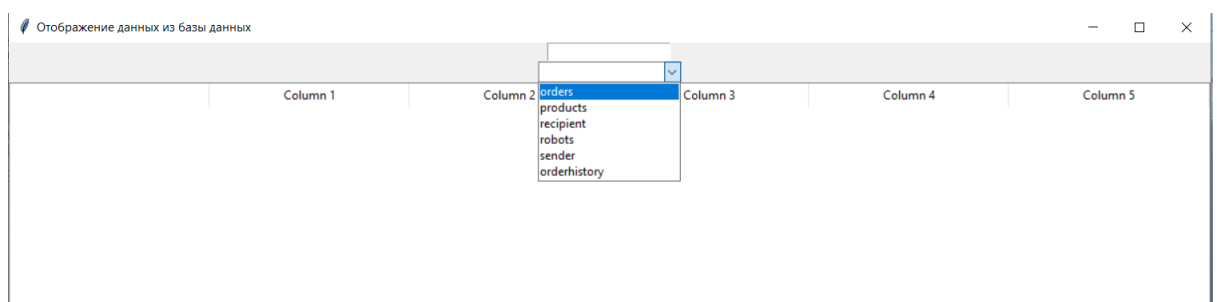


Рисунок 11 – Выбор таблицы

На рисунке 11 показан выбор таблицы из которой будут выгружены данные.

The screenshot shows a window titled "Отображение данных из базы данных". A dropdown menu is set to "robots". The table below shows the data for this table. A small form titled "Д..." is overlaid on the table, allowing for the addition of a new record. The form has four input fields: "Значение 1" (6), "Значение 2" (R2), "Значение 3" (Корпус3, кабинет11), and "Значение 4" (4). A "Сохранить" button is at the bottom of the form. Below the table, there are buttons for "Редактировать запись", "Удалить запись", and "Добавить запись".

id	модель	местоположение	id_заказа
2	R2	Корпус1, кабинет 10	5
3	R1.5	Корпус2, кабинет 10	2
4	R3	Корпус1, кабинет 12	1
5	R1	Корпус2, кабинет 10	4
1	R1	Корпус1, кабинет 15	3

Рисунок 12 – Добавления записи

На рисунке 12 показано как добавлять записи в выбранную таблицу.

The screenshot shows the same "Отображение данных из базы данных" window. The table data is the same as in Figure 12. A confirmation dialog titled "Успех" is displayed in the center, with the message "Запись успешно удалена." and an "OK" button. Below the table, the buttons "Удалить запись", "Поиск и сортировка", and "Сортировать по имени" are visible.

id	модель	местоположение	id_заказа
2	R2	Корпус1, кабинет 10	5
3	R1.5	Корпус2, кабинет 10	2
4	R3	Корпус1, кабинет 12	1
5	R1	Корпус2, кабинет 10	4
1	R1	Корпус1, кабинет 14	3

Рисунок 13 – Удаление записи

На рисунке 13 показано как удалить выбранную запись из данной таблицы.

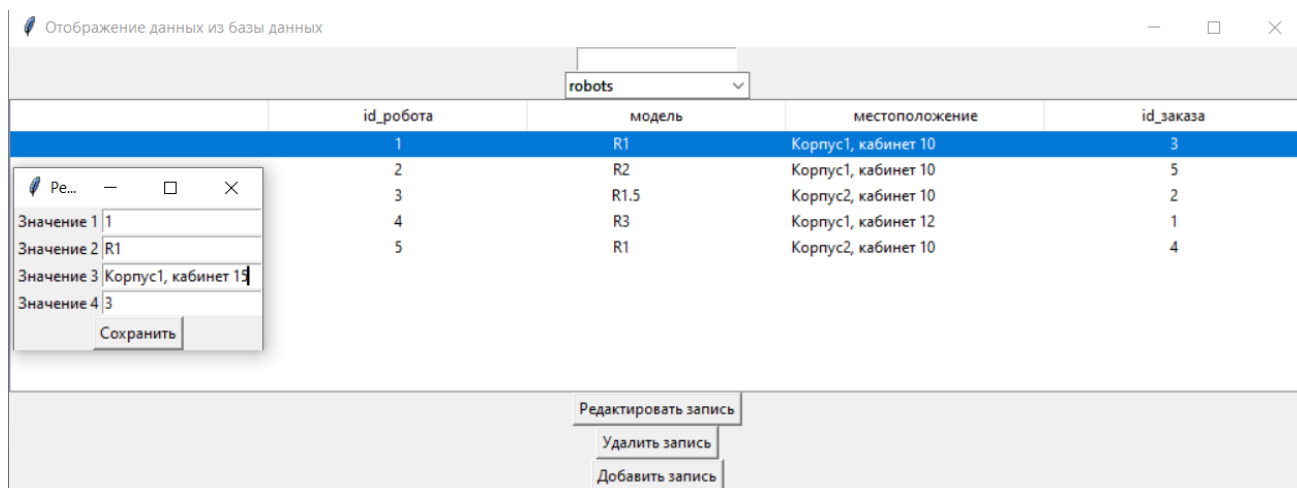


Рисунок 14 – Редактирование записи

На рисунке 14 показано редактирование выбранной записи и её сохранение.

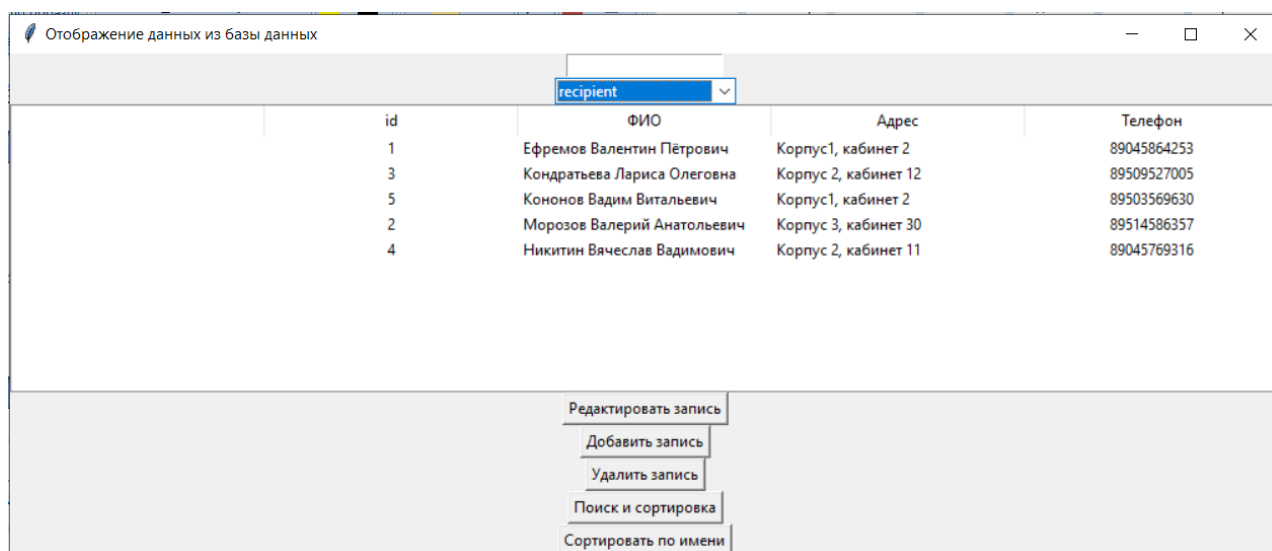


Рисунок 15 – Сортировка по алфавиту

На рисунке 15 показана сортировка данных по алфавиту. Сортировка происходит по 2 столбцу.

