МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«Санкт-Петербургский государственный университет аэрокосмического приборостроения»

ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

КУРСОВОЙ ПРОЕКТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Преподаватель |  | 05.07.2023 |  | И.Л. Рохманько |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЁТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКИ  В СОСТАВЕ ПРОФЕССИОНАЛЬНОГО МОДУЛЯ  ПМ.04 Разработка, администрирование и защита баз данных  *код и наименование профессионального модуля* |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | С021к |  | 05.07.2023 |  | В.Д. Панков |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2023

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

на прохождение производственной практики обучающегося по специальности 09.02.07 «Информационные системы и программирование»

1. Фамилия, имя, отчество обучающегося: Панков Василий Дмитриевич
2. Группа: С021К Сроки проведения практики: с « 15 » июня 2023г. по « 05 » июля 2023г.
3. Тема задания: приобретение профессиональных умений и опыта практической деятельности, закрепление сформированных компетенций по профессиональному модулю

ПМ.04 РАЗРАБОТКА, АДМИНИСТРИРОВАНИЕ И ЗАЩИТА БАЗ ДАННЫХ

1. Вопросы, подлежащие изучению:

* Обследование предметной области . Проектирование логической схемы базы данных с использованием современных case-средств;
* Обоснование выбора СУБД. Построение схемы базы данных в конкретной СУБД. Создание хранимых процедур и триггеров;
* Администрирование БД;
* Обеспечение информационной безопасности на уровне базы данных;
* Разработка пользовательского приложения для работы с базой данных;
* Установка базы данных на сервер.

1. Выполнение комплексных работ: Разработка, администрирование и защита базы данных учёта коллекционных записей экспедиционных сборов млекопитающих
2. Содержание отчетной документации:
   1. Отчёт, включающий в себя:

* титульный лист;
* индивидуальное задание;
* материалы о выполнении индивидуального задания (содержание определяется цикловой комиссией);
* список использованных источников.
  1. Аттестационный лист.

1. Срок представления отчета заместителю декана по учебно–производственной работе: «\_05\_\_»\_\_\_июля\_\_\_\_\_2022\_\_ г.

Руководитель практики от факультета СПО:

преподаватель 15.06.23 И.Л. Рохманько

должность, уч. степень, звание подпись, дата инициалы, фамилия

СОГЛАСОВАНО

Руководитель практики от организации:

заведующий лабораторией, ведущий научный сотрудник 15.05.23 Н.И. Абрамсон

должность подпись, дата инициалы, фамилия

Задание принял к исполнению:

Обучающийся

15.06.23 В.Д. Панков

дата подпись инициалы, фамилия

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc139280545)

[1 Анализ предметной области 5](#_Toc139280546)

[1.1 Общая характеристика объекта исследования 5](#_Toc139280547)

[1.2 Постановка и развернутое описание задачи 6](#_Toc139280548)

[1.3 Средства решения 7](#_Toc139280549)

[2 Разработка и реализация проекта базы данных 12](#_Toc139280550)

[2.1 Диаграмма прецедентов 12](#_Toc139280551)

[2.2 Определение логической структуры реляционной базы данных 13](#_Toc139280552)

[2.3 Разработанные хранимые процедуры, триггеры и представления 4](#_Toc139280553)

[2.4 Администрирование 6](#_Toc139280554)

[2.5 Реализация защиты базы данных 15](#_Toc139280555)

[2.6 Результат выполнения процедур 16](#_Toc139280556)

[3 Разработка клиентского приложения 21](#_Toc139280557)

[ЗАКЛЮЧЕНИЕ 27](#_Toc139280558)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 28](#_Toc139280559)

[ПРИЛОЖЕНИЕ А Созданные таблицы БД 30](#_Toc139280560)

[ПРИЛОЖЕНИЕ Б Разработанные хранимые процедуры, триггеры и представления 50](#_Toc139280561)

[ПРИЛОЖЕНИЕ В Листинг клиентского приложения 64](#_Toc139280562)

[ПРИЛОЖЕНИЕ Г Листинг выдачи прав пользователям 141](#_Toc139280563)

[ПРИЛОЖЕНИЕ Д Листинг конфигурационного файла 143](#_Toc139280564)

# ВВЕДЕНИЕ

Информационная система — это совокупность программных и аппаратных средств, предназначенных для сбора, хранения, обработки, анализа и распространения информации в организации или предприятии.

Существует множество лабораторией во всём мире, которые занимаются различными исследованиями с животными. Данную производственную практику я проходил в «Лаборатории геномики и палеогеномики» при «Государственном зоологическом институте Российской Федерации». Данная лаборатория занимается изучением геномов существующих и исчезнувших видов млекопитающих.

Целью моей работы является проектирование информационной системы, которая облегчит процесс учета коллекционных записей экспедиционных сборов млекопитающих, а также обеспечит возможность анализа полученной информации для научных исследований в данной области.

Для достижения поставленной цели необходимо решить следующие задачи:

* изучить методы и технологии учета и хранения информации о коллекционных записях экспедиционных сборов млекопитающих;
* определить функциональные требования к информационной системе;
* разработать структуру базы данных для хранения информации о коллекционных записях;
* разработать интерфейс для ввода и обработки данных;
* реализовать информационную систему на основе выбранных методов и технологий;

# Анализ предметной области

## Общая характеристика объекта исследования

В данной задаче практики предметной областью является учёт коллекционных записей экспедиционных сборов млекопитающих, а в качестве внешних пользователей могут выступать:

* заведующий лабораторией – лицо, которое может редактировать информацию о сотрудниках, редактировать топологию, а также выдавать доступ новым сотрудникам и менять им пароли;
* научный сотрудник – лицо, будет иметь доступ к просмотру всех коллекционных записей, имеет возможности редактирование и добавление записей и отправку запросов на их удаление, также пользователь может сменить свой пароль.
* гость – группа, пользователей, которая может только просматривать данные.

Научные экспедиции по изучению млекопитающих проходят в различных уголках мира, и собранные в ходе них образцы требуют качественного учёта и обработки. В информационной системе будут храниться данные о месте, времени сбора, полного названия млекопитающего, которое будет включать отряд, семейство, род и вид, его генетические характеристики.

Также некоторые экземпляры могут передавать в другие лаборатории или музеи, для их показа или исследования и иметь там свой уникальный номер.

В самой лаборатории используются собственная идентификация элементов, имеются комментарии к месту сбора и к самой записи.

## Постановка и развернутое описание задачи

Информационная система должна обеспечивать следующие функции:

* ввод, просмотр редактирование информацию о записи, авторе сборного материала, дате сбора, месте сбора, отряде, семействе, роде виде, полу, возрасту и выделена ли RNA;
* выполнять запросы для автоматизации (просмотр по месту сбора, фильтрация по множеству параметров: по автору сборного материала, промежутке сбора, месту сбора, по таксономии, полу, возрасту и выделена ли RNA;);
* возможность хранение ссылок на файлы c полногеномными данными;
* обеспечение быстрого добавления записей (указание точки на карте, выбор сборщика из списка, генерация идентификатора);
* авторизация пользователей и сохранение информации об внесённых ими изменений;
* возможность редактирования древовидной структуры таксономии.

Входные данные для проектирования информационной системы:

* информация о сборе: место сбора, сборщик материала, дата сбора, RNA, внутренний номер в лаборатории, отряд, семейство, род, вид и пол млекопитающего данных сбора.

Выходные данные: в качестве выходных данных будут результаты выполнения запросов.

Требования, предъявляемые к разрабатываемой программе:

Программа должна быть надёжна, простота в использовании и обладать достаточно высоким быстродействием.

Пользовательский интерфейс должен быть интуитивно понятен любому пользователю. В программе должна быть предусмотрена возможность корректировки базы данных.

## Средства решения

Для решения поставленной задачи были выбраны следующие средства:

Dart – объектно-ориентированный язык программированный, разработанный Google. Используется для создания быстрых и эффективных приложений на разных платформах. Одной из важных особенностей данного языка является его способность к компиляции в эффективный нативный код. Это позволяет создавать быстрые и производительные приложения, которые могут выполняться без промежуточной интерпретации. [3]

Material You – это эволюция концепции Material Design, разработанная Google. Он представляет собой дизайн-язык, который ставит на передний план персонализацию и индивидуальность пользователей.

