**Geekbrains**

**Специальность – программист Python, цифровые профессии**

**Тема дипломного проекта: Разработка информационной системы контроля трудовых ресурсов проекта**

**Студент: Шаверин Никита Владимирович**

Томск - 2023 г.

СОДЕРЖАНИЕ

[1. Введение 2](#_Toc142244069)

[1.1. Актуальность проекта 2](#_Toc142244070)

[1.2. Цели и задачи проекта, план работы 2](#_Toc142244071)

[1.3. Команда проекта 2](#_Toc142244072)

[2. Теоретическая часть 2](#_Toc142244073)

[2.1. Анализ существующего рынка программного обеспечения для реализации контроля трудовых ресурсов 2](#_Toc142244074)

[2.2. Определение требований к информационной системе 2](#_Toc142244075)

[2.3. Анализ архитектурных решений и алгоритмов для реализации информационной системы 2](#_Toc142244076)

[2.4. Изучение инструментов для создания информационной системы, определение основного стека технологий 2](#_Toc142244077)

[3. Практическая часть 2](#_Toc142244078)

[3.1. Архитектура информационной системы «Совместный труд» 2](#_Toc142244079)

[3.2. Разработка структуры базы данных информационной системы 2](#_Toc142244080)

[3.3. Разработка алгоритма модуля Telegram-bot 2](#_Toc142244081)

[3.4. Структура запросов и ответов модуля Telegram-bot 2](#_Toc142244082)

[3.5. Шаблоны отчётов 2](#_Toc142244083)

[3.5.1. Отчёт типа list 2](#_Toc142244084)

[3.5.2. Отчёт типа status 2](#_Toc142244085)

[3.5.3. Отчёт типа line 2](#_Toc142244086)

[3.5.4. Отчёт типа brigadier 2](#_Toc142244087)

[3.6. Обмен сообщениями между пользователями системы 2](#_Toc142244088)

[3.7. Автоматическая рассылка задач 2](#_Toc142244089)

[4. Заключение 2](#_Toc142244090)

[4.1. Выводы, полученные во время анализа теоретической базы и практического исследования 2](#_Toc142244091)

[4.2. Оценка проекта, описание его результатов 2](#_Toc142244092)

[4.3. Общий итог работы, её практическая значимость 2](#_Toc142244093)

[4.4. Предложения по совершенствованию информационной системы «Совместный труд» 2](#_Toc142244094)

[5. Список используемой литературы 2](#_Toc142244095)

[6. Приложения. 2](#_Toc142244096)

[6.1. Программный код модуля Telegram-bot. 2](#_Toc142244097)

[6.2. Скриншоты работы ИС. 2](#_Toc142244098)

1. Введение

1.1. Актуальность проекта

4 мая 1935 года на выпуске академиков Красной Армии в Кремлёвском дворце руководитель СССР И.В. Сталин произнёс фразу «Самый ценный капитал – это люди. Кадры решают всё!» [1]. Не смотря на неоднозначное отношение современников к личности и деятельности И.В. Сталина, многое из его тогдашнего выступления было положено в основу индустриализации и стремительного развития страны.

Действительно, трудовые ресурсы представляют собой важный фактор производства, рациональное использование которого обеспечивает не только повышение уровня производства и его экономической эффективности, но и качественное развитие всей социальной системы. Рациональное использование трудовых ресурсов предприятия влияет на результаты производственно-хозяйственной деятельности, выполнение бизнес-плана и на динамику выполнения плана производства.

Эффективное использование трудовых ресурсов может быть определено в результате оценки наличных трудовых ресурсов (анализа численности работающих, обеспечение предприятия рабочей силой, состава и структуры кадров, текучести кадров и т.д.) и разработки программы по их развитию. Цель анализа трудовых ресурсов состоит в том, чтобы разработать рекомендации по повышению эффективности использования трудовых ресурсов и обосновать планы по труду. Анализ трудовых ресурсов является важным элементом механизма управления предприятием и может применяться для обоснования принятия управленческих решений, в том числе в части оплаты труда работников предприятия.

Эффективно управлять организацией, не уделяя должного внимания контролю работы её работников, невозможно. Классическая триада менеджмента отдельного процесса проекта (постановка задачи, определение срока её исполнения, контроль и оценка выполнения задачи) наряду с возрастающей номенклатурой разновидностей работ и их диверсификацией обуславливает необходимость применения автоматизированных систем управления задачами проекта. Тщательно выстроенная система контроля требует скрупулезного отношения к таким вопросам, как правильное распределение нагрузки между работниками, оперативное отслеживание скорости и качества выполнения поставленных задач, формирование понятных схем взаимодействия между подразделениями и принципов удобного документооборота. Мониторинг всех этих аспектов может отбирать много сил. Сократить трудоемкость процесса помогают современные IT-решения, которые дают руководителя четкое представление о качестве работы персонала компании, статусе выполнения задач в режиме реального времени. Подобные программные системы, используемые вместе с иными мерами воздействия, позволяют предприятиям существенно улучшать ключевые показатели эффективности и выходить на качественно новый уровень в развитии бизнеса [2, 3].

1.2. Цели и задачи проекта, план работы

Целью настоящего проекта является разработка информационной системы контроля трудовых ресурсов.

Целевая аудитория проекта - трудовые коллективы и их руководители, которые:

- Ведут планирование и распределяют ресурсы;

- Собираются для обсуждения планов, распределения задач и подведения итогов;

- Во время проведения работ (возникновении проблем с их выполнением) имеют возможность оперативного оповещения коллег о процессе (выполнения работы);

- Имеют возможность проведения разнообразного анализа выполненной, текущей и предстоящей работы по проекту.

Объективные потребности:

Необходимо сделать доступной самостоятельную работу сотрудников и их локальных руководителей:

1. Информация об общей цели и о каждой задаче всегда были доступны;

2. Регулярные уведомления о работах;

3. До начала работ каждый сотрудник должен иметь информацию о том:

- где он находится;

- что он должен сделать;

- что ему потребуется;

- кто его задерживает;

- кого он задерживает;

- что является результатом работ.

4. Отслеживание эффекта от принятых решений:

- Сохранять историю и результаты работ;

- Рассчитывать эффективность хода работ (освоенный объём);

- Рассчитывать нагрузку сотрудников;

- Рассчитывать КПД объектов (оборудования и т.п.) над которым проводились работы.

Исходя из целей проекта основные задачи проекта определяются следующими:

1. Изучить текущее состояние рынка существующего программного обеспечения, адаптированного под специфику отечественного бизнеса;

2. Рассмотреть варианты создания программного обеспечения для оперативного контроля трудовых ресурсов;

3. Разработать архитектуру информационной системы, определить алгоритм её работы;

4. Изучить основные инструменты для реализации задач, определить необходимый стек технологий;

5. Реализовать информационный обмен с базой данных;

6. Реализовать сбор отчётов, необходимых руководителю бизнеса;

7. Разработать программу для работы информационной системы в Telegram (Telegram-bot).

План работы:

Постепенное и поступательное решение задач проекта. Активное участие в составе команды по разработке информационной системы контроля трудовых ресурсов проекта с целью доведения системы до стадии MVP (минимально жизнеспособный продукт) и последующего её модернизации.

Тестовое внедрение информационной системы планируется в Новосибирском агрокомбинате «Сады гиганта», Россия, Новосибирская область, рабочий посёлок Кольцово, 1/3, https:/sadygiganta.ru.

1.3. Команда проекта

Самый важный ресурс для достижения результата проекта - это люди, которые привлечены в проект. Разработка проекта в одиночку – устаревший бесперспективный подход в разработке программного обеспечения.

Команда проекта ― это малочисленная группа людей (обычно 3 -10 человек), которые владеют необходимыми для достижения единой цели знаниями и умениями и совместно отвечают за достижение результата [3].

Основной состав команды разработки проекта обычно такой:

* руководитель проекта
* менеджер проекта
* бэкенд-разработчик
* фронтенд разработчик
* UI/UX дизайнер
* тестировщик

Для достижения целей и выполнения задач проекта было принято решение о формировании классической команды разработчиков, состоящей, в основном, из действующих студентов сферы IT. Поскольку вначале проекта не было чёткого представления о виде и функциональности его конечного решения с учётом отсутствия финансирования разработка велась с использованием философии гибких методологий Agile. Принимая во внимание повседневную занятость членов команды, их удалённость друг от друга, работа над проектом велась с применением фреймворка Scrum и практик Kanban [15, 16].

В работе над проектом в разное время принимали участие следующие лица:

|  |  |  |
| --- | --- | --- |
| № | Фамилия, имя | Роль |
| 1 | Демченко Максим | Backend-developer |
| 2 | Джурабаев Дмитрий | Backend-developer |
| 3 | Ефремов Денис | Backend-developer |
| 4 | Захаров Александр | Frontend-developer |
| 5 | Орлов Дмитрий | Teamlead-BD/Analyst |
| 6 | Кузнецова Ксения | Frontend-developer |
| 7 | Ножкин Владислав | Teamlead Frontend-developer |
| 8 | Плотников Алексей | Product owner |
| 9 | Закиев Марат | Backend-developer |
| 10 | Шаверин Никита | Backend-developer |
| 11 | Бойко Никита | Project manager |
| 12 | Караткевич Артур | Backend-developer |
| 13 | Пещенкова Мария | Designer |
| 14 | Кашицина Анна | Project manager |
| 15 | Семочкин Илья | Frontend-developer |
| 16 | Матюшкин Алексей | Frontend-developer |
| 17 | Хазанов Максим | Frontend-developer |
| 18 | ??? Павел | Designer |
| 19 | Купрейчик Денис | Product manager |
| 20 | Котельников Алексей | Product manager |

Список разработчиков информационной системы является отрытым, особенность их статуса (действующие студенты в сфере IT) объясняет периодическую ротацию.