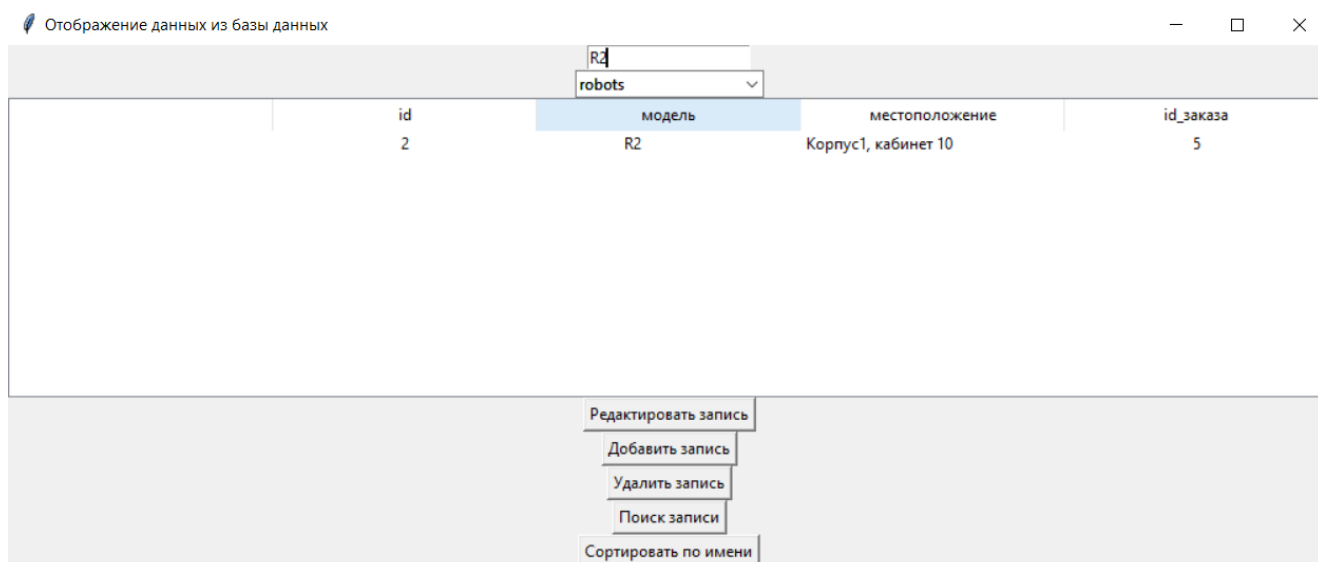


Рисунок 16 – Поиск записей по параметру

На рисунке 16 показана сортировка данных по заданному параметру. Параметр вводится самостоятельно в текстовом поле. Подходящие записи выводятся в отсортированном виде.

Подробная инструкция по работе и установке данной базы данных приведена в приложении 3 руководство пользователя.

Код для работы данной программы приведён в приложении 1 фрагмент исходного кода.

ЗАКЛЮЧЕНИЕ

В итоге данной курсовой работы становится ясным, что разработка базы данных для мобильных роботов доставщиков представляет собой важную часть для функционирования робота.

В процессе работы были рассмотрены ключевые моменты проектирование базы данных, способы обработки информации, работа с базой данных.

Были получены навыки по созданию, работе и администрированию базы данных. Познакомилась со структурой базы, с созданием связей данных в таблицах. Создала для неё графический интерфейс на языке python и реализовала в ней ряд взаимодействий с базой.

Занималась администрированием базы данных и сделала перенос её на другой сервер с помощью резервного копирования.

Данная программа была создана исключительно в учебных целях для дальнейшей работы с ней ее необходимо доработать.

СПИСОК ЛИТЕРАТУРЫ

1. База данных MS Access: что это такое и где находится, основные понятия, описание: сайт. – URL: <https://wiki.fenix.help/informatika/baza-dannyh-access> (дата обращения: 18.10.2023). – Текст: электронный.
2. Беспилотный робот доставщик Яндекс: сайт. – URL: https://yandex.ru/video/preview/7745208256999577511?from=tabbar&reqid=169998469b6396475-11465947086358061576-balancer-17leveler-kubr-yp-vla-40-BAL-5503&suggest_reqid=593423819163413205482277552990489&text (дата обращения: 20.10.2023). – Текст: электронный.
3. Беспилотный робот Яндекса занялся доставкой еды из ресторанов: сайт. – URL: https://auto.ru/mag/article/yarobotrestdelivery/?ysclid=loyp2xkn7i532197231&utm_referrer=yandex.ru (дата обращения: 18.01.2023). – Текст: электронный (дата обращения: 23.10.2023). – Текст: электронный.
4. В "Яндексе" ответили на вопрос о запуске летающих роботов-курьеров: сайт. – URL: <https://ria.ru/20230909/roboty-kurery-1895140857.html?ysclid=loyou2mbp878205761> (дата обращения: 29.10.2023). – Текст: электронный.
5. Встречаем ровер третьего поколения: история создания робота-курьера Яндекса: сайт. – URL: <https://habr.com/ru/companies/yandex/articles/590997/> (дата обращения: 05.11.2023). – Текст: электронный.
6. Глава 3. Программная реализация бд для курьерской службы "Московская доставка": сайт. – URL: <https://studfile.net/preview/16470734/page:11/> (дата обращения: 10.11.2023). – Текст: электронный.
7. ГОСТ 19.504–79. Единая система программной документации. Руководство программиста. Требования к содержанию и оформлению: сайт. – URL: <https://internet-law.ru/gosts/gost/31293/> (дата обращения: 17.11.2023). – Текст: электронный.

8. Как выглядит и как работает робот курьер Яндекса: сайт. – URL: <https://hi-news.ru/gadgets/kak-vyglyadit-i-kak-rabotaet-yandeks-rover-teper-uznat-eto-mozhet-kazhdyj.html> (дата обращения: 18.11.2023). – Текст: электронный.
9. Как работает Колобок: Корреспондент "РГ" стал очевидцем испытаний нового шарообразного робота курьера: сайт. – URL: <https://www.google.com/amp/s/rg.ru/amp/2023/08/28/reg-cfo/s-pylu-s-sharu.html> (дата обращения: 25.11.2023). – Текст: электронный.
10. Как роботы-доставщики "Яндекса" ездят зимой? Где уже есть роверы и сколько их?: сайт. – URL: <https://tass.ru/ekonomika/13136257> (дата обращения: 02.12.2023). – Текст: электронный.
11. Как работают роботы-курьеры Яндекса: сайт. – URL: <https://tproger.ru/articles/vopreki-nepogode-i-pod-opekoju-babuwek-kak-rabotayut-roboty-kurery-yandeksa> (дата обращения: 10.12.2023). – Текст: электронный.
12. Моделирование данных: обзор/Хабр (habr.com): сайт. – URL: <https://habr.com/ru/articles/556790/> (дата обращения: 18.12.2023). – Текст: электронный.
13. Основы правил проектирования базы данных: сайт. – URL: <https://habr.com/ru/articles/514364/> (дата обращения: 20.12.2023). – Текст: электронный.
14. Представлен первый в мире полностью автономный робот-курьер: сайт. – URL: <https://www.rbc.ru/life/news/63b7f7799a79475e882a3be9?ysclid=loy4mkdd350431645> (дата обращения: 26.12.2023). – Текст: электронный.
15. Приложения: Последние новости России и мира – Коммерсантъ Логистика (133682) – Ровер на посылках: сайт. – URL: <https://www.Kommersant.ru/doc/5076675> (дата обращения: 26.12.2023). – Текст: электронный.
16. Робот – курьер выясняет отношения, один против всех: сайт. – URL: <https://www.youtube.com/watch?v=iQWEQ9ot6j0> (дата обращения: 07.01.2024). – Текст: электронный.