Основные преимущества Material You:

1. персонализация: Material You позволяет пользователям настраивать визуальный стиль своих устройств и приложений, включая цвета, шрифты и стили. Это создает более индивидуальный и удовлетворяющий опыт использования.
2. адаптивность: Material You обеспечивает адаптивный интерфейс, который подстраивается под различные размеры экранов и контексты использования. Приложения выглядят и работают хорошо на разных устройствах.
3. системные возможности: Material You интегрируется с функциями операционной системы и платформы, позволяя использовать новые возможности и API для создания более глубокого пользовательского опыта.
4. совместимость с Material Design: Material You соответствует принципам Material Design, обеспечивая согласованный визуальный стиль и поведение приложений на разных устройствах и платформах.
5. поддержка разработчиков: Material You получил широкую поддержку разработчиков и сообщества. Google предоставляет ресурсы, инструменты и руководства для помощи разработчикам внедрить Material You в свои приложения. [7]

Flutter – это открытый кроссплатформенный фреймворк разработки мобильных приложений, созданный Google. Он позволяет создавать высокопроизводительные приложения для iOS, Android, веб и настольных компьютеров с использованием общего кода на языке программирования Dart. Flutter отличается своим собственным движком рендеринга, обеспечивающим быструю и плавную визуализацию. Он предлагает готовые виджеты и инструменты для удобной разработки пользовательского интерфейса. Благодаря функции горячей перезагрузки, изменения в коде отображаются мгновенно, что ускоряет разработку и упрощает тестирование приложений. Flutter также поддерживает пользовательские анимации, интеграцию с нативными функциями устройства и управление состоянием приложения.

Основные принципы Flutter:

* кроссплатформенность: Flutter позволяет разрабатывать приложения для разных платформ, таких как iOS, Android, веб и настольные компьютеры, с использованием общего кода. Это позволяет сократить время разработки и упростить поддержку приложений на разных платформах;
* горячая перезагрузка: Горячая перезагрузка (Hot Reload) в Flutter позволяет мгновенно видеть изменения в коде и мгновенно обновлять интерфейс приложения без перезапуска. Это значительно ускоряет процесс разработки и позволяет разработчикам быстро итерироваться над интерфейсом и функциональностью приложения;
* однородность интерфейса: Все элементы интерфейса в Flutter являются виджетами, включая даже текст, отступы и цвета. Это создает единообразие во всем интерфейсе приложения и облегчает его масштабирование, адаптацию и изменение визуального стиля;
* производительность: Flutter обеспечивает высокую производительность и плавность работы приложений благодаря использованию собственного движка рендеринга. Он позволяет создавать быстрые и отзывчивые пользовательские интерфейсы, а также оптимизирует использование ресурсов устройства;
* открытость и активное сообщество: Flutter является открытым проектом, и у него активное сообщество разработчиков. Это обеспечивает доступ к большому количеству инструментов, библиотек и пакетов, которые расширяют возможности разработки и помогают решать разнообразные задачи;
* Material You дизайн: Flutter тесно интегрирован с Material You дизайном, стандартным дизайн-языком Google. Это позволяет разработчикам создавать приложения с современным и стильным внешним видом, соответствующим рекомендациям Google по пользовательскому опыту. [4]

Syncfusion DataGrid for Flutter – гибкий компонент пользовательского интерфейса для отображения и редактирования табличных данных в приложениях Flutter. [5]

PostgreSQL– это мощная, открытая реляционная система управления базами данных. PostgreSQL предлагает расширенный набор функций и возможностей, обеспечивая надежное хранение и эффективное управление данными.

Основные характеристики PostgreSQL:

* реляционная модель данных: PostgreSQL основан на реляционной модели данных, что позволяет организовывать данные в таблицы с определенными структурами и связями. Это обеспечивает структурированное хранение информации и возможность выполнения сложных запросов и аналитических операций;
* масштабируемость и производительность: PostgreSQL способен обрабатывать большие объемы данных и обеспечивать высокую производительность при выполнении запросов. Он поддерживает параллельную обработку, индексы, оптимизацию запросов и другие механизмы для оптимизации производительности;
* поддержка стандартов SQL: PostgreSQL полностью совместим с ANSI SQL и поддерживает широкий набор стандартных операторов, функций и запросов. Это облегчает разработку и переносимость приложений, использующих PostgreSQL;
* расширяемость и гибкость: PostgreSQL предлагает механизмы для создания пользовательских типов данных, функций и расширений. Это позволяет разработчикам расширять функциональность СУБД и адаптировать ее под уникальные требования проекта;
* транзакционная безопасность: PostgreSQL обеспечивает транзакционную безопасность, поддерживая ACID-свойства (атомарность, согласованность, изолированность, долговечность). Это гарантирует целостность данных и защиту от потери информации в случае сбоев или ошибок;
* многоязыковая поддержка: PostgreSQL поддерживает множество языков программирования, включая Python, Java, C/C++, Ruby и многие другие. Это облегчает интеграцию СУБД с различными приложениями и разработку на разных платформах; [2]

RESTful API (Representational State Transfer) – это архитектурный стиль для проектирования и взаимодействия сетевых приложений, основанный на принципах REST. Он обеспечивает стандартизированный способ обмена данными между клиентом и сервером с использованием HTTP-протокола.

PostgREST – это программное обеспечение с открытым исходным кодом, которое предоставляет простой и удобный RESTful интерфейс для работы с базой данных PostgreSQL. Он позволяет вам взаимодействовать с базой данных через стандартные HTTP-методы, такие как GET, POST, PUT и DELETE, и автоматически генерирует API-эндпоинты для каждой таблицы в базе данных. PostgREST обеспечивает функции фильтрации, сортировки, пагинации и аутентификации данных, а также поддерживает вложенные запросы и пользовательские функции. Использование PostgREST значительно упрощает разработку и создание API на основе PostgreSQL, обеспечивая гибкость и масштабируемость веб-приложений и микросервисов. [2]

Nominatim (от латинского «by name») – это инструмент для поиска данных OSM по имени и адресу и для создания синтетических адресов точек OSM (обратное геокодирование). [9]

# Разработка и реализация проекта базы данных

## Диаграмма прецедентов

Диаграмма вариантов использования – диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей. Диаграмма use-case для проекта представлена на рисунке 1.

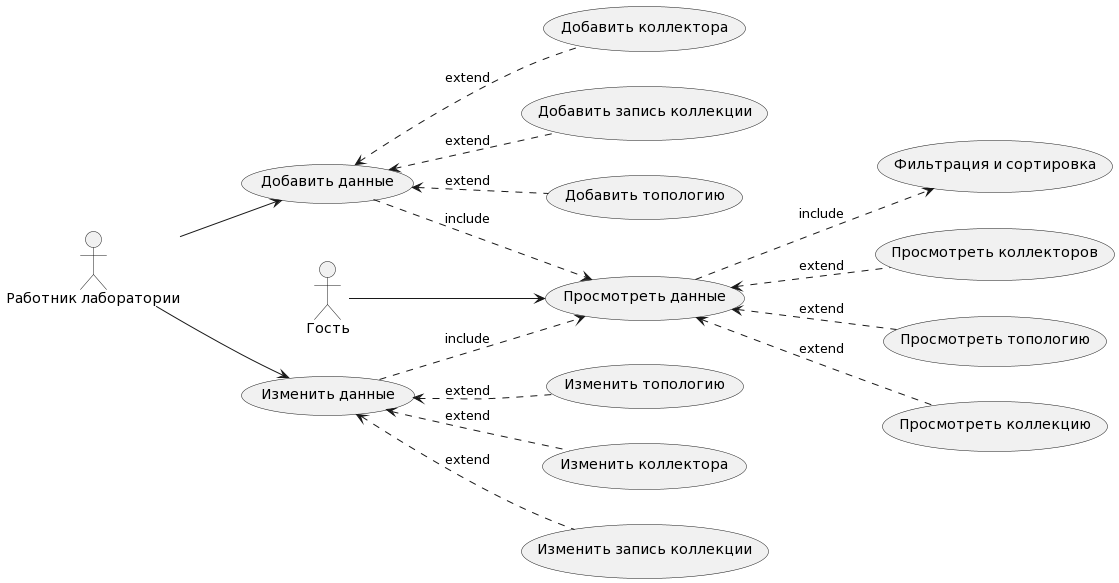


Рисунок 1 – Таблица users

## Определение логической структуры реляционной базы данных

Даталогическая структура реляционной базы данных определяется совокупностью логически связанных реляционных таблиц.

Логические связи соответствуют структурным связям между объектами в концептуальной модели, каждый объект в логической модели отображается соответствующей реляционной таблицей.

Связи между таблицами осуществляются посредством общих атрибутов. Реляционная модель проектируемой базы данных.

Таблица 1 содержит информацию о возрастах.

Таблица 1 – Описание логической структуры таблицы «age»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK | id | Числовой | - | NOT NULL |
| - | name | Символьный (переменный) | 20 | - |

Таблица 2 содержит информацию о возрастах.