На момент моего включения в команду разработчиков я был способен разрабатывать алгоритмы выполнения программ, писать код на языке программирования Python, делать запросы на SQL, пользоваться GitHub, работать в операционной системе Linux, имел представление о «железе» персонального компьютера и компьютерных сетях. В процессе работы над проектом мои знания и умения существенно расширились.

2. Теоретическая часть

2.1. Анализ существующего рынка программного обеспечения для реализации контроля трудовых ресурсов

В ходе работы над проектом был изучен рынок программного обеспечения, предназначенного для организации управления проектами и задачами.

Основные продукты:

1. Bitcop

Bitcop – это система контроля выполнения задач и учета рабочего времени, которая способна анализировать все действия персонала за компьютером, что позволяет выявлять узкие места и оптимизировать бизнес-процессы. Система позволяет собирать данные о фактических затратах рабочего времени на задачи и проекты. Bitcop состоит из двух приложений. Одно, Bitcop Agency, собирает данные об активности сотрудников, мониторит их работу за компьютером, а второе, Bitcop Server, анализирует ту информацию, которая стекается на сервер со всех подключенных аккаунтов. Продукт может интегрироваться с другими цифровыми решениями (Redmine, Jira и т.д.), что позволяет собирать дополнительные данные и агрегировать их в системе [2].

Основные функции системы:

1. Формирование отчетов и наглядных графиков по затратам рабочего времени.
2. Оценка продуктивности разных временных интервалов.
3. Мониторинг открытых окон и заголовков в документах.
4. Отслеживание оффлайн-активности сотрудников.
5. Контроль нарушений и уведомление о таковых.
6. Периодическое создание скриншотов экрана ПК и их хранение в архиве.
7. Возможность установления нерабочего времени, в течение которого система не будет отслеживать действия сотрудника за компьютером.

Разработчик – российская компания «Биткоп».

1. Redmine

Redmine – открытое серверное веб-приложение для управления проектами и задачами [4].

Возможности системы:

- ведение нескольких проектов;

- система отслеживания ошибок;

- оповещение об изменениях посредством электронной почты и RSS-каналов;

- настраиваемые статусы задач;

- настраиваемые произвольные поля для задач, временных затрат, проектов и пользователей;

- учет временных затрат (часов);

- диаграммы Ганта и календарь;

- Wiki для каждого проекта;

- ведение новостей проекта, управление файлами и документами;

- форумы для каждого проекта;

- многоязыковой интерфейс, в том числе русский;

- легкая интеграция с репозиториями (SVN, CVS, Git, Mercurial, Bazaar и Darcs);

- система разделения доступа, основанная на ролях;

- поддержка множественной аутентификации LDAP;

- возможность самостоятельной регистрации новых пользователей;

- расширение функциональности системы посредством установки дополнительных плагинов;

- поддержка СУБД: MySQL, PostgreSQL, SQLite, MS SQL Server.

Разработчик Jean-Philippe Lang, Швейцария.

1. JIRA

JIRA – это продукт, предназначенный для организации управления проектами и задачами. Применение JIRA наиболее эффективно в компаниях, занимающихся разработкой каких-либо проектов или программного обеспечения, для управления деятельностью сотрудников и анализа эффективности [5].

JIRA приносит большой эффект любой компании, деятельность которой можно интерпретировать как выполнение каких-либо проектов, задач имеющих тематические и временные рамки. JIRA оперирует такими понятиями как проекты и задачи. Задачи создаются в проектах, на выполнение задач назначаются исполнители. Назначенный исполнитель приступает к выполнению задачи, комментирует ее, вводит проработанное время, присоединяет файлы к описанию задачи, просматривает файлы, относящиеся к задаче с поддержкой версионности. Все важные действия исполнителя журналируются, и о них получают сообщения руководители проекта и другие заинтересованные пользователя имеющие на это право.

JIRA позволяет контролировать ход выполнения задач и более крупно - проектов, управлять загрузкой исполнителей, выделять «узкие» места в проектах и важные задачи. Также, имеется возможность предоставить пользователям специальный доступ для ввода сообщений об ошибках и предложений по усовершенствованию.

В JIRA имеется система движения задач (workflow), позволяющая настраивать процесс движения задач в соответствии с требованиями принятыми на предприятии, управлять видом и набором полей описания задачи.

Разработчик JIRA – компания Atlassian, Австралия.

1. КСК.Service&Teamwork

КСК.Service&Teamwork – это облачное решение для организации работы и коммуникации команды как в офисе, так и в режиме удаленной работы. Позволяет контролировать, что происходит с проектом в целом и с отдельными задачами, настраивать списки задач под себя, использовать фильтры, чтобы получить больше информации о происходящем в проекте [7].

* Простые и сложные проекты в удобном интерфейсе
* Последовательное и параллельное исполнение задач, подзадачи
* Удобные инструменты для совместной работы по проектам
* Группировка задач вне проектов для исполнения и контроля
* Контроль, аналитика, отчеты

КСК.Service&Teamwork является российским продуктом, разработчик КСК Технологии.

1. A2B

А2Б - онлайн-система управления предприятием, предлагаемая к использованию на основе [SaaS](https://www.tadviser.ru/index.php/SaaS" \o "SaaS) (от англ. Software as a Service – программное обеспечение как услуга). Использовать А2Б можно с любого устройства (напр. смартфона, планшета, ноутбука), подключенного к сети интернет. Система представляет собой комплексное решение задач планирования, управления проектами, ведения продаж (CRM, контроля поручений, документооборота), организации взаимодействия между сотрудниками, учета идей [8].

А2Б – это удобные интернет-сервисы для организации таких внутренних бизнес-процессов как: долгосрочное и среднесрочное планирование, контроль поручений, управление структурой компании и персоналом, управление проектами, управление взаимоотношениями с клиентами, корпоративный портал.

Модули системы:

* CRM система - система управления взаимоотношениями с клиентами
* Проекты - система управления проектами
* Поручения - контроль исполнения поручений
* СЭД - система электронного документооборота
* Цели - стратегические цели компании
* Планы - оперативные планы подразделений и сотрудников
* Идеи - учет идей и рациональных предложений
* Расписания - ведение расписаний компании
* Главная страница - корпоративный портал компании
* Календарь - электронный ежедневник сотрудника
* Документы - хранилище электронных документов
* Персонал - телефонный справочник и управление структурой

Разработчик – ООО «А2Б», Россия.

1. Битрикс24

«Битрикс24» - это программный комплекс (сервис), предназначенный для оптимизации ведения бизнеса и контроля за производственным процессом. Битрикс24 – это набор полезных инструментов, которые помогают бизнесу работать: CRM, видеозвонки, документы онлайн, задачи и проекты, контакт-центр, конструктор сайтов и интернет-магазины. Битрикс24 помогает руководителю выстроить в своём коллективе эффективную работу [6].

Основные возможности Битрикс24:

* управление задачами и проектами;
* планирование рабочего времени;
* работа с документацией;
* автоматизация продаж;
* общение с клиентами.

Разработчик и провайдер – российская компания «Битрикс».

1. Аспро.Cloud

Аспро.Cloud – облачная платформа для совместного ведения проектов, постановки задач, управления продажами и контроля финансов. При помощи Аспро.Cloud компании автоматизируют бизнес-процессы, сокращают рутинных задач и налаживают совместную работу всех отделов. Платформа подойдёт для структурирования работы в малом бизнесе, небольших строительных компаний, рекламных и digital-агентств, юридических фирм, дизайн-студий и агентств недвижимости, производственных компаний и организаций, работающих по проектному подходу [9].

Особенности Аспро.Cloud:

* Ведение проектов любой сложности
* CRM с подробной историей по клиентам и автоматизацией
* Статистика и детализированные финансовые отчёты
* Гибкие права доступа для сотрудников
* Интеграция с популярными сервисами (телефония, банки, почта, конструкторы сайтов и т.д.)
* Выставление счетов, формирование КП и отправка актов
* 5 способов мониторинга задач: канбан-доска, диаграмма Ганта, календарь, список, GTD-планировщик
* Встроенный чат и моментальные уведомления
* Возможность создать собственную базу знаний и встроить ее на свой сайт
* Гостевой доступ для клиентов и подрядчиков
* Современное мобильное приложение

Разработчик – российская компания «Аспро».

1. Planny24

Planny24 – облачный сервис для управления процессами технической эксплуатации [13].

Основные возможности:

* Управление работами (нарядами);
* Ведение данных оборудования;
* Планирование работ;
* Отчётность и аналитика;
* Хранение технической документации;
* Облачный сервис.

Основные недостатки:

* Больше подходит для фрилансеров, среднего и малого бизнеса, индивидуальных предпринимателей;
* Только коробочное решение, нет возможности подобрать функционал под себя;
* Нет технической поддержки сотрудников;
* Авторизация с помощью e-mail-аккаунта;
* Многопользовательский доступ отсутствует;
* Отсутствует интеграция с программами.

Разработчик – ЗАО «Си Проект», Россия, г. Санкт-Петербург.

1. Мегаплан

Мегаплан - СРМ-система, созданная в первую очередь для управления проектами и обеспечения контроля над сотрудниками. Инструментарий направлен на организацию труда, оптимизацию рутинных обязанностей и автоматизацию бизнес-процессов [14].

Преимущества:

* Много функций и настроек;
* Детальный help-сервис;
* Интегрируется со многими другими программами;
* Ведение данных оборудования;
* Отчётность и аналитика;
* Управление работами.

Недостатки:

* Нужно устанавливать мобильное приложение;
* Платный облачный сервис с ежемесячной оплатой, либо дорогостоящая покупка и установка ПО на сервер компании;
* Медленная техническая поддержка.

Разработчик – российская компания «Первый бит», г. Омск.