17. Робот-курьер "Яндекса" начал доставлять заказы в жилые дома в Москве: сайт. – URL: <https://tass.ru/ekonomika/10675833> (дата обращения: 10.01.2024). – Текст: электронный.

18. Робот-курьер Ровер «Яндекса»: дорогостоящий проект с сомнительными перспективами или «убийца» профессии доставщиков и курьеров: сайт. – URL: <https://www.ixbt.com/live/car/rover-robot-kurer-ot-yandeksa-dorogostoyaschiy-proekt-s-somnitelnymi-perspektivami-ili-ubiyca-professii-dostavschikov-i-kureroov.html?ysclid=loyoqrpjh1v147637210> (дата обращения: 16.01.2024). – Текст: электронный.

19. Робот-курьер Сбербанка: сайт. – URL: <https://www.tadviser.ru/index.php> (дата обращения: 20.01.2024). – Текст: электронный.

20. Робот-курьер, доставляющий посылки: как работает новый сервис Почты России: сайт. – URL: <https://www.google.com/amp/s/riamo.ru/amp/524725/robot-kurer-dostavlyayuschij-posylki-kak-rabotaet-novyj-servis-pochty-rossii-x> (дата обращения: 21.01.2024). – Текст: электронный.

21. Роботы «Яндекса» будут доставлять еду в кампусах университетов США: сайт. – URL: <https://www.forbes.ru/newsroom/biznes/434133-roboty-yandeksa-budut-dostavlyat-edu-v-kampusah-universitetov-ssha> (дата обращения: 22.01.2024). – Текст: электронный.

22. Роботы захватывают мир? | Роботы-доставщики, города роботов и прочий беспредел: сайт. – URL: <https://domznaniy.school/tpost/b9hykeka61-roboti-zahvativayut-mir-roboti-dostavsch> (дата обращения: 25.01.2024). – Текст: электронный.

23. Роботы-курьеры «Яндекса» начнут доставлять посылки «Почты России» в Москве: сайт. – URL: <https://vc.ru/trade/309853-roboty-kurery-yan-deksa-nachnut-dostavlyat-posylki-pochty-rossii-v-moskve> (дата обращения: 25.01.2024). – Текст: электронный.

24. Создаем базу данных на примере службы доставки и разбираем запросы SQL: сайт. – URL: <https://habr.com/ru/articles/545550/> (дата обращения: 27.01.2024). – Текст: электронный.

25. Смогут ли роботы–доставщики вытеснить людей: сайт. – URL: <https://finance.rambler.ru/economics/47014419-smogut-li-roboty-dostavschiki-vyte-snit-lyudey/> (дата обращения: 30.01.2024). – Текст: электронный.
26. Транспортный робот для закрытых территорий и промышленных предприятий: сайт. – URL: <https://www.smprobotics.ru/produktsiya/avtonomnyy-mobilnyy-robot-dostavshchik/> (дата обращения: 03.02.2024). – Текст: электронный.
27. Уличные роботы–курьеры: сайт. – URL: <https://robotrends.ru/robope-dia/ulichnye-roboty-kurery> (дата обращения: 06.02.2024). – Текст: электронный.
28. Ходил 2 часа за роботом–доставщиком Яндекс и разобрался, как всё устроено: сайт. – URL: <https://trashbox.ru/link/yandex-rover-in-real-life-is-our-future?ysclid=loyr3geo4o581275527> (дата обращения: 09.02.2024). – Текст: электронный.
29. Что делают «Яндекс»–роботы? – Афиша Daily: сайт. – URL: <https://daily.afisha.ru/infoporn/22873-posmotrite-na-priklyucheniya-robota-kurera-v-rossii/> (дата обращения: 10.02.2024). – Текст: электронный.
30. Яндекс доставка в работе. Робот–курьер Ровер «Яндекса»: сайт. – URL: <https://www.youtube.com/watch?v=lpE4WeZIOnE> (дата обращения: 14.02.2024). – Текст: электронный.
31. Яндекс.Еда (Яндекс.Ровер): сайт. – URL: <https://www.tadviser.ru/index.php> (дата обращения: 16.02.2024). – Текст: электронный.
32. «Яндекс» запустил в работу третье поколение роботов–курьеров: сайт. – URL: https://4pda.to/2021/11/18/393240/yandeks_zapustil_v_rabotu_trete_pokolenie_robotov_kurero_v_video/ (дата обращения: 16.02.2024). – Текст: электронный.
33. «Яндекс» начал тестирование робота–курьера на колесах: сайт. – URL: <https://www.vedomosti.ru/technology/articles/2019/11/07/815680-yandex-robot-kurer> (дата обращения: 22.02.2024). – Текст: электронный.

34. «Яндекс» объявил о планах на 60% расширить парк роботов-курьеров в РФ: сайт. – URL: <https://habr.com/ru/news/755146/> (дата обращения: 25.02.2024). – Текст: электронный.

35. Яндекс проводит испытания автономного робота – доставщика Ровер: сайт. – URL: <https://habr.com/ru/news/474886/> (дата обращения: 28.02.2024). – Текст: электронный.

36. Kyocera ведет разработку роботов-доставщиков: сайт. – URL: <https://kiosksoft.ru/news/2023/03/02/kyocera-vedet-razrabotku-robotov-dostavshiko-v-28169> (дата обращения: 28.02.2024). – Текст: электронный.

37. Oracle Database: что это такое за база данных: сайт. – URL: <https://blog.skillfactory.ru/glossary/oracledatabase/?ysclid=ltjpdw3ost87499629> (дата обращения: 28.02.2024). – Текст: электронный.

38. PostgreSQL: что это за СУБД, основы и преимущества: сайт. – URL: <https://blog.skillfactory.ru/glossary/postgresql/?ysclid=ltjq41t3hq252737470> (дата обращения: 01.03.2024). – Текст: электронный.

39. SQLite – что это: сайт. – URL: <https://blog.skillfactory.ru/glossary/sqlite/?ysclid=ltjp3eu6o4282282612> (дата обращения: 01.03.2024). – Текст: электронный.

40. TAdviser: Новости ИТ-рынка России: сайт. – URL: <https://www.tadviser.ru/index.php:Starship> (дата обращения: 01.03.2024). – Текст: электронный.