Таблица 2 – Описание логической структуры таблицы «collection»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| PK | id | Числовой | - | NOT NULL |
| - | CatalogueNumber | Текстовый | - | - |
| - | collect\_id | Текстовый | - | - |
| FK | kind\_id | Числовой | - | NOT NULL |
| FK | subregion\_id | Числовой | - | NOT NULL |
| - | gen\_bank\_id | Символьный (переменный) | 20 | - |
| - | point | География (точка) | - | - |
| FK | vouch\_inst\_id | Числовой | - | - |
| Продолжение таблицы 2 | | | | |
| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| - | rna | Логический | - | DEFAULT false NOT NULL |
| FK | sex\_id | Числовой | - | - |
| FK | age\_id | Числовой | - | - |
| - | day | Числовой | - | - |
| - | month | Числовой | - | - |
| - | year | Числовой | - | NOT NULL |
| - | comment | Текстовый | - | - |
| - | geo\_comment | Текстовый | - | - |
| - | file\_url | Текстовый | - | - |

Таблица 3 содержит информацию о коллекторах.

Таблица 3 – Описание логической структуры таблицы «collector»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK | id | Числовой | - | NOT NULL |
| - | last\_name | Символьный (переменный) | 100 | - |
| - | first\_name | Символьный (переменный) | 100 | - |
| - | second\_name | Символьный (переменный) | 100 | - |

Таблица 4, является смежной таблицей, которая содержит информацию о соотношении коллекторов к коллекции.

Таблица 4 – Описание логической структуры таблицы «collector\_to\_collection»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK, FK | collector\_id | Числовой | - | NOT NULL |
| PK, FK | collection\_id | Числовой | - | NOT NULL |

Таблица 5 содержит информацию о странах.

Таблица 5 – Описание логической структуры таблицы «country»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK | id | Числовой | - | NOT NULL |
| - | name | Текстовый | - | NOT NULL |

Таблица 6 содержит информацию о регионах.

Таблица 6 – Описание логической структуры таблицы «region»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK | id | Числовой | - | NOT NULL |
| FK | country\_id | Числовой | - | NOT NULL |
| - | name | Текстовый | - |  |

Таблица 7 содержит информацию о субрегионах.

Таблица 7 – Описание логической структуры таблицы «subregion»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| PK | id | Числовой | - | NOT NULL |
| FK | region\_id | Числовой | - | NOT NULL |
| - | name | Текстовый | - |  |

Таблица 8 содержит информацию об отрядах.

Таблица 8 – Описание логической структуры таблицы «order»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK | id | Числовой | - | NOT NULL |
| - | name | Текстовый | - |  |

Таблица 9 содержит информацию о семействах.

Таблица 9 – Описание логической структуры таблицы «family»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK | id | Числовой | - | NOT NULL |
| FK | order\_id | Числовой | - | NOT NULL |
| - | name | Текстовый | - |  |

Таблица 10 содержит информацию о родах.

Таблица 10 – Описание логической структуры таблицы «genus»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK | id | Числовой | - | NOT NULL |
| FK | family\_id | Числовой | - | NOT NULL |
| - | name | Текстовый | - |  |

Таблица 11 содержит информацию о видах.

Таблица 11 – Описание логической структуры таблицы «kind»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK | id | Числовой | - | NOT NULL |
| FK | genus\_id | Числовой | - | NOT NULL |
| - | name | Текстовый | - |  |

Таблица 12 содержит информацию о полах.

Таблица 12 – Описание логической структуры таблицы «sex»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK | id | Числовой | - | NOT NULL |
| - | name | Символьный (переменный) | 40 | NOT NULL |

Таблица 13 содержит информацию о ваучерных институтах.

Таблица 13 – Описание логической структуры таблицы «voucher\_institute»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK | id | Числовой | - | NOT NULL |
| - | name | Текстовый | - | NOT NULL |

На рисунке 2 показана схема связей данных.

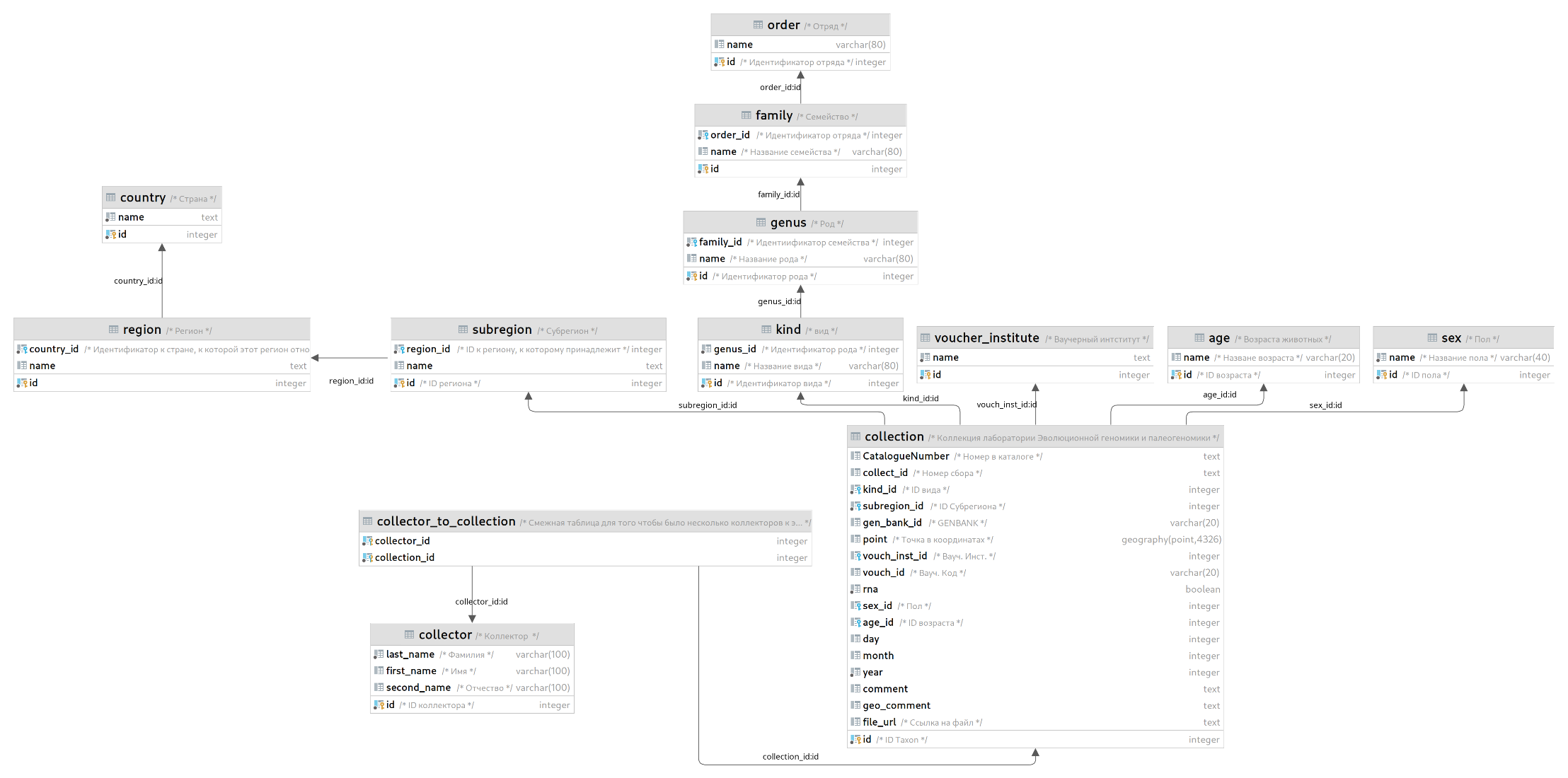


Рисунок 2 – Схема связей данных в СУБД PostgreSQL

## Разработанные хранимые процедуры, триггеры и представления

В таблице 15 представлен список хранимых процедур и их описание.

Таблица 14 – Список хранимых процедур

| Название | Описание |
| --- | --- |
| get\_age\_id | Возвращает идентификатор возраста по его названию |
| get\_sex\_id | Возвращает идентификатор пола по его названию |
| get\_order\_id | Возвращает идентификатор порядка по его названию |
| get\_family\_id | Возвращает идентификатор семейства по его названию и порядку |
| get\_genus\_id | Возвращает идентификатор рода по его названию и семейству |
| get\_kind\_id | Возвращает идентификатор вида по его названию и роду |
| get\_country\_id | Возвращает идентификатор страны по её названию |
| get\_region\_id | Возвращает идентификатор региона по его названию и стране |
| get\_subregion\_id | Возвращает идентификатор подрегиона по его названию и региону |
| get\_vouch\_inst\_id | Возвращает идентификатор института-владельца ваучера по его названию |
| get\_collector\_id | Возвращает идентификатор коллекционера по его фамилии, имени и отчеству |
| add\_collection | Добавляет запись о коллекции в базу данных |
| remove\_collection\_by\_id | Удаляет запись о коллекции по её идентификатору |
| update\_collection\_by\_id | Обновляет запись о коллекции по её идентификатору |
| auth.login | Проверяет логин и пароль, возвращает JWT-токен в случае успешной аутентификации |
| auth.get\_user\_info | Возвращает информацию о пользователе (логин, URL аватара, роль) |
| auth.user\_role | Получение роли пользователя по его логину и паролю |
| check\_auth | Проверка авторизации пользователя |
| add\_topology | Добавление топологии, по частичному указанию, отряда, семейства, рода, вида. |

Листинг разработанных хранимых процедур находится в Приложении Б.