На отечественном рынке пока находят себе применение такие иностранных приложения, как SingularityApp, TickTick, Todoist и др., однако в свете усиления санкционного давления недружественных стран с 2022 года их использование в ближайшей перспективе сопряжено с определенными рисками. К тому же некоторые приложения имеют скудный функционал в бесплатной версии, для них нужно устанавливать специальное приложение, у них высокая цена платных версий.

2.2. Определение требований к информационной системе

Анализ существующих программных продуктов обнаружил некоторые недостатки, обуславливающие дальнейшее проведение разработки ПО в данном направлении:

* необходимость установки дополнительного программного обеспечения на устройство конечного пользователя;
* сложность настойки системы для отдельного клиента;
* отсутствие возможности работы в системе при помощи существующих мессенджеров;
* сложность дальнейшей поддержки систем, разработчиком которых является иностранная компания.

Учитывая недостатки и преимущества существующих систем были сформированы требования к разрабатываемой системе.

Крайне желательно использовать существующую инфраструктуру. Без использования специальных мобильных приложений, процедур авторизации. Это должно дать лёгкое подключение пользователей в коллектив.

Информационный ресурс системы (база данных, СУБД, модули программного кода, конфигурационная информация) должен находиться на удалённом сервере. Общение с системой должно быть простым, без прохождения обучения. Максимально быстрое внедрение системы.

Проекту дадут конкурентные преимущества:

I) использование имеющихся приложений:

1. Телеграмм:

- для оперативного получения информации по задачам, долгам и планам;

- для получения уведомлений о задачах;

- общения с коллегами;

- отправки результатов работ;

2. Браузер:

- для отображения всего плана работ;

- отображения распределения нагрузки;

- карточки объекта;

II) Использование диаграмм для представления планов на различный период в прошлом и будущем:

- Упрощает инструментарий, единый интерфейс;

- Умещает весь период на одном экране;

- Позволяет переключаться и детализировать между группами оборудования, сотрудниками, работами;

- Отображает коэффициенты эффективности;

- Отображает информацию по объекту, сотруднику или работе;

III) Общение с телеграм-ботом:

1. Естественный язык запросов, например: Мой план на завтра;

2. Использование синонимов и умолчаний:

- Мой план/Мои задачи на завтра/задачи на завтра;

3. Возможность переключиться на меню;

4. Регулярные уведомления;

5. Рассылка сообщений.

25 марта 2023 года на очередном собрании членов команды разработчиков системы было утверждено её название - «Совместный труд». Основной лозунг - ставь задачи, береги время, достигай цели.

2.3. Анализ архитектурных решений и алгоритмов для реализации информационной системы

Архитектура информационной системы – это описание продукта программирования (операционных систем, приложений, программ) с точки зрения пользователя, заказчика и специалиста по проектированию. В ней объясняется, из чего состоит система, как элементы взаимодействуют между собой [10]. Любая информационная система включает в себя три компонента:

1. Управление данными;
2. Бизнес-логика;
3. Пользовательский интерфейс.

Данные хранятся в базах данных, а управление ими осуществляется с помощью системы управления базами данных. Бизнес-логика определяет правила, по которым обрабатываются данные. Она реализуется набором процедур, написанных на различных языках программирования. Пользователь работает с интерфейсом, где логика работы информационной системы представлена в виде элементов управления – полей, кнопок, списков, таблиц и т.д. Компоненты в разных информационных системах взаимодействуют между собой различными способами [11].

Существуют следующие виды архитектур информационных систем [11, 12]:

1. Локальные информационные системы. В этом случае все компоненты системы располагаются на одном компьютере. Очевидным недостатком этой архитектуры является возможность работать только одному пользователю. Другие пользователи не имеют возможности получить доступ к данным даже для чтения
2. Файл-серверная архитектура. В данной архитектуре данные хранятся в файлах на выделенном специально для этой цели компьютере (файловом сервере). Компьютеры пользователей соединены с сервером сетью, поэтому доступ к данным могут получить несколько пользователей одновременно. Однако, кроме функций хранения данных и обеспечения доступа к ним, файловый сервер никаких функций не выполняет. Приложения, обрабатывающие данные, находятся на пользовательских компьютерах. Обработка данных на пользовательском компьютере всегда сопровождается передачей по сети большого количества информации. Недостатками данной архитектуры являются высокая загруженность сети (и, как следствие, низкая скорость работы), сложность поддержания непротиворечивости данных из-за несогласованной обработки разными пользователями.
3. Клиент-серверная архитектура. В данном виде архитектуры в состав системы управления базами данных включается новый компонент – процедурный язык программирования, с помощью которого появляется возможность создания хранимых процедур обработки данных и обработки их значительного объёма непосредственно на сервере. Данная архитектура позволяет существенно разгрузить сеть и поддерживать непротиворечивость данных за счёт их централизованной обработки. Однако языки хранимых процедур не приспособлены для полноценной реализации бизнес-логики и она реализуется на клиентском компьютере. Такой подход имеет следующие недостатки: любые изменения в бизнес-логике требуют обновления на клиентском компьютере, клиентские компьютеры должны быть достаточно производительными, система слабо защищена от взломов.
4. Трёхуровневая (многоуровневая) архитектура. В данном виде архитектуры реализация бизнес-логики перенесена с клиентских компьютеров на специальный промежуточный уровень – сервер приложений, представляющий собой комплекс программ, выполняемых на сервере. Использование сервера приложений позволяет максимально разгрузить клиентские компьютеры и сделать обработку данных ещё более централизованной, что повышает скорость и надёжность информационной системы.
5. Многоуровневая интернет-архитектура. В данном виде архитектуры к рассмотренным выше уровням добавляется web-сервер (сервер, принимающий HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы, как правило, вместе с HTML-страницей, изображением, файлом, медиа-потоком или другими данными). Примером многоуровневой интернет-архитектуры можно назвать веб-приложения. Их отличительной особенностью является передача запросов и результатов их обработки через сеть Интернет и использование в качестве клиентского приложения стандартного Интернет-браузера. При этом структура информационного системы приобретает следующий вид: браузер – web-сервер – сервер приложений – сервер баз данных. Благодаря интеграции Интернет-технологии и архитектуры клиент-сервер процесс внедрения и сопровождения корпоративной информационной системы существенно упрощается при сохранении достаточно высокой эффективности и простоты совместного использования информации.

2.4. Изучение инструментов для создания информационной системы, определение основного стека технологий

Для создания информационной системы «Совместный труд» разработчиками была выбрана многоуровневая интернет-архитектура. Её реализация потребует наличие следующих инструментов:

Таблица 1. Инструменты для создания многоуровневой интернет-архитектуры

|  |  |  |  |
| --- | --- | --- | --- |
| № | Инструмент | Уровень | Варианты |
|  |  | Браузер | Google Chrome, Internet Explorer, Mozilla FireFox |
|  |  | Web-сервер | Apache, Nginx |
|  |  |  |  |
|  |  | Сервер приложений |  |
|  |  |  |  |
|  |  | Система управления базой данных | PostgreSQL, MySQL |
|  |  |  |  |

Таблица2. Стек основных технологий для создания информационной системы «Совместный труд»

|  |  |  |
| --- | --- | --- |
| № | Технология | Задачи |
|  | Аспро.Cloud | Управление процессом разработки программного обеспечения. |
|  | Figma | Cовместное проектирование дизайна пользовательских интерфейсов |
|  | Html, CSS, JavaScript, WebSocket | Разработка Web-приложения |
|  | Python (native и фреймворки Telebot, MatPlotLib, NumPy, SciPy) | Разработка сервера приложений |
|  | PostgreSQL | Система управления базой данных |
|  | ОС Linux (Ubuntu 22.04.2) | Сервер |
|  | Git, GitHub | Управление версиями, хранилище программного кода |
|  | PyCharm | Среда разработки программного обеспечения |
|  | DBeaver | Администрирование базы данных |
|  | Apache | Web-сервер |

3. Практическая часть

3.1. Архитектура информационной системы «Совместный труд»

Описание

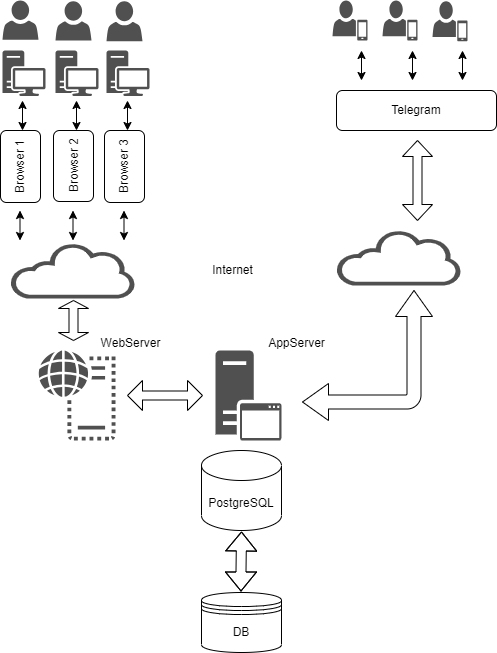


Рисунок 1. Структура информационной системы «Совместный труд»