Фрагменты исходного кода

```
import tkinter as tk
from tkinter import ttk
import psycopg2
from tkinter import messagebox
pupirka =
['orders', 'products', 'recipient', 'robots', 'sender', 'orderhistory']
stolbec = 1
def get_table_data(table_name):
    try:
        # Подключение к базе данных PostgreSQL
        connection = psycopg2.connect(
            dbname='postgres',
            user='postgres',
            password='admin',
            host='127.0.0.1',
            port='5432')
        cursor = connection.cursor()
        if table_combobox.get() == 'orders':
            pupirka =
['id', 'id_получателя', 'id_отправителя', 'id_товара', 'Статус']
            tree["columns"] = tuple(range(5))
            stolbec = 5
        if table_combobox.get() == 'products':
            pupirka = ['id', 'Название', 'Описание', 'Количество']
            tree["columns"] = tuple(range(4))
            stolbec = 4
        if table_combobox.get() == 'recipient':
            pupirka = ['id', 'ФИО', 'Адрес', 'Телефон']
            tree["columns"] = tuple(range(4))
            stolbec = 4
        if table_combobox.get() == 'robots':
            pupirka = ['id', 'модель', 'местоположение', 'id_заказа']
            tree["columns"] = tuple(range(4))
            stolbec = 4
        if table_combobox.get() == 'sender':
            pupirka = ['id', 'ФИО', 'Телефон', 'Адрес']
            tree["columns"] = tuple(range(4))
            stolbec = 4
        if table_combobox.get() == 'orderhistory':
            pupirka =
['id', 'id_заказа', 'id_робота', 'Дата', 'Отзыв']
            tree["columns"] = tuple(range(5))
            stolbec = 5
        for i in range(stolbec):
            tree.column(i, anchor='center')
            tree.heading(i, text=pupirka[i])
        # Выполнение запроса к базе данных
        cursor.execute(f"SELECT * FROM {table_name}")
```

```

        data = cursor.fetchall()
        # Закрытие соединения с базой данных
        connection.close()
        return data
    except psycopg2.Error as error:
        messagebox.showerror("Ошибка", f"Ошибка при доступе к базе
данных PostgreSQL: {error}")
def on_table_select(event):
    selected_table = table_combobox.get()
    if selected_table:
        # Получение данных для выбранной таблицы
        table_data = get_table_data(selected_table)
        # Очистка данных в Treeview
        for row in tree.get_children():
            tree.delete(row)
        # Вставка данных в таблицу
        for row in table_data:
            tree.insert("", tk.END, values=row)
def sort_by_name():
    selected_table = table_combobox.get()
    if selected_table:
        # Получение данных для выбранной таблицы
        table_data = get_table_data(selected_table)
        # Сортировка данных по столбцу "name"
        sorted_data = sorted(table_data, key=lambda x: x[1]) #
Предположим, что "name" находится во втором столбце
        # Очистка данных в Treeview
        for row in tree.get_children():
            tree.delete(row)
        # Вставка отсортированных данных в таблицу
        for row in sorted_data:
            tree.insert("", tk.END, values=row)
def edit_record():
    selected_item = tree.selection()
    if not selected_item:
        messagebox.showwarning("Предупреждение", "Выберите запись
для редактирования.")
        return
    selected_values = tree.item(selected_item)["values"]
    edit_window = tk.Toplevel()
    edit_window.title("Редактирование записи")
    entry_fields = []
    for i, value in enumerate(selected_values):
        tk.Label(edit_window, text=f"Значение {i+1}").grid(row=i,
column=0)
        entry_field = tk.Entry(edit_window)
        entry_field.grid(row=i, column=1)
        entry_field.insert(tk.END, str(value))
        entry_fields.append(entry_field)
    def save_changes():
        try:
            connection = psycopg2.connect(

```



```

        dbname='postgres',
        user='postgres',
        password='admin',
        host='127.0.0.1',
        port='5432')
    cursor = connection.cursor()
    if table_combobox.get() == 'orders':
        pupirka =
['id','id_получателя','id_отправителя','id_товара','Статус']
        if table_combobox.get() == 'products':
            pupirka =
['id','Название','Описание','Количество']
            if table_combobox.get() == 'recipient':
                pupirka = ['id','ФИО','Адрес','Телефон']
            if table_combobox.get() == 'robots':
                pupirka =
['id','модель','местоположение','id_заказа']
            if table_combobox.get() == 'sender':
                pupirka = ['id','ФИО','Телефон','Адрес']
            if table_combobox.get() == 'orderhistory':
                pupirka =
['id','id_заказа','id_робота','Дата','Отзыв']
                update_query = f"UPDATE {table_combobox.get()} SET "
                for i, value in enumerate(entry_fields):
                    update_query += pupirka[i]
                    update_query += f"= '{value.get()}'"
                    if i != len(entry_fields) - 1:
                        update_query += ", "
                update_query += f" WHERE id = '{selected_values[0]}'"
                cursor.execute(update_query)
                connection.commit()
                connection.close()
                messagebox.showinfo("Успех", "Запись успешно
обновлена.")
                edit_window.destroy()
                on_table_select(None)
            except psycopg2.Error as error:
                messagebox.showerror("Ошибка", f"Ошибка при обновлении
записи: {error}")
                save_button = tk.Button(edit_window, text="Сохранить",
command=save_changes)
                save_button.grid(row=len(selected_values), columnspan=2)
def delete_record():
    selected_item = tree.selection()
    if not selected_item:
        messagebox.showwarning("Предупреждение", "Выберите запись
для удаления.")
    return
    selected_id = tree.item(selected_item)["values"][0]
    try:
        connection = psycopg2.connect(
            dbname='postgres',

```

```

        user='postgres',
        password='admin',
        host='127.0.0.1',
        port='5432')
    cursor = connection.cursor()
    delete_query = f"DELETE FROM {table_combobox.get()} WHERE
id = %s"
    cursor.execute(delete_query, (selected_id,))
    connection.commit()
    messagebox.showinfo("Успех", "Запись успешно удалена.")
    on_table_select(None)
    connection.close()
except psycopg2.Error as error:
    messagebox.showerror("Ошибка", f"Ошибка при удалении
записи: {error}")
def add_record():
    add_window = tk.Toplevel()
    add_window.title("Добавление записи")
    entry_fields = []
    if table_combobox.get() == 'orders':
        stolbec = 5
    if table_combobox.get() == 'products':
        stolbec = 4
    if table_combobox.get() == 'recipient':
        stolbec = 4
    if table_combobox.get() == 'robots':
        stolbec = 4
    if table_combobox.get() == 'sender':
        stolbec = 4
    if table_combobox.get() == 'orderhistory':
        stolbec = 5
    for i in range(stolbec): # Предположим, что у нас есть 3
колонки для добавления данных
        tk.Label(add_window, text=f"Значение {i+1}").grid(row=i,
column=0)
        entry_field = tk.Entry(add_window)
        entry_field.grid(row=i, column=1)
        entry_fields.append(entry_field)
    def save_record():
        try:
            connection = psycopg2.connect(
                dbname='postgres',
                user='postgres',
                password='admin',
                host='127.0.0.1',
                port='5432')
            cursor = connection.cursor()
            if stolbec == 6:
                insert_query = f"INSERT INTO
[table_combobox.get()] VALUES (%s, %s, %s, %s, %s, %s)"
            if stolbec == 5:

```

```

        insert_query = f"INSERT INTO
{table_combobox.get()} VALUES (%s, %s, %s, %s, %s)"
        record_values = tuple(entry_field.get() for
entry_field in entry_fields)
        cursor.execute(insert_query, record_values)
        connection.commit()
        messagebox.showinfo("Успех", "Запись успешно
добавлена.")
        add_window.destroy()
        on_table_select(None)
        connection.close()
    except psycopg2.Error as error:
        messagebox.showerror("Ошибка", f"Ошибка при добавлении
записи: {error}")
        save_button = tk.Button(add_window, text="Сохранить",
command=save_record)
        save_button.grid(row=len(entry_fields), columnspan=2)
# Функция для выполнения поиска и сортировки
def search_and_sort():
    keyword = search_entry.get()
    selected_table = table_combobox.get()
    if selected_table:
        # Получение данных для выбранной таблицы
        table_data = get_table_data(selected_table)
        # Фильтрация данных по ключевому слову
        filtered_data = [row for row in table_data if
keyword.lower() in str(row).lower()]
        # Сортировка отфильтрованных данных (здесь предполагается
сортировка по первому столбцу)
        sorted_data = sorted(filtered_data, key=lambda x: x[0])
        # Очистка данных в Treeview
        for row in tree.get_children():
            tree.delete(row)
        # Вставка отсортированных и отфильтрованных данных в
таблицу
        for row in sorted_data:
            tree.insert("", tk.END, values=row)
# Создание окна
display_window = tk.Tk()
display_window.title("Отображение данных из базы данных")
# Добавление текстового поля для ввода данных
search_entry = tk.Entry(display_window)
search_entry.pack()
# Создание выпадающего списка для выбора таблицы
table_combobox = ttk.Combobox(display_window, state="readonly")
table_combobox.pack()
# Добавьте список таблиц в Combobox, например:
table_combobox['values'] = ['orders', 'products', 'recipient',
'robots', 'sender', 'orderhistory']
# или загрузите список таблиц из базы данных
# Создание Treeview для отображения данных
tree = ttk.Treeview(display_window)

```

```

# Определение колонок в таблице
# Замените это на логику, соответствующую вашей схеме таблицы
tree["columns"] = tuple(range(5)) # Здесь 3 - количество колонок
for i in range(5):
    tree.column(i, anchor='center')
    tree.heading(i, text=f"Column {i+1}")
tree.pack()
# Привязка события выбора таблицы к обработчику
table_combobox.bind("<<ComboboxSelected>>", on_table_select)
# Кнопки для редактирования и удаления записи
edit_button = tk.Button(display_window, text="Редактировать запись", command=edit_record)
edit_button.pack()
# Создание кнопки для добавления записи
add_button = tk.Button(display_window, text="Добавить запись", command=add_record)
add_button.pack()
delete_button = tk.Button(display_window, text="Удалить запись", command=delete_record)
delete_button.pack()
# Создание кнопки для добавления записи
search_button = tk.Button(display_window, text="Поиск записи", command=search_and_sort)
search_button.pack()
# Создание кнопки для сортировки по столбцу "name"
sort_button = tk.Button(display_window, text="Сортировать по имени", command=sort_by_name)
sort_button.pack()
# Запуск главного цикла окна
display_window.mainloop()

```

Руководство программиста

PostgreSQL служит для разнообразных целей, предоставляя мощные инструменты для управления базами данных: Поиск нужной информации, гибкий доступ и организация данных, защита данных, контроль версий и одновременный доступ, мониторинг состояния базы данных, настройка и контроль доступа [7].

1. Установить последнюю версию PostgreSQL (в 11 версии PostgreSQL поддерживаются только 64-х битные редакции Windows).
2. В процессе установки установить галочки на пунктах: PostgreSQL Server, PgAdmin 4, Stack Builder, Command Line Tools
3. Установите пароль для пользователя.
4. Установить порт, который нужно будет добавить в исключения в правилах фаервола.
5. Создать базу данных.
6. Изучить управление правами доступа для защиты информации.
7. Создать резервную копию базы данных, для предотвращения потери информации.
8. Настроить права удалённого доступа к базе.
9. Подключиться к созданной базе данных с помощью CMD.
10. Рассмотреть таблицы, поля и связи в базе данных, чтобы понимать как хранится информация.

ПРИЛОЖЕНИЕ 3

Руководство пользователя

1. Для установки PostgreSQL перейдите на сайт <https://www.postgresql.org> и перейдите в Download (Рисунок 17).



Рисунок 17 – Сайт

2. Выберите на какую операционную систему вы устанавливаете приложение (Рисунок 18).



Рисунок 18 – Выбор ОС

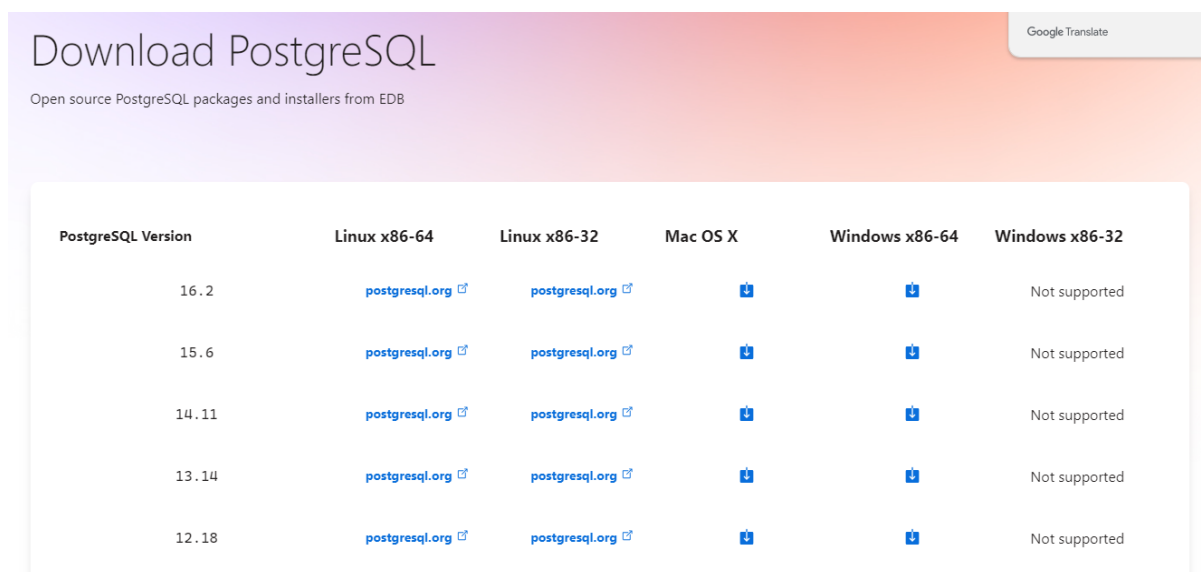
3. Перейдите по ссылке Download the installer. (Рисунок 19).

Interactive installer by EDB

Download the installer certified by EDB for all supported PostgreSQL versions.

Рисунок 19 – Ссылка

4. Скачайте последнюю версию дистрибутива для Windows, на сегодняшний день это версия PostgreSQL 11 (в 11 версии PostgreSQL поддерживаются только 64-х битные редакции Windows) (Рисунок 20).



Download PostgreSQL

Open source PostgreSQL packages and installers from EDB

PostgreSQL Version	Linux x86-64	Linux x86-32	Mac OS X	Windows x86-64	Windows x86-32
16.2	postgresql.org	postgresql.org			Not supported
15.6	postgresql.org	postgresql.org			Not supported
14.11	postgresql.org	postgresql.org			Not supported
13.14	postgresql.org	postgresql.org			Not supported
12.18	postgresql.org	postgresql.org			Not supported

Рисунок 20 – Версии программы

5. В процессе установки установить галочки на пунктах: PostgreSQL Server, pgAdmin 4, Stack Builder, Command Line Tools (Рисунок 21).