В таблице 16 представлен список триггеров и их описание.

Таблица 15 – Список триггеров

| Название | Описание |
| --- | --- |
| ensure\_user\_role\_exists | Проверяет, существует ли указанная роль в базе данных. Если роль не существует, генерирует ошибку. |
| encrypt\_pass | Хеширует пароль перед сохранением в таблице auth.users. |

Листинг разработанных триггеров находится в Приложении Б.

В таблице 17 представлен список представлений и их описание.

Таблица 16 – Список представлений

|  |  |
| --- | --- |
| Название | Описание |
| basic\_view | Соединяет все таблицы из схемы public и показывают коллекцию, в удобном виде для просмотра и анализа данных. |

Листинг разработанных представлений находится в Приложении Б.

## Администрирование

В базе данных были созданы следующие группы пользователей:

* web\_​anon - гость, который может просматривать все таблицы в публичном доступе (scheme public):
  + collection;
  + age;
  + family;
  + sex;
  + country;
  + voucher\_​institute;
  + region;
  + genus;
  + collector\_​to\_​collection;
  + order;
  + collector;
  + subregion;
  + kind;
  + collection.
* lab\_worker - работник лаборатории, имеет те же привелегии, что и web\_anon, но также может добавлять и изменять любые таблицы в публичном доступе (scheme public), а также функции добавления и изменения записей в таблице collection;
* head\_​lab - заведующий лабораторией, имеет те же привелегии, что и lab\_​worker, но также может выполнять все функции в публичном доступе (scheme public) и все процедуры в авторизации (scheme auth), что позволяет ему:
  + добавлять пользователя;
  + удалять пользователя;
  + изменять пользователя.
* database\_​admin - администратор базы данных обладает всеми привилегиями на обе схемы (auth и public).
* Дополнительная роль аутентификатор (authenticator), у этой роли есть права всех пользователей выше, благодаря этому модуль postgrest реализовывает распределение ролями.

В приложении Г, представлен код выдачи прав всем пользователям выше.

### Авторизация

Для авторизации была реализована отдельная схема auth.

Таблица 14 содержит информацию о пользователях.

Таблица 17 – Описание логической структуры таблицы «users»

| Признак ключа | Имя поля | Тип данных поля | Длина | Ограничения |
| --- | --- | --- | --- | --- |
| PK | login | Текстовый | - | NOT NULL |
| - | pass | Текстовый | <512 | NOT NULL |
| - | role | name(Имя роли или пользователя) | <512 | NOT NULL |
| - | avatar | Текстовый | - | - |

Хранимые процедуры, функции и триггеры данной схемы приведены в соответствующем разделе «Разработанные хранимые процедуры, триггеры и представления», а их код приведён в приложении Б.

На рисунке 3, представлено демонстрируются тестовые данные в таблице user и вид хешированное пароля.

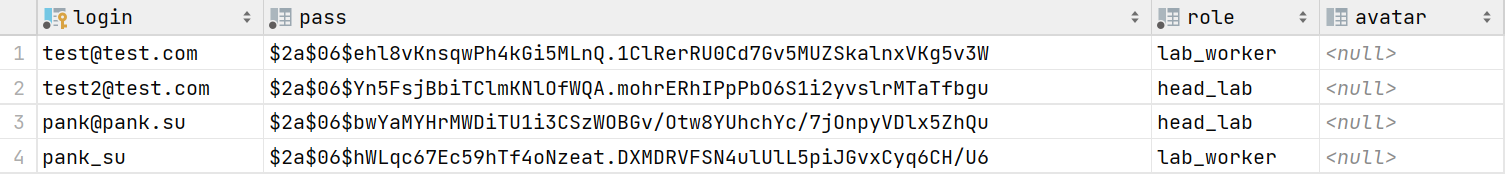


Рисунок 3 – Таблица users

Авторизация пользователя происходит через RESTful API, созданного с помощью приложения PostgREST, конфигурационный файл приведён в приложении Д.

### Демонстрация работы

В рамках демонстрации ограничений возможностей пользователей разных групп будет произведено наглядное представление с использованием HTTP запросов к серверу и соответствующих ответов. Кроме того, будет представлена декодировка HTTP запросов в SQL для полного понимания процесса.

Демонстрация со стороны анонимного пользователя – роль web\_anon.

Получение записи по id=1 в таблице «collection».

HTTP запрос:

GET http://localhost:3000/collection?id=eq.1

Аналогичный SQL запрос:

SELECT \* FROM collection WHERE id = 1;

Ответ:

[

{

"id": 1,

"CatalogueNumber": "ZIN-TER-M-55",

"collect\_id": "1",

"kind\_id": 1,

"subregion\_id": 1,

"gen\_bank\_id": null,

"point": "0101000020E6100000CACDE459D130554008D8BBFD41264940",

"vouch\_inst\_id": 1,

"vouch\_id": "91130",

"rna": false,

"sex\_id": 2,

"age\_id": 3,

"day": 12,

"month": 6,

"year": 2005,

"comment": "По сиквенсам со1 и цитохрома - красно-серая, первоначально определена как красная JF713496",

"geo\_comment": "Алтай, Республика, Усть-Коксинский р-он, Банное, село, р. Колчулу"

}

]

Попытка добавить запись в таблицу «age».

HTTP запрос:

POST http://localhost:3000/age

Content-Type: application/json

{"name": "test age"}

Аналогичный SQL запрос:

INSERT INTO age VALUES ('test age')

Ответ:

{

"code": "42501",

"details": null,

"hint": null,

"message": "нет доступа к таблице age"

}

Демонстрация, что таблица age доступна:

HTTP запрос:

GET http://localhost:3000/age

Аналогичный SQL запрос:

SELECT \* FROM age

Ответ:

[

{

"id": 0,

"name": "Unknown"

},

{

"id": 2,

"name": "subadult"

},

{

"id": 3,

"name": "adult"

},

{

"id": 1,

"name": "juvenile"

},

{

"id": 4,

"name": "subadult or adult"

},

{

"id": 19,

"name": "test"

}

]

Демонстрация со стороны работника лаборатории – роль lab\_worker.

Авторизация:

HTTP запрос:

POST http://localhost:3000/rpc/login

Content-Type: application/json

{"email": "test@test.com", "pass":"test"}

Аналогичный SQL запрос:

SELECT login(‘test@test.com’, ‘test’)

Ответ, при верном пароле:

{

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjoibGFiX3dvcmtlciIsImVtYWlsIjoidGVzdEB0ZXN0LmNvbSIsImV4cCI6MTY4Mjc3MjEyN30.cI4CN4iIeOPw-jsMLSLF3NrtDPSYsCfm9M1EPbaqHbw"

}

Ответ, при неверном пароле или логине:

{

"code": "28P01",

"details": null,

"hint": null,

"message": "invalid user or password"

}

Добавление записи в коллекцию:

HTTP запрос:

POST http://localhost:3000/rpc/add\_collection

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjoibGFiX3dvcmtlciIsImVtYWlsIjoidGVzdEB0ZXN0LmNvbSIsImV4cCI6MTY4Mjk1MzU3MX0.IKV4EPcylqRBEoumGboCXfegJJ9bMHsyXngEy1I84I4

Content-Type: application/json

{ "catalog\_number": "test", "collect\_id": "test", "order" : "test", "family" : "test",

"genus" : "test", "kind" : "test", "age" : "test", "sex" : "test", "vauch\_inst" : "test",

"vauch\_id" : "test", "point" : "POINT(0 0)", "country" : "test", "region" : "test", "subregion" : "test",

"geocomment": "test", "date\_collect" : "2004-4-4", "comment" : "Это тестовая запись",

"collectors" : "{{\"Панков\", \"\", \"\" }, {\"Турсунова\", \"\", \"\"}}"}

Аналогичный SQL запрос:

SELECT add\_collection(‘test’, ‘test’, ‘test’, ‘test’, ‘test’, ’test’, ‘POINT(0 0)’, ‘test’, ‘test’, ‘test’, ‘test’, ‘2004-4-4’, ‘Это тестовая запись’, ‘"{{"Панков", "", "" }, {"Турсунова", "", ""}}’)

Ответ пустой.