Описание

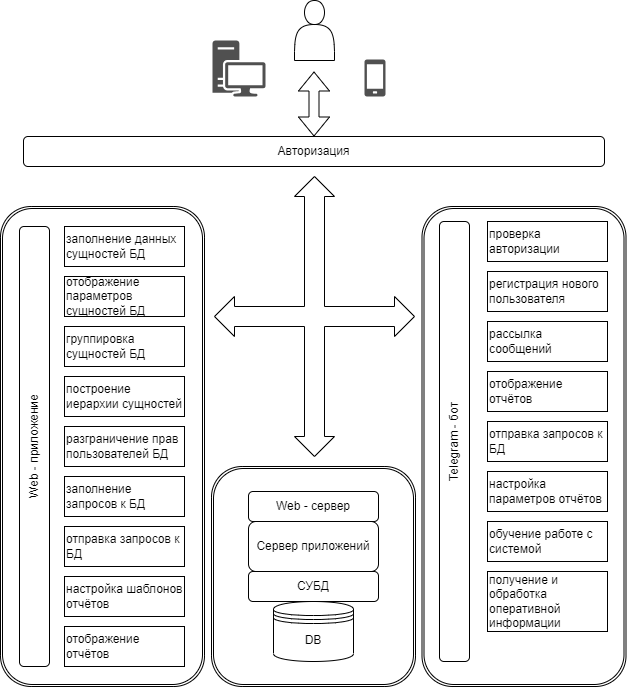


Рисунок 2. Функциональная схема информационной системы «Совместный труд»

Блок-схема с описанием и местонахождением модулей

Проект на GitHub - <https://github.com/lesenkagit/sovtrud>

Сервер на Ubuntu 22.04.2 LTS

Сервер БД по адресу - http://91.77.164.63/

Telegram-bot - https://t.me/sovtrudBot

3.2. Разработка структуры базы данных информационной системы

База данных - организованная структура, предназначенная для хранения информации. Современные базы данных позволяют размещать в своих структурах не только данные, но и методы, с помощью которых происходит первичная обработка хранящейся в базе данных информации. Структура базы данных определяется составом и типами хранимых в ней записей, их организацией, связью между собой, а также особенностями предметной области и механизмами взаимодействия с сервером приложений (программным кодом приложения).

Для информационной системы «Совместный труд» была выбрана реляционная модель базы данных, использующая табличный способ хранения данных и состоящая из нескольких раздельных таблиц. В качестве СУБД была определена одна из наиболее популярных СУБД объектно-реляционная PostgreSQL, версия 15.3. Одним из определяющих факторов в пользу выбора СУБД PostgreSQL стала возможность хранения тип данных JSON в таблице SQL, поскольку данный формат позволяет существенно упростить масштабируемость информационной системы и неизменным добавлением априори неизвестных типов данных.

Сущностями информационной системы являются:

* Object – объект предметной области (проект, основное средство, предмет и т.д.);
* Relation – работа, выполняемая с объектом, её описание;
* Subject – физическое лицо, производящее работу. Тот, на кого можно назначить выполнение работы (постановку задачи).

В общем виде структура сущностей внутри БД одинакова.

Задачи, возлагаемые на СУБД:

1. Хранение информации о сущностях;
2. Хранение информации о задачах и их описании;
3. Хранение информации о правах и группах сущностей;
4. Хранение информации о сообщениях между сущностями;
5. Хранение информации об авторизации;
6. Хранение информации об отчётах;
7. Хранение информации о callback\_data Inline Keyboard телеграмм-бота. Необходимо из-за ограничения Telegram в 64 байта.

Для реализации поставленных задач в информационной системе предусмотрены следующие таблицы:

1. Entity – информация о сущностях;
2. Plan – информация о задачах;
3. Giduid – таблица соответствия групп и сущностей;
4. Authwait – таблица авторизации;
5. Message – таблица сообщений;
6. Comments – таблица комментариев;
7. Keyscallback – таблица calback\_data telegram-бота.

Рассмотрим основные из них.

Таблица ??? Entity. Хранение информации о сущностях.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя столбца | Тип данных | Описание |
| 1 | id | serial4 | Entity id |
| 2 | parentid | int4 | id родительской сущности |
| 3 | uid | int4 | id создателя |
| 4 | gid | int4 | id группы сущности |
| 5 | type | varchar(32) | тип сущности (object, group, subject, relation, project) |
| 6 | name | varchar(128) | имя сущности |
| 7 | parameters | jsonb | параметры сущности, хранятся в NoSQL бинарном представлении данных формата json |
| 8 | rights | varchar(10) | права сущности |
| 9 | is\_active | bit(1) | состояние сущности |

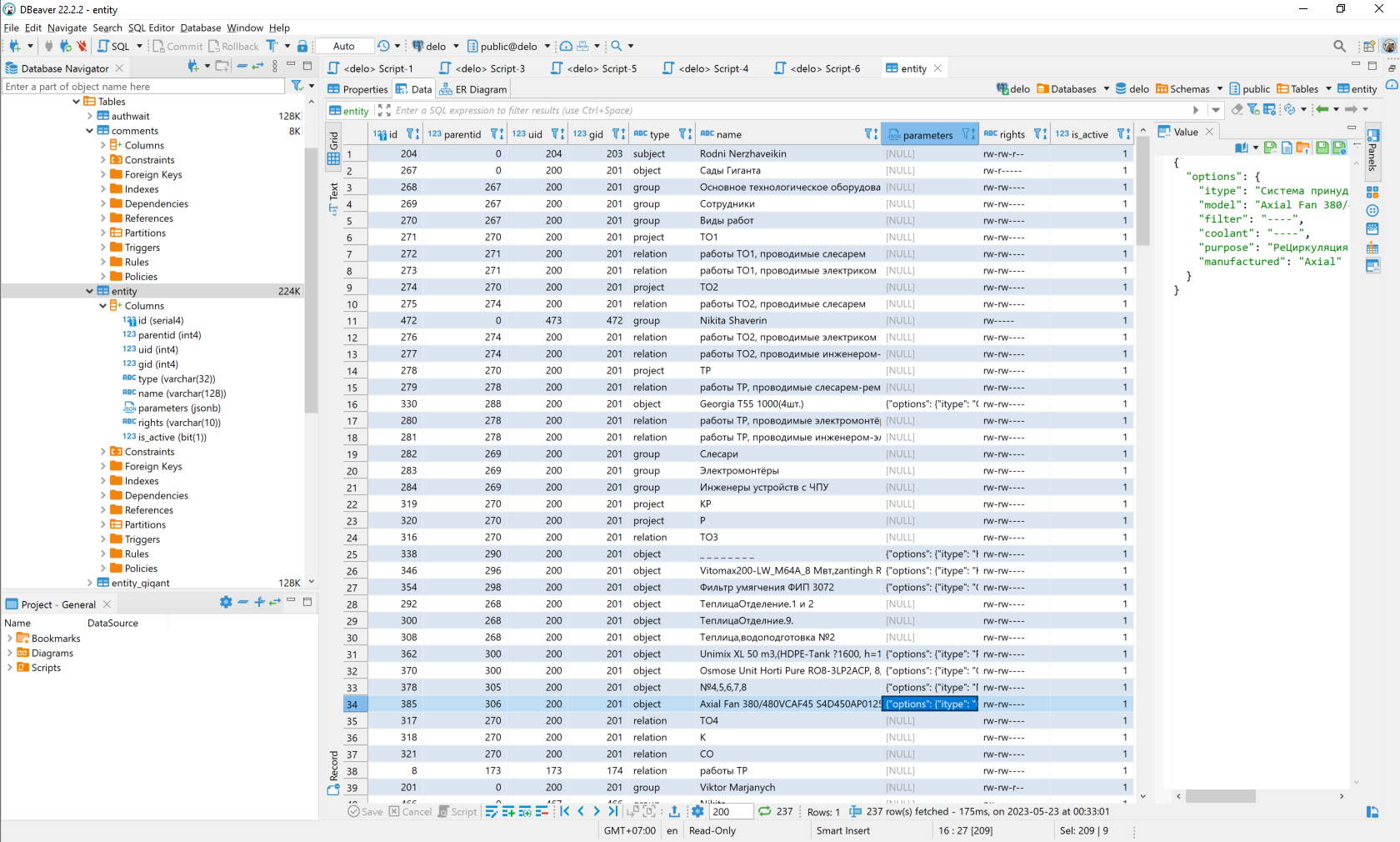


Рисунок ??? Скриншот фрагмента таблицы Entity DBeaver

Таблица ??? Plan. Хранение информации о задачах.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя столбца | Тип данных | Описание |
| 1 | id | serial4 | Plan id |
| 2 | subjectid | int4 | id субъекта, на кого поставлена задача |
| 3 | relationid | int4 | id работы, которая включена в задачу |
| 4 | objected | int4 | id объекта, который включён в задачу |
| 5 | plandate | date | плановая дата выполнения задачи |
| 6 | factdate | date | фактическая дата выполнения задачи |
| 7 | anoncedate | date | дата постановки задачи |
| 8 | prev | int4 | id предыдущей задачи |
| 9 | next | int4 | id следующей задачи |
| 10 | projectid | int4 | id проекта |
| 11 | projectmark | varchar | поле комментария |