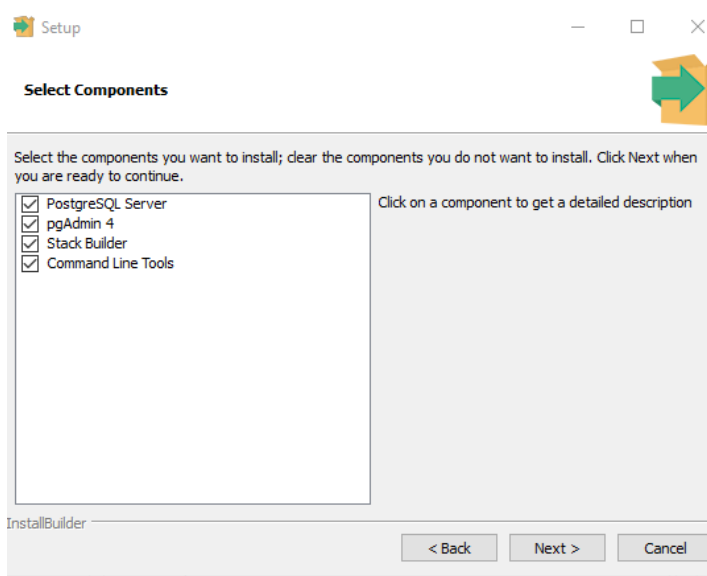


Рисунок 21 – Select Components

6. Установите пароль для пользователя postgres (он создается по умолчанию и имеет права суперпользователя) (Рисунок 22).

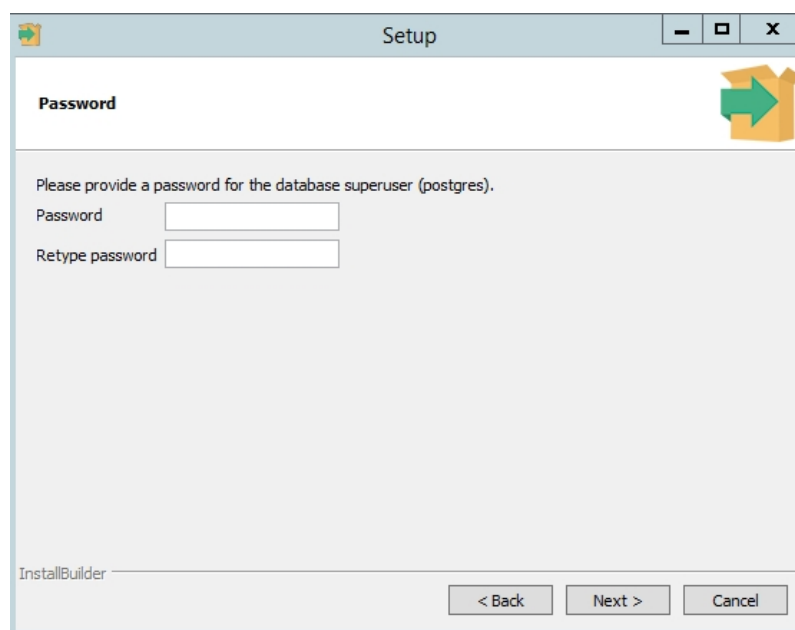


Рисунок 22 – Установка пароля

7. По умолчанию СУБД слушает на порту 5432, который нужно будет добавить в исключения в правилах фаервола (Рисунок 23).

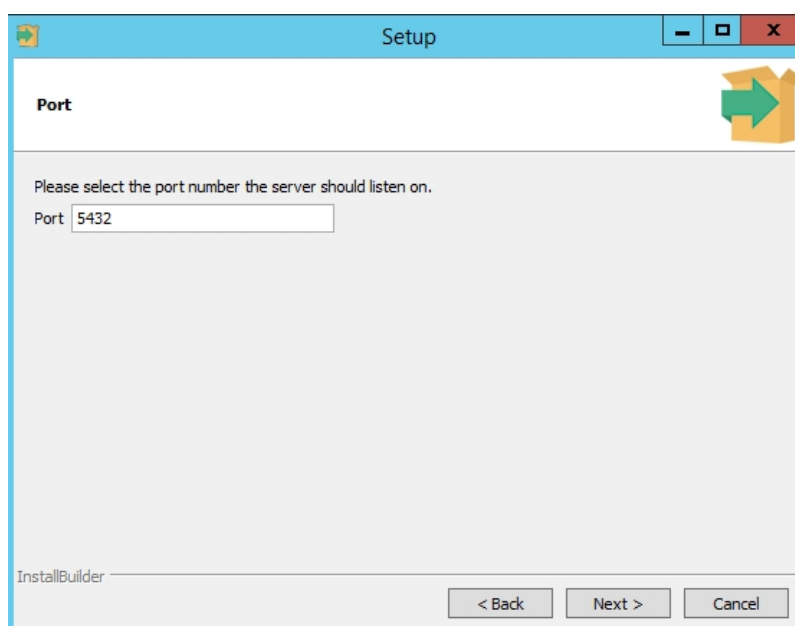
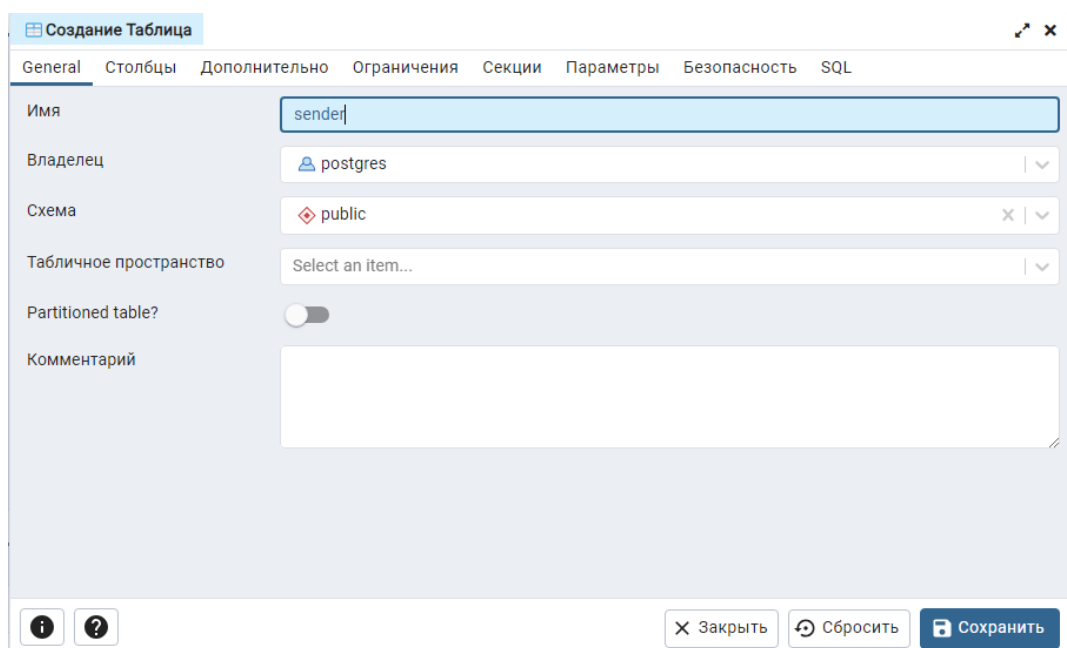


Рисунок 23 – Установка порта

8. Нажимаете Далее, Далее, на этом установка PostgreSQL завершена.
9. Открыть PostgreSQL и создать таблицы (Рисунок 24).