Проверка, что запись добавилась:

HTTP запрос:

GET http://localhost:3000/collection?id=eq.6081

Аналогичный SQL запрос:

SELECT \* FROM collection WHERE id = 6081

Ответ:

[

{

"id": 6081,

"CatalogueNumber": "test",

"collect\_id": "test",

"kind\_id": 270,

"subregion\_id": 154,

"gen\_bank\_id": null,

"point": "0101000020E610000000000000000000000000000000000000",

"vouch\_inst\_id": 46,

"vouch\_id": "test",

"rna": false,

"sex\_id": 18,

"age\_id": 19,

"day": 4,

"month": 4,

"year": 2004,

"comment": "Это тестовая запись",

"geo\_comment": "test"

}

]

Попытка удаления записи:

HTTP запрос:

POST http://localhost:3000/rpc/remove\_collection\_by\_id

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjoibGFiX3dvcmtlciIsImVtYWlsIjoidGVzdEB0ZXN0LmNvbSIsImV4cCI6MTY4Mjk1NzI1N30.YaqxuxgQeeZPTQBppSYg154FDUPlrTh8WhgDvgEd5sQ

Content-Type: application/json

{"col\_id": 6081}

Аналогичный SQL запрос:

SELECT remove\_collection\_by\_id(6081)

Ответ:

{

"code": "42501",

"details": null,

"hint": null,

"message": "нет доступа к таблице collection"

}

Демонстрация со стороны заведующего лабораторией – роль head\_lab.

Попытка удаления:

HTTP запрос:

POST http://localhost:3000/rpc/remove\_collection\_by\_id

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjoiaGVhZF9sYWIiLCJlbWFpbCI6InRlc3QyQHRlc3QuY29tIiwiZXhwIjoxNjgyOTU4MjA3fQ.mkzS0XwjVWW\_yook7JUPx3UyypScqrWh9mkKCmWWAN0

Content-Type: application/json

{"col\_id": 6081}

Аналогичный SQL запрос:

SELECT remove\_collection\_by\_id(6081)

Пустой ответ.

Демонстрация, что запись удалена:

HTTP запрос:

GET http://localhost:3000/collection?id=eq.6081

Аналогичный SQL запрос:

SELECT \* FROM collection WHERE id = 6081

Ответ:

[]

## Реализация защиты базы данных

В данном курсовом проекте для защиты базы данных использовались расширение pgcrypto, а именно функции:

* crypt - функция, которая шифрует строку с солью, собственным алгоритмом, который работает быстрее и безопаснее, чем алгоритмы sha-1 и md-5, но при этом основывается на них.

Данная функция поддерживает проверку пароля не раскрывая соль, которая использовалась при генерации, что обеспечивает дополнительную безопасность;

* gen\_​salt - функция, которая генерирует соль и указывает основной алгоритм шифрования данных.

Защиту, также обеспечивала программа PostgREST, описание которого приведено в разделе «Администрирование», так как он не даёт возможности запуска полноценных SQL скриптов на сервере со стороны пользователя, что исключает SQL инъекции и возможность входа на базу данных, при изолировании её на отдельном сервер.

Аналогично написание многих хранимых процедур и триггеров остерегает базу данных от SQL инъекций и оберегает от возможного человеческого фактора, при администрировании базы данных.

## Результат выполнения процедур

Ниже представлены результаты выполнения процедур, исходные данные таблиц представлены в приложении А, а код процедур представлен в приложении Б.

Код запуска процедуры get\_age\_id:

SELECT get\_age\_id('adult');

Результат выполнения процедуры get\_age\_id, представлен на рисунке 4.



Рисунок 4 – Результат выполнения процедуры get\_age\_id

Код запуска процедуры get\_collector\_id:

SELECT get\_collector\_id('Стекольников');

Результат выполнения процедуры get\_collector\_id, представлен на рисунке 5.

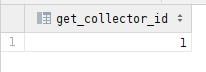


Рисунок 5 – Результат выполнения процедуры get\_collector\_id

Код запуска процедуры get\_country\_id:

SELECT get\_country\_id('Россия');

Результат выполнения процедуры get\_country\_id, представлен на рисунке 6.

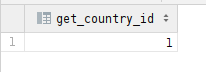


Рисунок 6 – Результат выполнения процедуры get\_country\_id

Код запуска процедуры get\_order\_id:

SELECT get\_order\_id('rodentia');

Результат выполнения процедуры get\_order\_id, представлен на рисунке 7.



Рисунок 7 – Результат выполнения процедуры get\_order\_id

Код запуска процедуры get\_family\_id:

SELECT get\_family\_id('cricetidae', 1);

Результат выполнения процедуры get\_family\_id, представлен на рисунке 8.



Рисунок 8 – Результат выполнения процедуры get\_family\_id

Код запуска процедуры get\_genus\_id:

SELECT get\_genus\_id('clethrionomys', 1);

Результат выполнения процедуры get\_genus\_id, представлен на рисунке 9.



Рисунок 9 – Результат выполнения процедуры get\_genus\_id

Код запуска процедуры get\_kind\_id:

SELECT get\_kind\_id('rufocanus', 1);

Результат выполнения процедуры get\_kind\_id, представлен на рисунке 10.



Рисунок 10 – Результат выполнения процедуры get\_kind\_id

Код запуска процедуры add\_collection:

SELECT add\_collection(vauch\_inst := 'ЗИН', date\_collect := '2023-10-1');

Добавленная запись, с помощью процедуры add\_collection, представлена на рисунке 11.

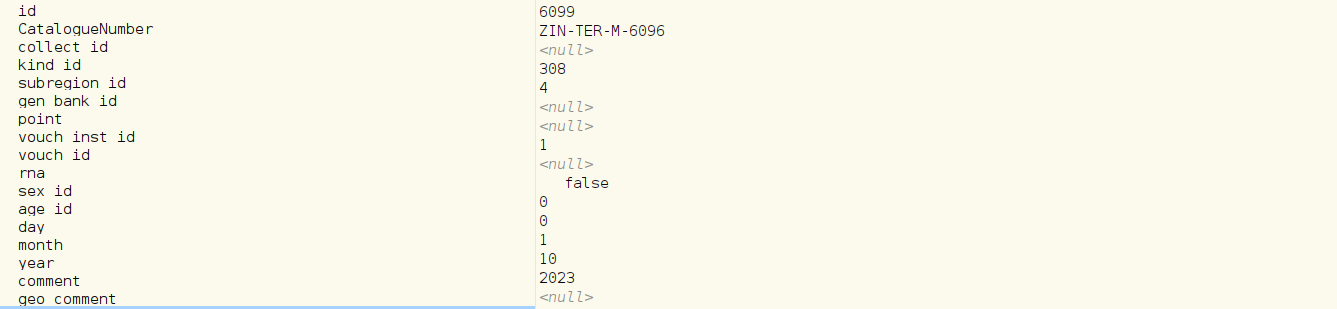


Рисунок 11 – Запись, добавленная процедурой add\_collection

Код запуска процедуры get\_region\_id:

SELECT get\_region\_id('Алтай, Республика', 1);

Результат выполнения процедуры get\_region\_id, представлен на рисунке 12.



Рисунок 12 – Результат выполнения процедуры get\_region\_id

Код запуска процедуры get\_subregion\_id:

SELECT get\_subregion\_id('Юг', 34);

Исходные данные для запроса get\_subregion\_id, представлены на рисунке 13.

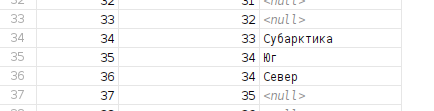


Рисунок 13 – Исходные данные для запроса get\_subregion\_id

Результат выполнения процедуры get\_subregion\_id, представлен на рисунке 14.



Рисунок 14 – Результат выполнения процедуры get\_subregion\_id

Код запуска процедуры get\_sex\_id:

SELECT get\_sex\_id('Мужской');

Результат выполнения процедуры get\_sex\_id, представлен на рисунке 15.



Рисунок 15 – Результат выполнения процедуры get\_sex\_id

Код запуска процедуры get\_vouch\_inst\_id:

SELECT get\_vouch\_inst\_id('ЗИН');

Результат выполнения процедуры get\_vouch\_inst\_id, представлен на рисунке 16.



Рисунок 16 – Результат выполнения процедуры get\_vouch\_inst\_id

Код запуска процедуры add\_topology:

SELECT add\_topology('Vasya', 'Pankov');

Записи, добавленные в таблицы order и family в результате выполнения процедуры add\_topology, представлен на рисунках 17-18.