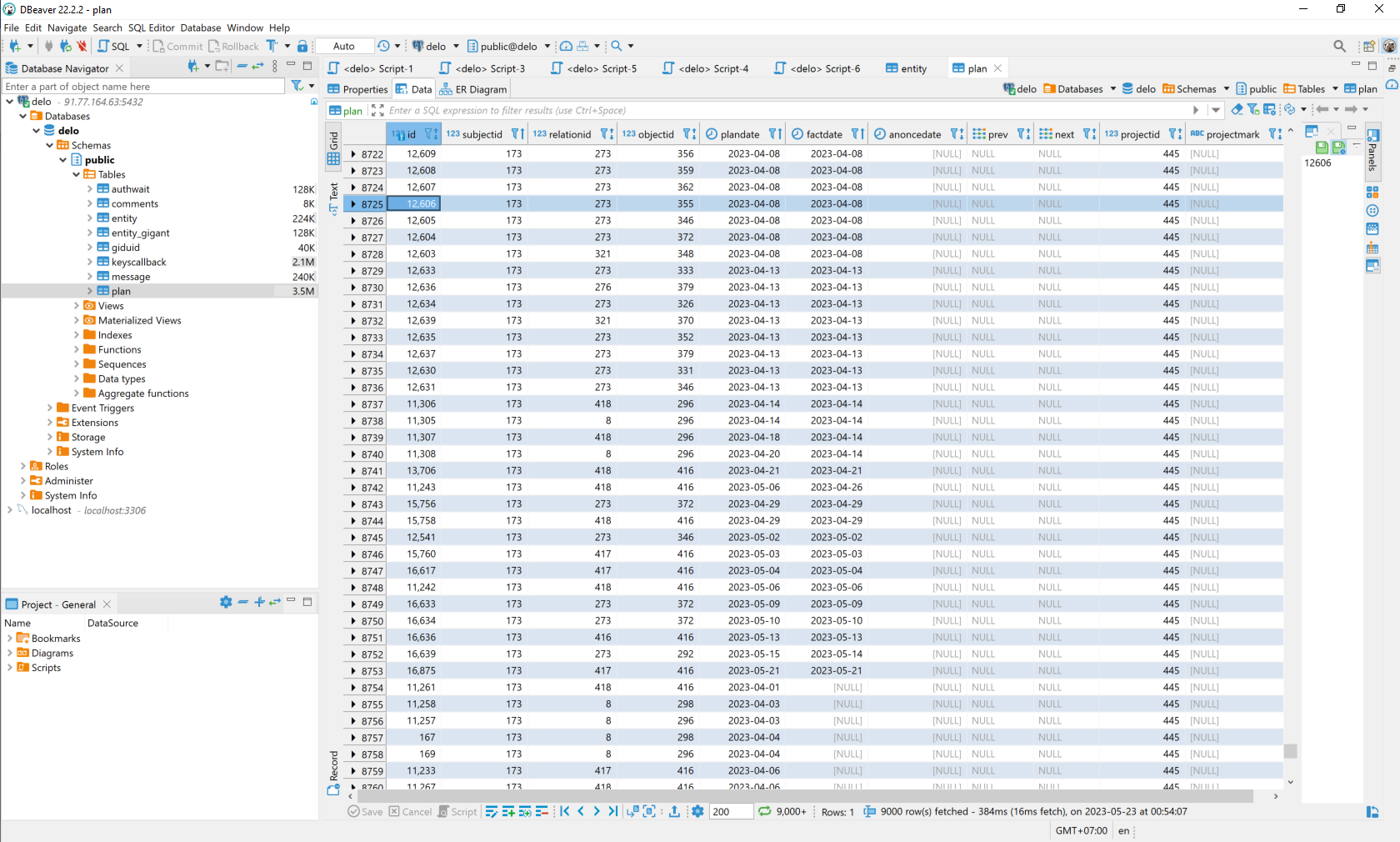


Рисунок ??? Скриншот фрагмента таблицы Plan DBeaver

Таблица ??? GidUid. Группы и пользователи.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя столбца | Тип данных | Описание |
| 1 | gid | int4 | Group\_id |
| 2 | entity\_id | int4 | Entity id |

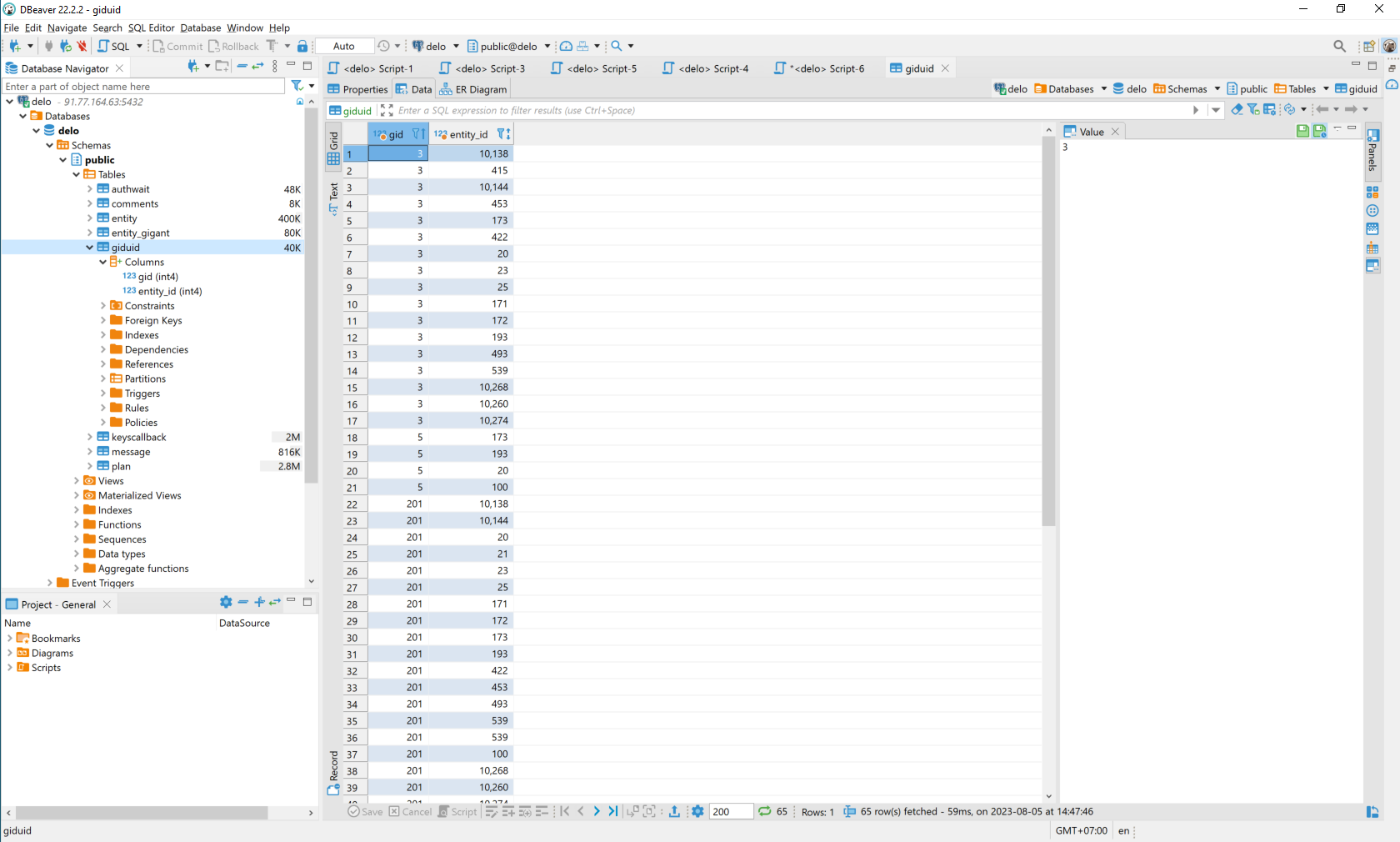


Рисунок ??? Скриншот фрагмента таблицы GidUid DBeaver

Таблица ??? Message. Сообщения пользователей.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя столбца | Тип данных | Описание |
| 1 | uid | int4 | id субъекта, кому предназначено сообщение |
| 2 | plandate | timestamp | время записи сообщения в базу данных |
| 3 | senddate | timestamp | время отправки сообщения телеграмм-ботом получателю |
| 4 | text | varchar | текст сообщения |
| 5 | parse\_mode | varchar | формат сообщения |
| 6 | mid | serial4 | идентификационный номер сообщения |