Создание Таблица

General Столбцы Дополнительно Ограничения Секции Параметры Безопасность SQL

Имя: sender

Владелец: postgres

Схема: public

Табличное пространство: Select an item...

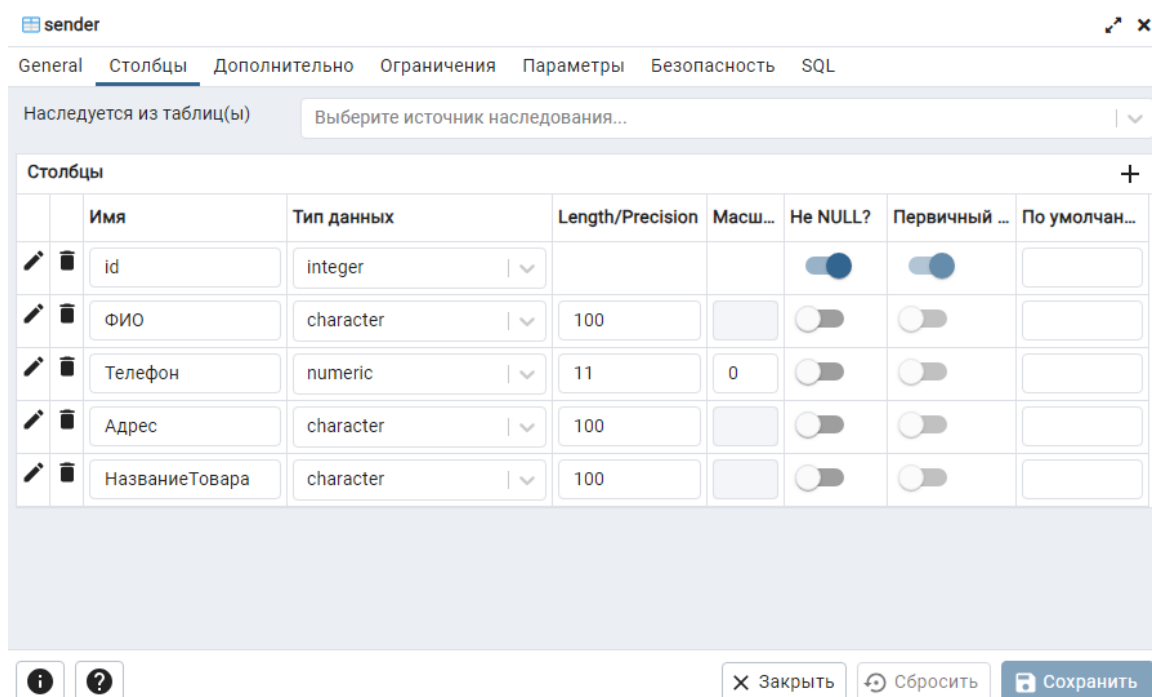
Partitioned table? ☐

Комментарий

Закреть Сбросить Сохранить

Рисунок 24 – Создание базы данных

10. В таблице задать названия столбцов и типы данных (Рисунок 25).



sender

General Столбцы Дополнительно Ограничения Параметры Безопасность SQL

Наследуется из таблиц(ы): Выберите источник наследования...

Столбцы

	Имя	Тип данных	Length/Precision	Масш...	Не NULL?	Первичный ...	По умолчан...
	id	integer			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
	ФИО	character	100		<input type="checkbox"/>	<input type="checkbox"/>	
	Телефон	numeric	11	0	<input type="checkbox"/>	<input type="checkbox"/>	
	Адрес	character	100		<input type="checkbox"/>	<input type="checkbox"/>	
	НазваниеТовара	character	100		<input type="checkbox"/>	<input type="checkbox"/>	

Закреть Сбросить Сохранить

Рисунок 25 – Добавление столбцов в таблицу

11. Заполнить таблицы данными.
12. Создать связи между таблицами, установить ключи.
13. Запускаем приложение.
14. Если приложение не запускается проверить данные используемые при подключении к базе данных (Рисунок 26).

```
# Подключение к базе данных PostgreSQL
connection = psycopg2.connect(
    dbname='postgres',
    user='postgres',
    password='admin',
    host='127.0.0.1',
    port='5432'
)
```

Рисунок 26 – Подключение к базе

15. Сверху из списка выбираем нужную таблицу для вывода данных этой таблицы (Рисунок 27).

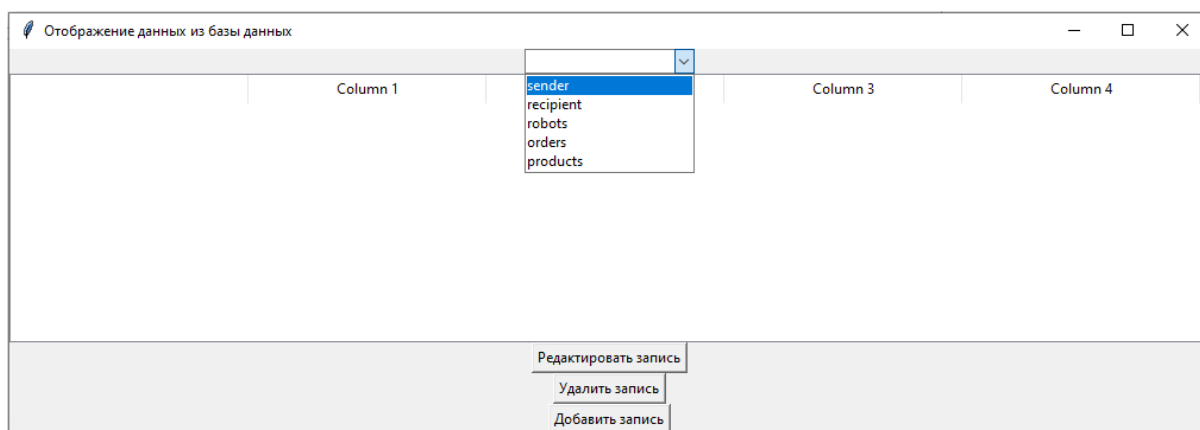


Рисунок 27 – Выбор таблицы

16. Для редактирования записи в таблице необходимо нажать на редактируемую запись и нажать на кнопку редактировать запись. Откроется

окно в котором нужно исправить данные и после нажать сохранить (Рисунок 28).