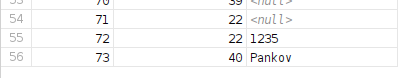


Рисунок 17 – Добавленная запись в таблицу family

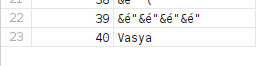


Рисунок 18 – Добавленная запись в таблицу order

# Разработка клиентского приложения

Код клиентского приложения представлен в приложении Д. На рисунках 19-20 представлены скриншоты разработанного клиентского приложения.

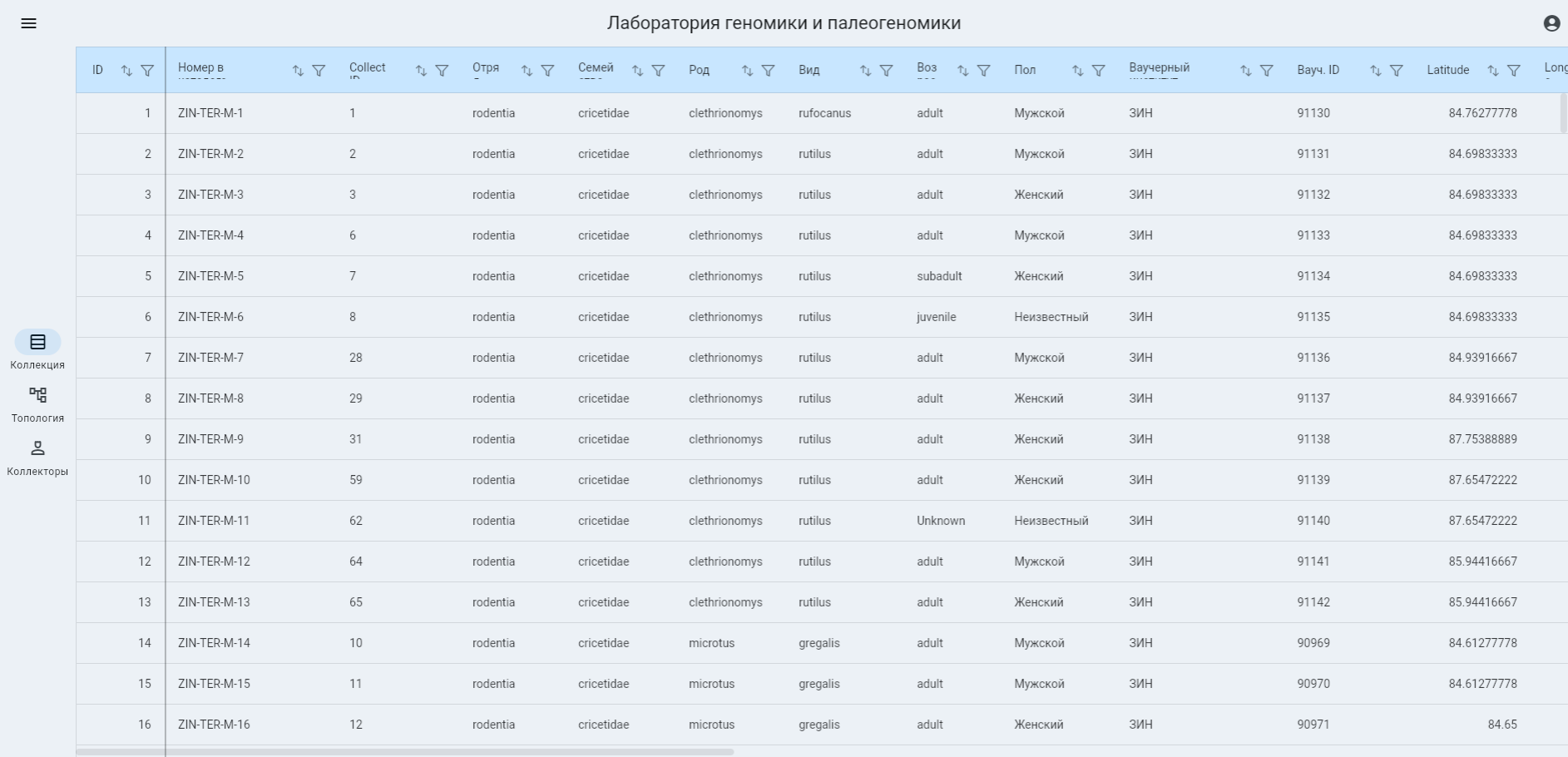


Рисунок 19 – Таблица коллекция

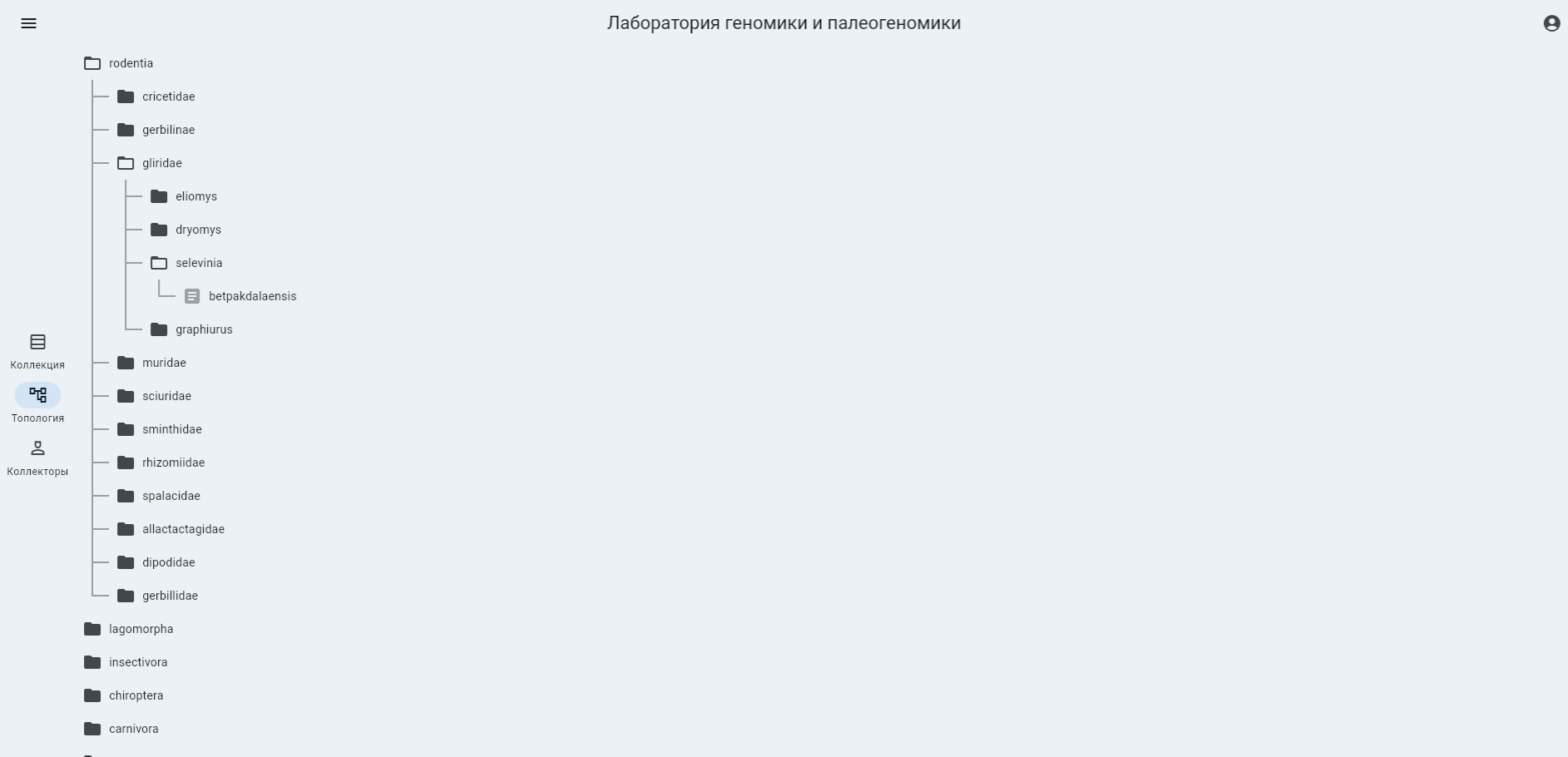


Рисунок 20 – Отображение топологии

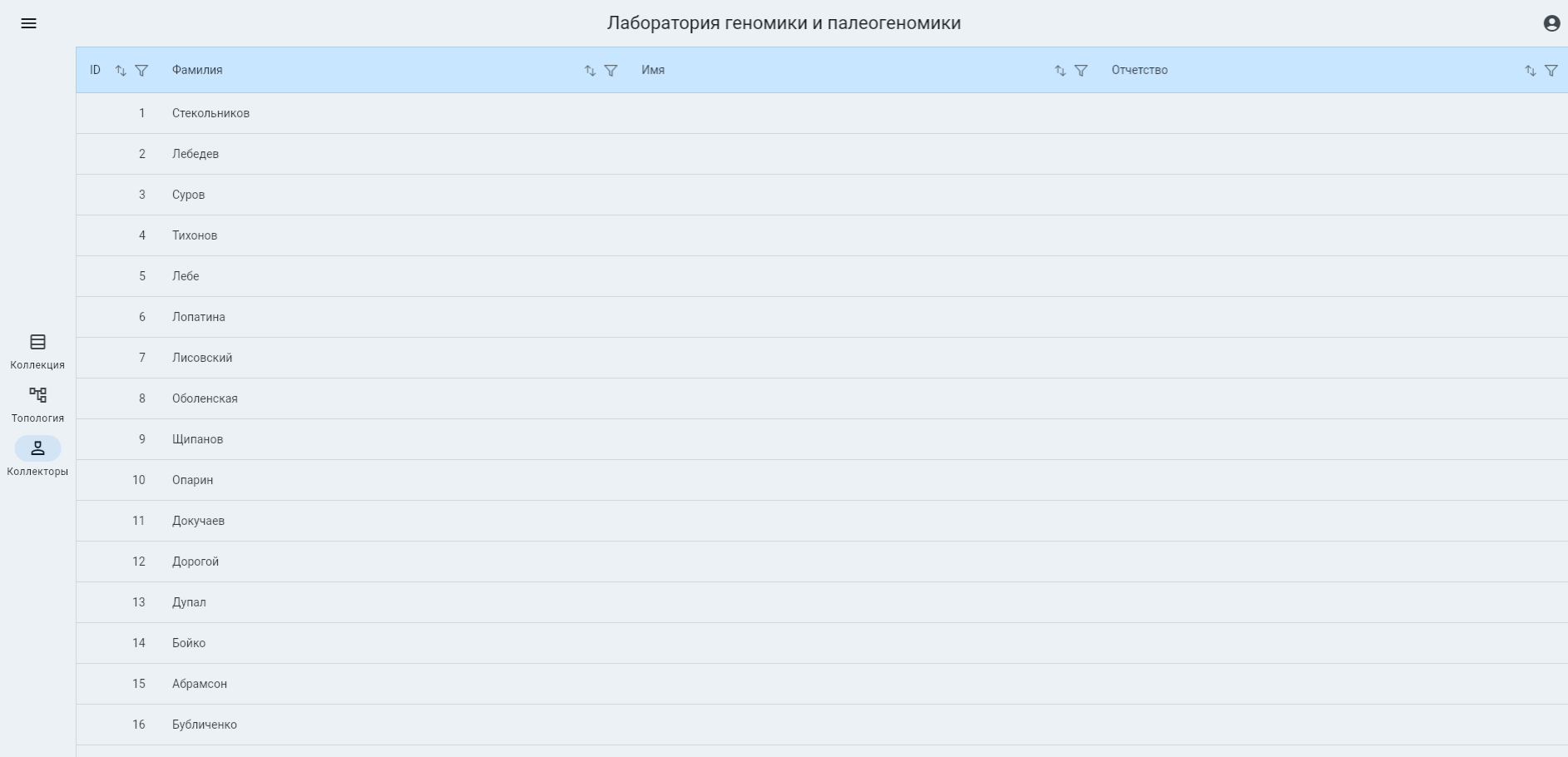


Рисунок 21 – Отображение коллекторов

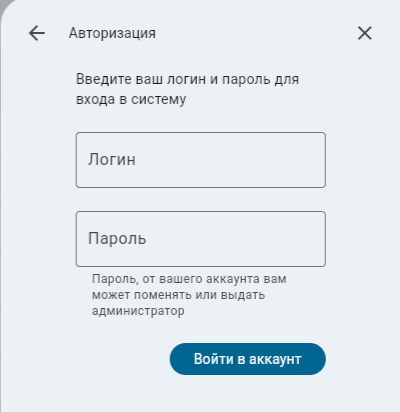


Рисунок 22 – Отображение авторизации

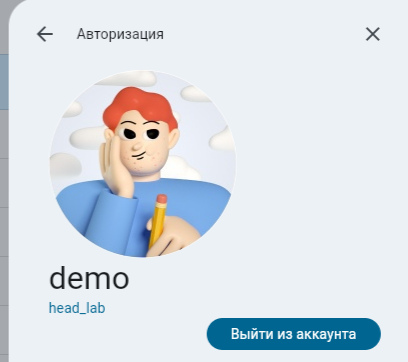


Рисунок 23 – Отображение профиля пользователя

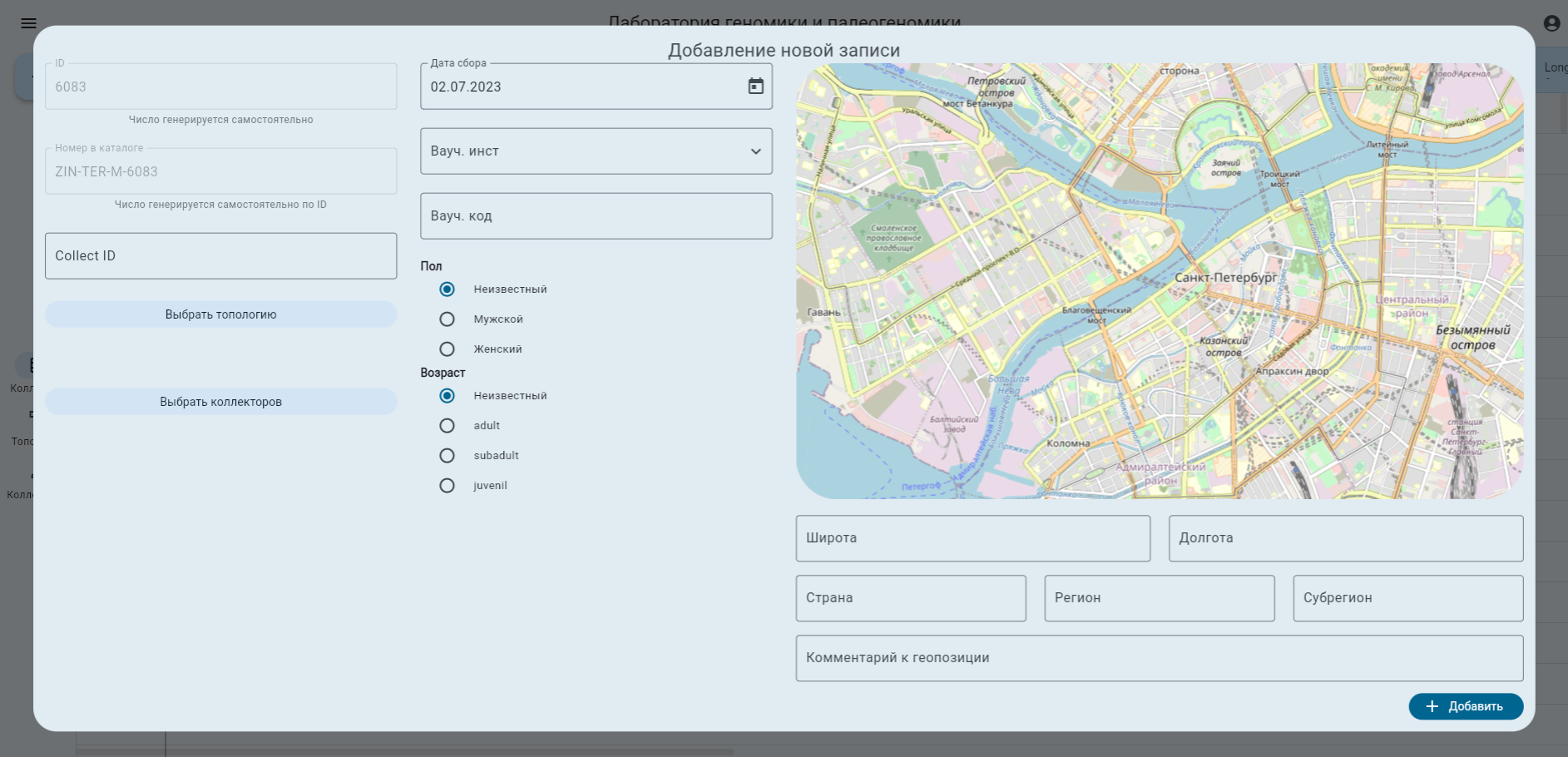


Рисунок 24 – Отображение добавления новой записи в коллекцию

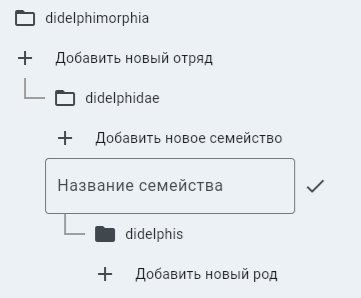


Рисунок 25 – Отображение добавления новой топологии

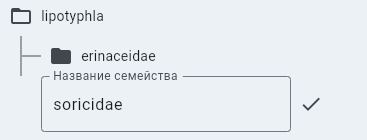


Рисунок 26 – Отображение редактирования топологии

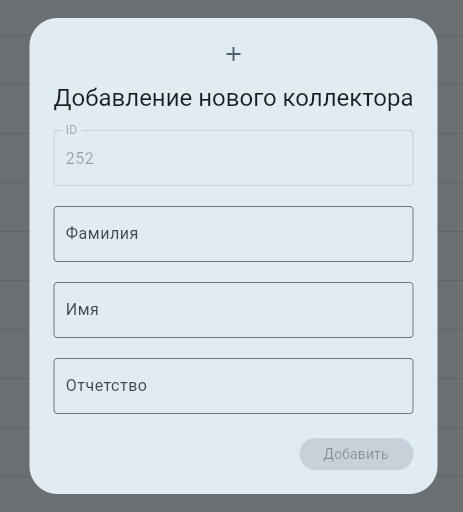


Рисунок 27 – Отображение добавления нового коллектора

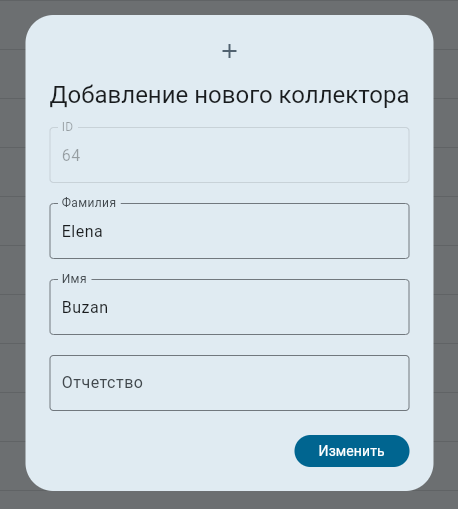


Рисунок 28 – Отображение изменение коллектора

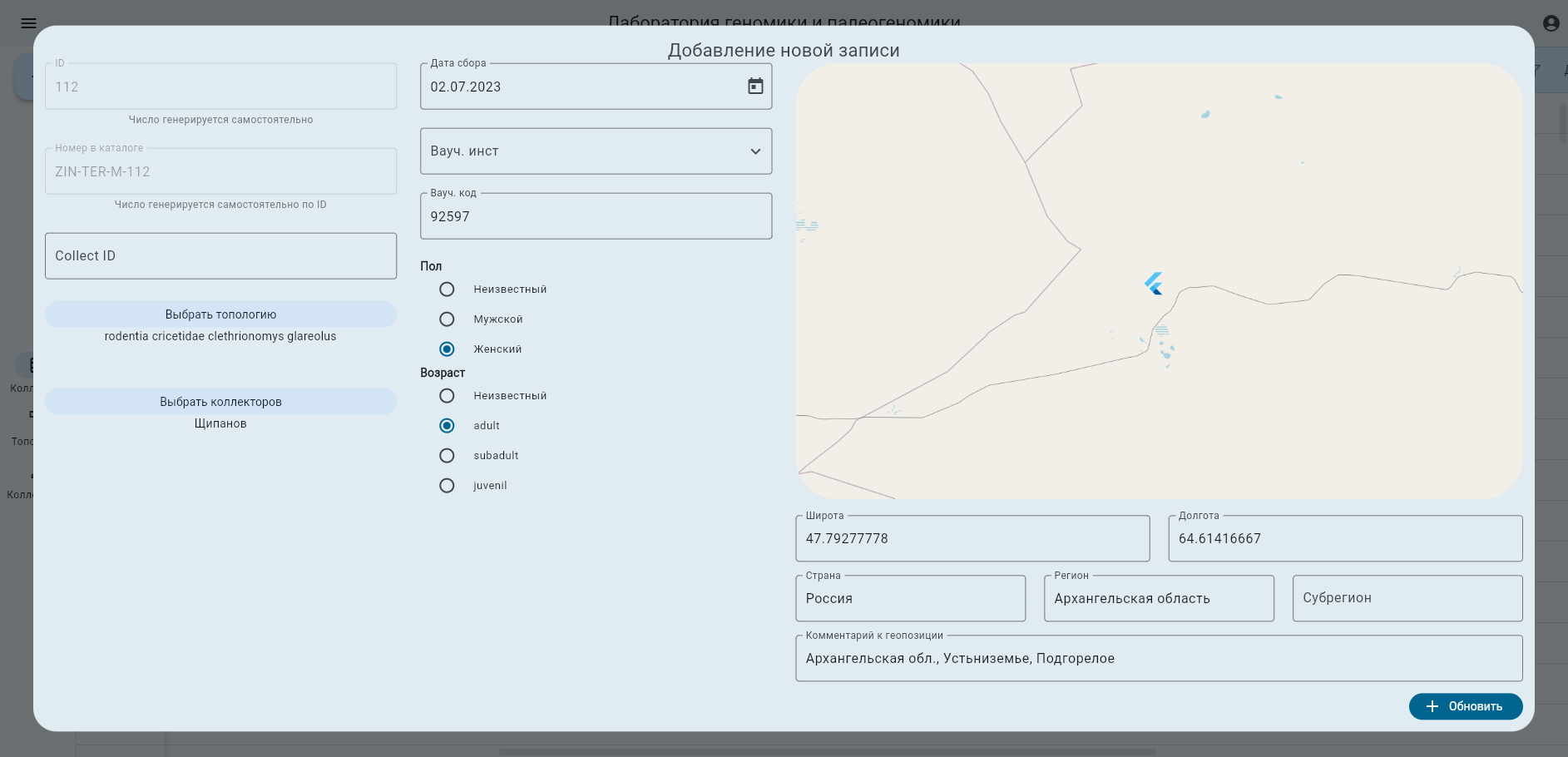


Рисунок 29 – Отображение изменение записи коллекции