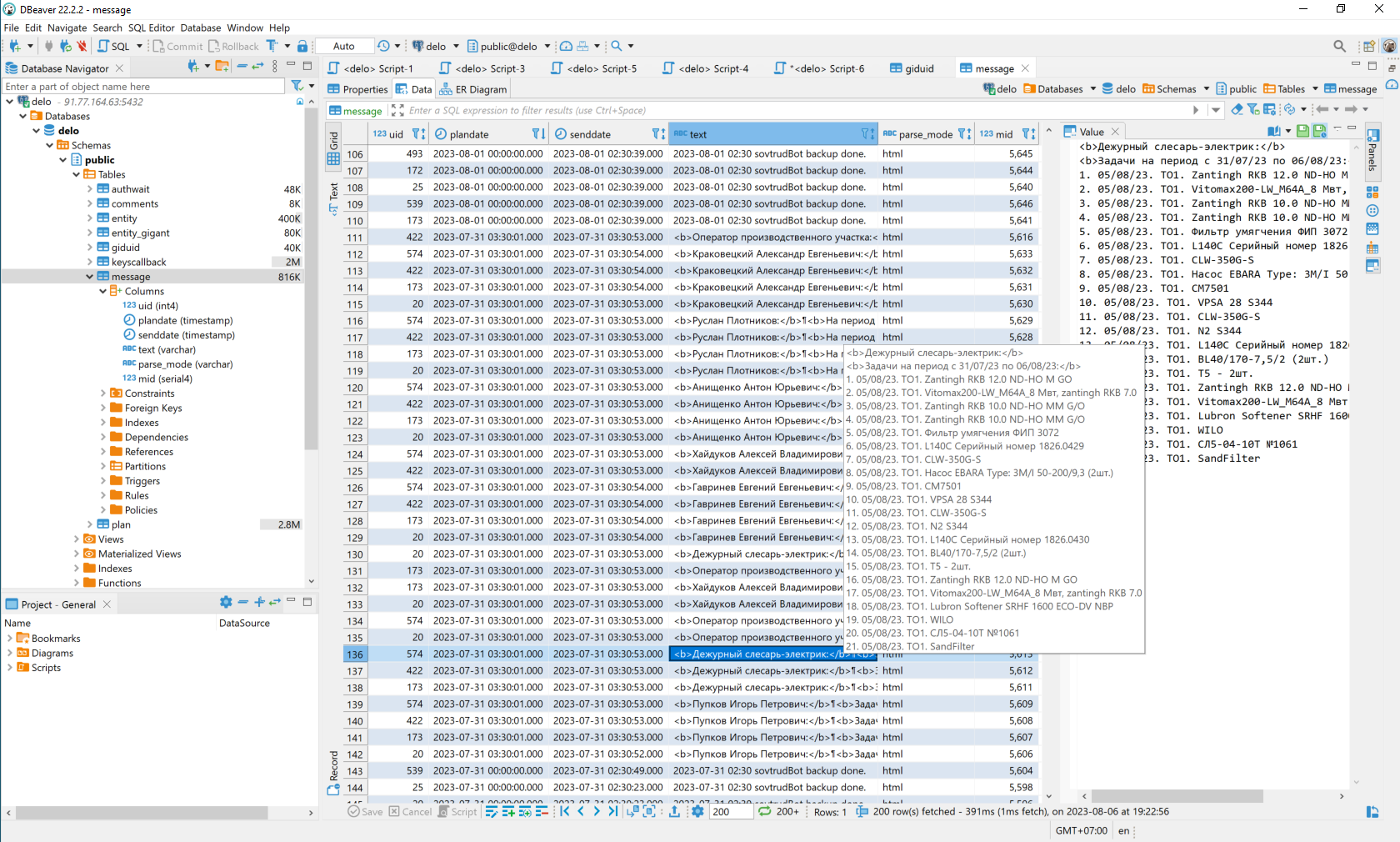


Рисунок ??? Скриншот фрагмента таблицы Message DBeaver

Таблица ??? Keyscallback. Таблица calback\_data telegram-бота.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя столбца | Тип данных | Описание |
| 1 | chatid | int8 | telegram chatid |
| 2 | hash | varchar | hash callback |
| 3 | callback | varchar | callback на button Inline Keyboard telegram-бота |
| 4 | timeout | timestamp | время жизни callback |

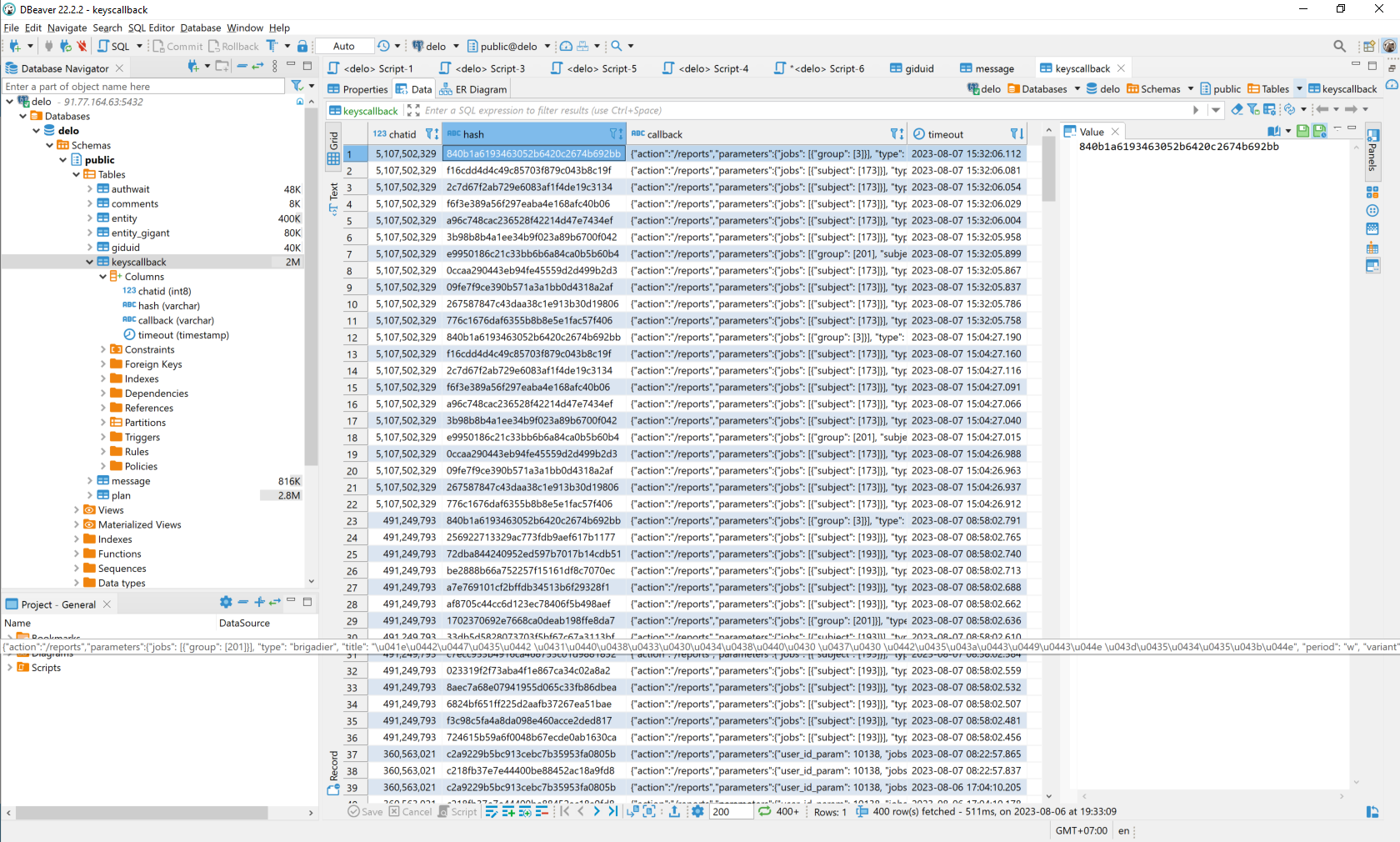


Рисунок ??? Скриншот фрагмента таблицы Keyscallback

На уровне SQL реализованы следующие функции:

|  |  |  |
| --- | --- | --- |
| № | Функция | Назначение |
| 1 | get\_all\_node\_children | получение потомков сущности |
| 2 | get\_group\_entities | получение сущностей, входящих в группу |
| 3 | get\_user\_groups | получение групп, в которые входит пользователь |
| 4 | isrights | получение прав сущности |
| 5 | newuser | создание нового пользователя |

3.3. Разработка алгоритма модуля Telegram-bot

Модуль Telegram-bot разработан с использованием библиотеки telebot [17, 18]. Алгоритм его работы приведён на рисунках ??? В начале работы происходит загрузка конфигурационной информации из файла (параметры подключения к базе данных, token телеграмм-бота, параметры логирования и т.д.), демоном запускается телеграмм-бот, который с заданной периодичностью проверяет наличие новых сообщений. В случае прихода новых сообщений стартует модуль worker, в который передаётся полученное сообщение. Модуль worker запускается в отдельном потоке для каждого сообщения. Если новых сообщений с момента прошедшего опроса состояния не поступило, запускается модуль housekeeper, который делает рассылку всех неотправленных сообщений из таблицы message согласно указанным в них пользовательским telegram\_id. Также модуль housekeeper производит чистку безнадёжных сообщений в таблице message.

Модуль worker посредством модуля tokens парсит входящее сообщение и формирует управляющую структуру Tokens. В случае получения какого-либо файла во входящем сообщении, модуль worker разбирает подпись к нему, определяет его тип, именует и сохраняет в соответствующей директории на сервере. В последующем полученные файлы могут быть отправлены адресату или проанализированы руководителем проекта при наличии соответствующих прав доступа.

После разбора сообщения и формирования управляющей структуры Tokens модуль worker передаёт её в модуль route, который проверяет наличие пользователя с указанным в Tokens telegram\_id в базе данных. В случае отсутствия такого пользователя происходит его регистрация (при установке соответствующего маркера) в базе данных и создание необходимых шаблонов, подписок, параметров telegram и т.д.

Далее из управляющей структуры извлекается параметр action (действие, выбранное пользователем), согласно которому запускается соответствующий обработчик (start, date, message, settings, help, time\_zone, reports). Управляющая структура Tokens передаётся в обработчик в качестве аргумента . По завершению своей работы каждый обработчик модифицирует Tokens согласно полученному результату (формирует текст сообщения, клавиатуры InlineKeyboard и/или ReplyKeyboard, сохраняет графические файлы и ссылки на них и т.д.) и возвращает Tokens, согласно которому модуль tokens формирует ответное сообщение и отправляет его в соответствующему пользователю.

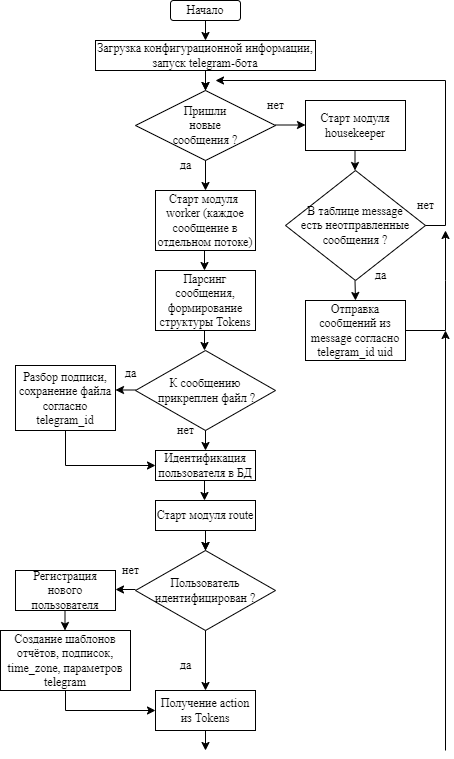


Рисунок ???. Алгоритм работы модуля Telegram-бот

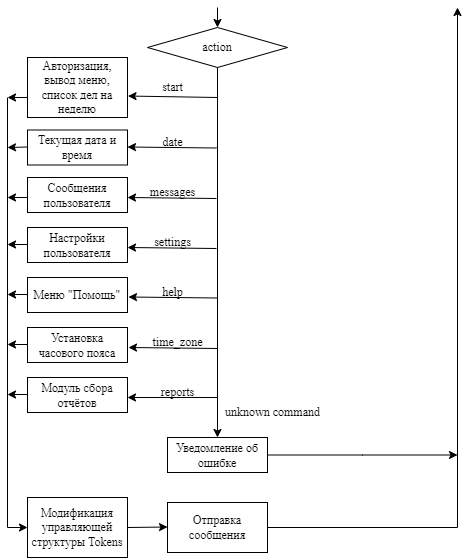


Рисунок ???. Алгоритм работы модуля Telegram-бот (продолжение)