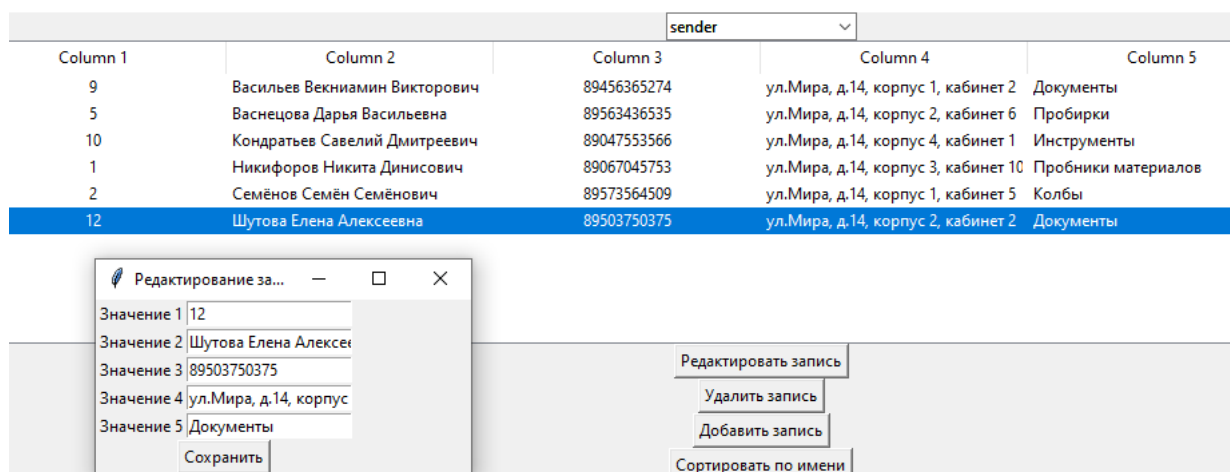


Рисунок 28 – Редактирование записи

17. Для удаления записи необходимо выбрать редактируемую запись и нажать удалить запись (Рисунок 29).

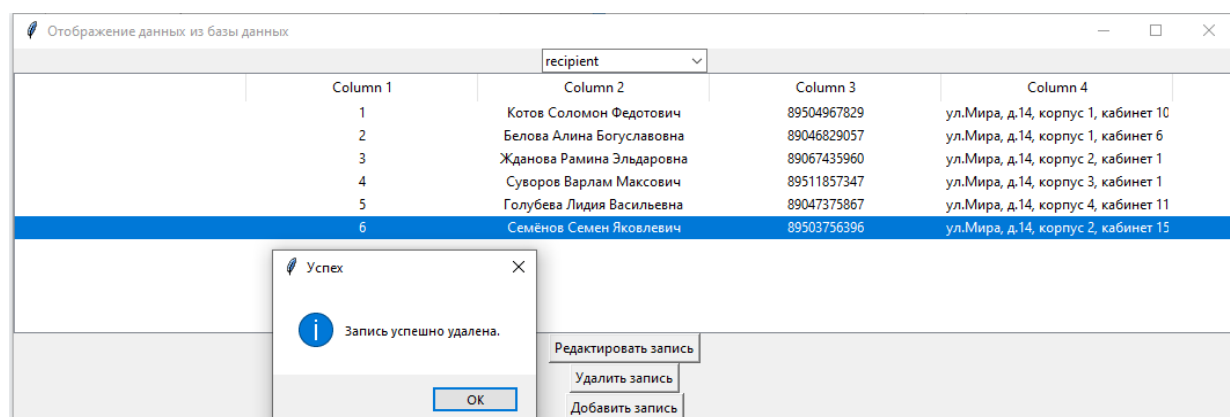


Рисунок 29 – Удаление записи

18. Для добавления записи нажимаем Добавить запись. В появившемся окне вводим данные и нажимаем сохранить (Рисунок 30).

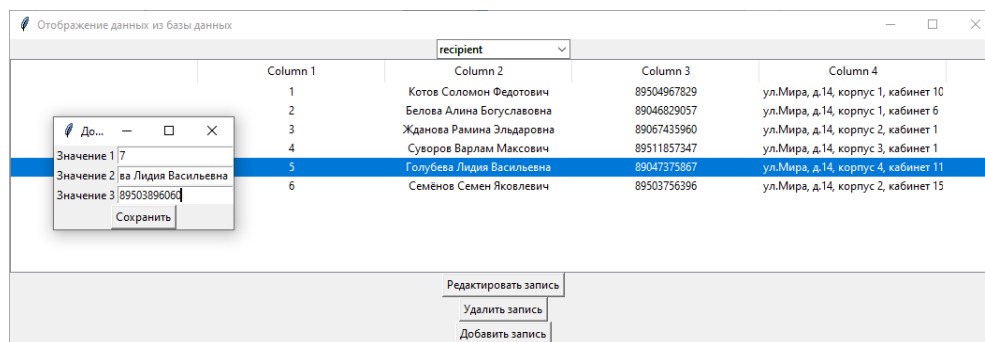


Рисунок 30 – Добавление записи

19. Для сортировки имени по алфавиту выбираем Сортировать по имени и данные сортируются (Рисунок 31).

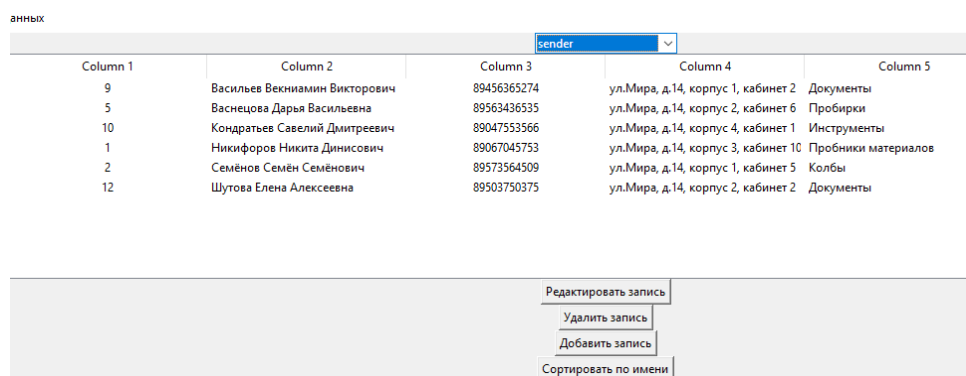


Рисунок 31 – Сортировка по алфавиту

20. Для сортировка данных по параметру следует в текстовом поле написать какие данные вам необходимо выбрать и нажать на поиск записи (Рисунок32).

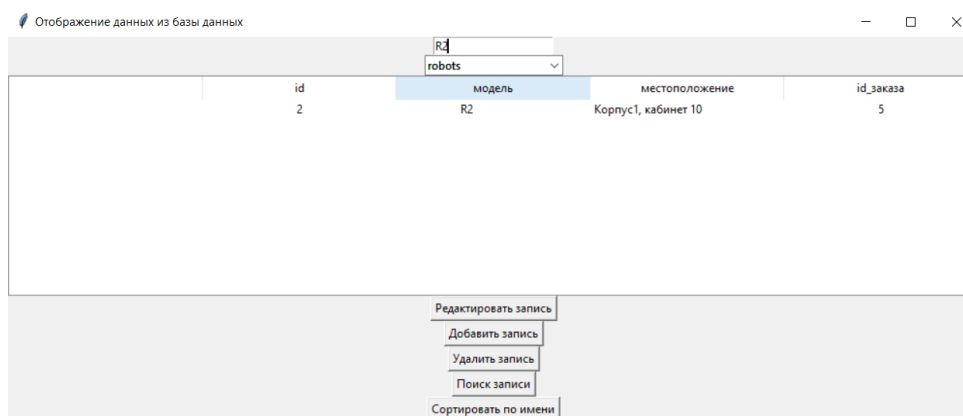


Рисунок 32 – Поиск записей по параметру