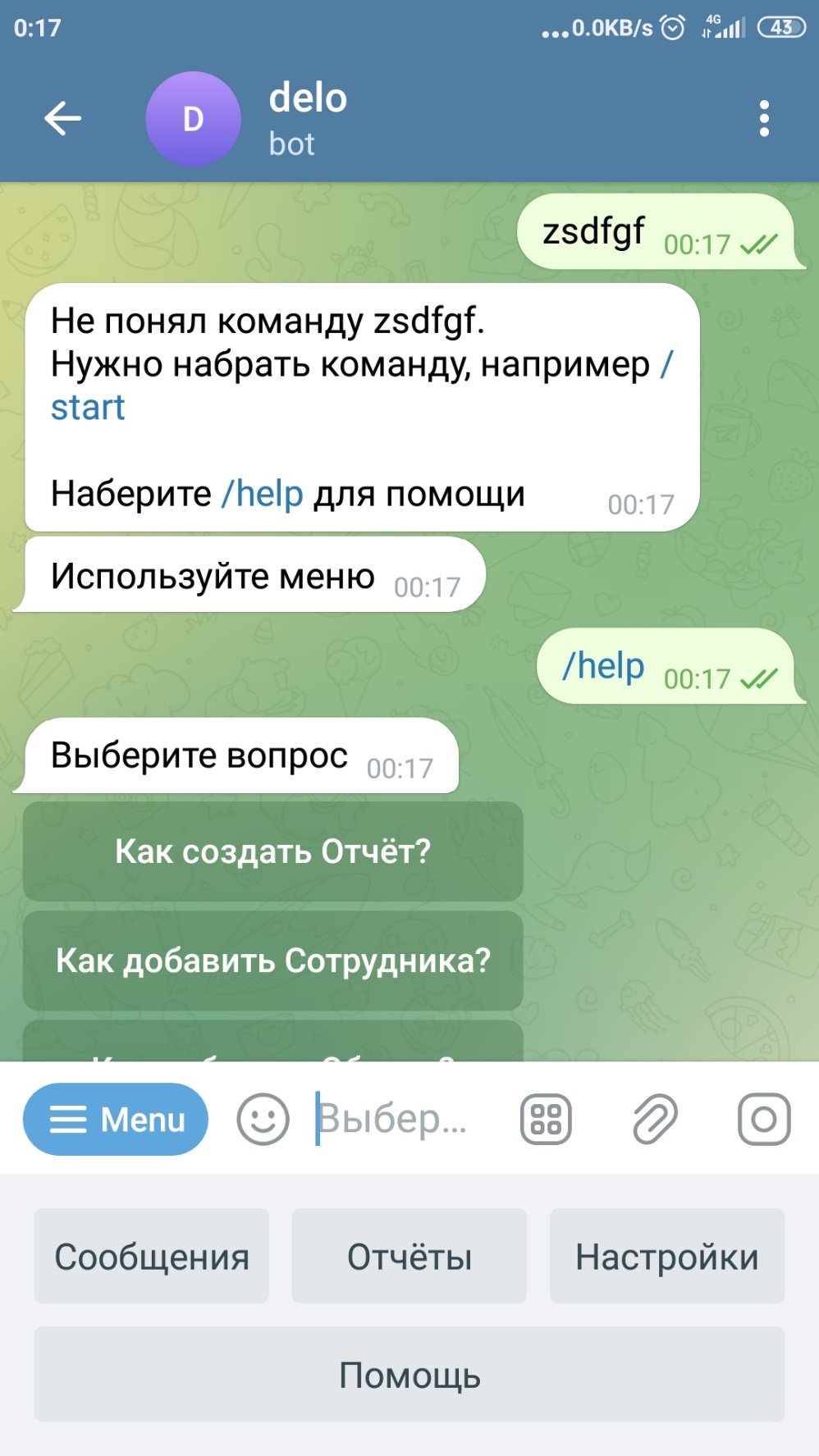


Рисунок ???. Меню Telegram-бот

Наиболее информативным для пользователя системы является модуль сбора отчётов, который запускается нажатием соответствующей кнопки ReplyKeyboard или ввода текстовой команды пользователем.

При первом вызове модуля сбора отчётов происходит формирование перечня отчётов для данного пользователя согласно шаблонам, записанным в базе данных. Формируется InlineKeyboard, в которой одна кнопка соответствует одному отчёту. InlineKeyboard размещается в управляющей структуре Tokens, согласно которой формируется ответное сообщение пользователю в виде набора кнопок.

При нажатии пользователем на одну из кнопок InlineKeyboard происходит считывание типа отчёта (list, status, line, brigadier) и вызов соответствующего модуля-обработчика. Аргументом обработчика передаётся управляющая структура Tokens.

Обработчик определяет период отчёта, отправляет запросы в базу данных, получает необходимую информацию, формирует ответ пользователю (InlineKeyboard со списком задач, сводную информацию в текстовом виде, график), создаёт преамбулу (составитель отчёта, период и т.д.), модифицирует и возвращает управляющую структуру Tokens.

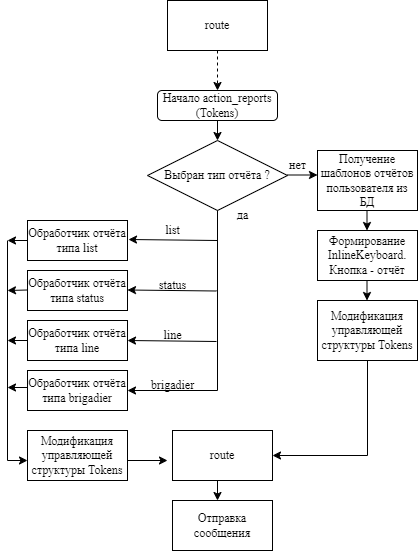


Рисунок ???. Алгоритм работы модуля сбора отчётов

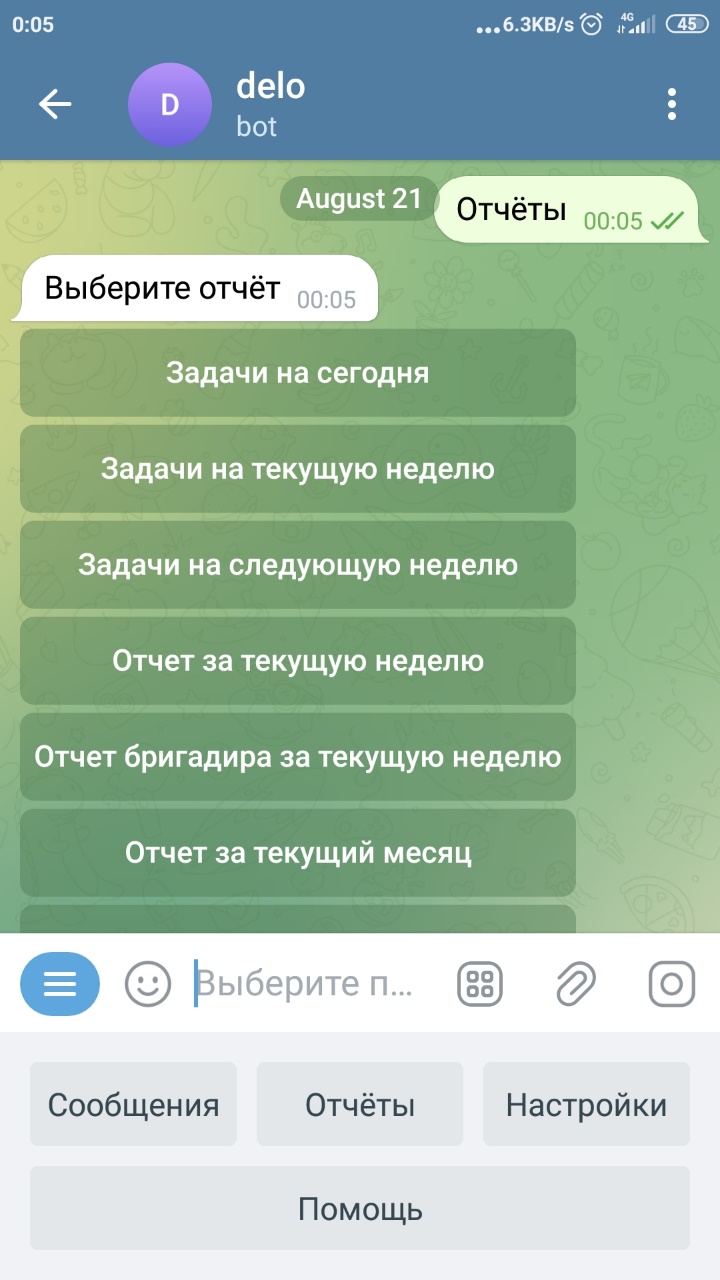


Рисунок ???. Меню «Отчёты»

3.4. Механизм запросов и ответов модуля Telegram-bot

Информационный обмен между модулями системы организован с использованием управляющей структуры Tokens. Каждый модуль системы получает Tokens в виде аргумента, модифицирует её в соответствии с результатом своей работы и возвращает обратно в вызывающий модуль. В конечном итоге модуль tokens на основе данных управляющей структуры Tokens формирует сообщение пользователю (текст, кнопки, файлы, изображения и т.д.), которое впоследствии отправляется в чат Telegram.

Управляющая структура Tokens представляет собой словарь Python:

Tokens = dict()  
Tokens['chat\_id'] – id chat telegram,Tokens['user'] = dict() – данные пользователя:

Tokens['user']['id'] – telegram user id,  
Tokens['user']['first\_name'] – имя пользователя,  
Tokens['user']['last\_name'] – фамилия пользователя,  
Tokens['user']['username'] – никнейм пользователя;

Tokens['request'] = dict() – запрос пользователя:

Tokens['request']['type'] - тип запроса (сообщение или callback\_query),  
Tokens['request']['id'] – message\_id,  
Tokens['request']['date'] – дата сообщения,  
Tokens['request']['text'] – текст сообщения,  
Tokens['request']['data'] – данные сообщения (в случае нажатия пользователем клавиши InlineKeyboard)

Tokens['answer'] = dict() – ответ на запрос пользователя:

Tokens['answer']['text'] – текст ответа,

Tokens['answer']['parse\_mode'] – режим вывода ответа (text / html),

Tokens['answer']['bottomKeys'] – меню ReplyKeyboard,

Tokens['answer']['img'] – ссылки на графические файлы для вывода,

Tokens['answer']['keys'] – список кнопок InlineKeyboard,

Tokens['answer']['currentkeyrow'] – ряд кнопок InlineKeyboard,

Tokens['answer']['keyboard'] – клавиатура (Inline / Reply).

3.5. Шаблоны отчётов

Для сбора и вывода пользователям информации из базы данных о задачах в системе реализовано хранение шаблонов отчётов. Каждая новая запись таблицы entity с типом subject (момент регистрации пользователя в системе) получает в поле parameters json объект, содержащий шаблоны отчётов. Каждый шаблон содержит информацию о работах (subject и group), типе отчёта, его названии, периоде сбора и варианте вывода.

"reports": [

{

"jobs": [

{

"subject": [

173

]

}

],

"type": "list",

"title": "Задачи на сегодня",

"period": "d",

"variant": "plan"

},

{

"jobs": [

{

"subject": [

173

]

}

],

"type": "list",

"title": "Задачи на текущую неделю",

"period": "w",

"variant": "plan"

},

{

"jobs": [

{

"subject": [

173

]

}

],

"type": "list",

"title": "Задачи на следующую неделю",

"period": "+w1",

"variant": "plan"

},

{

"jobs": [

{

"subject": [

173

]

}

],

"type": "status",

"title": "Отчет за текущую неделю",

"period": "w",

"variant": "all"

},

{

"jobs": [

{

"group": [

201

],

"subject": [

173

]

}

],

"type": "brigadier",

"title": "Отчет бригадира за текущую неделю",

"period": "w",

"variant": "all"

},

{

"jobs": [

{

"subject": [

173

]

}

],

"type": "status",

"title": "Отчет за текущий месяц",

"period": "m",

"variant": "all"

},

{

"jobs": [

{

"subject": [

173

]

}

],

"type": "line",

"title": "Отчет-график за текущий месяц",

"period": "m",

"variant": "all"

},

{

"jobs": [

{

"subject": [

173

]

}

],

"type": "status",

"title": "Отчет за прошлую неделю",

"period": "-w1",

"variant": "all"

},

{

"jobs": [

{

"subject": [

173

]

}

],

"type": "status",

"title": "Отчет за прошлый месяц",

"period": "-m1",

"variant": "all"

},

{

"jobs": [

{

"subject": [

173

]

}

],

"type": "line",

"title": "Отчет-график за прошлый месяц",

"period": "-m1",

"variant": "all"

},

{

"jobs": [

{

"group": [

3

]

}

],

"type": "bubble",

"title": "График бригадира за текущий месяц",

"period": "m",

"variant": "fact"

}

],

На основании значения ключа “type” обработчик action\_reports определяет тип отчёта, получает из поля parameters таблицы entity базы данных параметры запрашиваемого отчёта и вызывает модуль-обработчик соответствующего отчёта.

Всего по состоянию на конец августа 2023 года в системе предусмотрено четыре типа отчётов:

1. Отчёт типа list;
2. Отчёт типа status;
3. Отчёт типа line;
4. Отчёт типа brigadier.

3.5.1. Отчёт типа list

Отчёт типа list служит для сбора и отображения информации о назначенных и невыполненных на момент запроса задачах пользователя. Отчёт отображается в виде запрограммированных кнопок InlineKeyboard. Период сбора задач определяется параметром шаблона period (день, неделя, месяц и т.д.). Работники (группы работников), попадающие в выборку отчёта, определяются параметром отчёта jobs.

При нажатии на кнопку какой-либо работы выводится подробная информация и возможность отметить её выполнение, сообщить о проблеме и назначить другого работника (при наличии соответствующих прав у текущего пользователя).

|  |  |  |
| --- | --- | --- |
|  |  |  |

Рисунок ???. Варианты вывода отчёта типа list

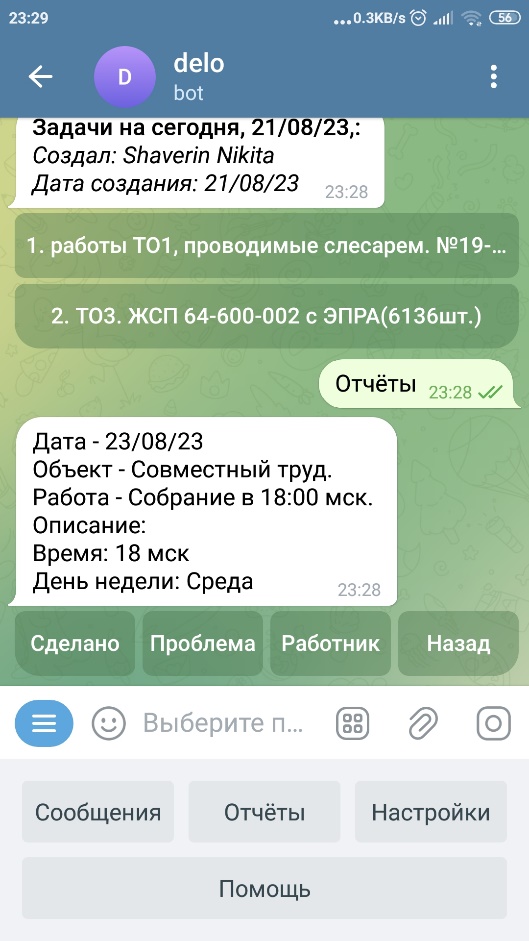


Рисунок ???. Информация о назначенной работе

3.5.2. Отчёт типа status

Отчёт типа status предназначен для сбора и отображения статистической информации о задачах и их статусах на определённый период времени. Отчёт отображается в текстовом режиме в виде таблицы. Параметры сбора отчёта определяются его шаблоном.

|  |  |
| --- | --- |
|  |  |

Рисунок ???. Варианты вывода отчёта типа list

3.5.3. Отчёт типа line

Описание отчёта, пример, скриншоты.

3.5.4. Отчёт типа brigadier

Отчёт бригадира – разновидность отчёта типа list

Отчёт bubble – графическое представление отчёта бригадира

3.6. Обмен сообщениями между пользователями системы

Из отчёта list

Сделано, проблема, работник м.б. что-то ещё

3.7. Автоматическая рассылка задач

Week\_tasks, cron

4. Заключение

4.1. Выводы, полученные во время анализа теоретической базы и практического исследования

41

4.2. Оценка проекта, описание его результатов

42

4.3. Общий итог работы, её практическая значимость

43

4.4. Предложения по совершенствованию информационной системы «Совместный труд»

Бизнес-цели

Вывести новый продукт на рынок - привлечь к тестированию 3 компании.

Закрепиться на рынке - найти заказчика, готового платить за развитие проекта.

Положительные моменты

- сокращение на рынке предметного софта

- быстрое внедрение. Без необходимости, что-то скачивать, обновлять и проходить обучение

- базовые функции бесплатные

- без разработки дорогих (долгих по времени) функций: планирования, отслеживания связей, учёт рабочих дней и пр.

В следующих релизах использовать объектно-ориентированный подход программирования (значения в Tokens, обработка запросов от пользователя и др.)

Реализовать информационный обмен с системой 1С

5. Список используемой литературы

1. <https://кц-прогресс.рф/history.html>
2. <https://bitcop.ru/blog/kontrol-vypolnenija-zadach-sotrudnikami>
3. <https://sisu.ut.ee/basic_of_project_work/>
4. <https://www.redmine.org/>
5. <http://jira.ru/>
6. <https://www.bitrix24.ru/>
7. <https://subscribe.kck.ru/#/st>
8. <https://a2b.su/>
9. <https://aspro.cloud/>
10. <https://gb.ru/blog/arhitektuta-sistemy/>
11. Рогозов Ю.И., Свиридов А.С., Кучеров С.А. Архитектура информационных систем: учебное пособие. – Ростов-на-Дону: Изд-во ЮФУ, 2014. – 117 c.
12. <https://spravochnick.ru/bazy_dannyh/bazy_dannyh_vvedenie/arhitektura_informacionnoy_sistemy/>
13. <https://planny24.ru/>
14. <https://megaplan.ru/>
15. <https://www.atlassian.com/ru/agile/scrum>
16. <https://www.atlassian.com/ru/agile/kanban>
17. <https://pypi.org/project/pyTelegramBotAPI/>
18. <https://core.telegram.org/bots/api>
19. Литература2

6. Приложения.

6.1. Программный код модуля Telegram-bot.

Код модулей

Код надо причесать, добавить комментарии

6.2. Скриншоты работы ИС.