# ЗАКЛЮЧЕНИЕ

В результате проведения производственной практики в Лаборатории геномики и палеогеномики при Государственном зоологическом институте Российской Федерации была поставлена цель разработки информационной системы, которая облегчит учет коллекционных записей экспедиционных сборов млекопитающих и обеспечит возможность анализа полученной информации для научных исследований в данной области.

В ходе работы были решены следующие задачи: изучены методы и технологии учета и хранения информации о коллекционных записях экспедиционных сборов млекопитающих; определены функциональные требования к информационной системе; разработана структура базы данных для хранения информации о коллекционных записях; разработан интерфейс для ввода и обработки данных; реализована информационная система на основе выбранных методов и технологий.

Результаты данной работы имеют важное значение для дальнейшего развития исследований в области геномики и палеогеномики млекопитающих. Созданная информационная система обеспечивает удобный и эффективный способ учета и анализа коллекционных записей, что позволит проводить более точные и всесторонние исследования. Кроме того, разработанный интерфейс обеспечивает удобство использования системы как для исследователей, так и для других заинтересованных сторон.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1 Документация расширения pgcrypto для шифрования PostgreSQL – URL: https://www.postgresql.org/docs/current/pgcrypto.html (дата обращения 01.06.2023).

2 Документация программы для создания restful API на основе базы данных PostgreSQL – URL: https://postgrest.org/en/stable/ (дата обращения 01.06.2023).

3 Документация языка программирования Dart – URL: https://dart.dev/guides (дата обращения 01.06.2023).

4 Документация библиотеки для разработки кроссплатформенных приложений Flutter – URL: https://docs.flutter.dev (дата обращения 01.06.2023).

5 Документация библиотеки для Flutter для отображения таблиц Syncfusion DataGrid – URL: https://help.syncfusion.com/flutter/datagrid/getting-started (дата обращения 01.06.2023).

6 Расширение для создание jwt токенов в PostgrSQL – URL: https://github.com/michelp/pgjwt (дата обращения 01.06.2023).

7 Дизайн система от Google Material 3 – URL: https://m3.material.io (дата обращения 01.06.2023).

8 Библиотека для организации http запросов в языке программировании Dart – URL: https://pub.dev/packages/http (дата обращения 01.06.2023).

9 Nominatim API – API для поиска мест по адресу или по координатам – URL: https://nominatim.org/release-docs/latest/api/Overview/ (дата обращения 01.06.2023).

10 Библиотека для работы с картами – Flutter Map – URL: https://pub.dev/packages/flutter\_map (дата обращения 01.06.2023).

11 Классификация организмов 9 класс – URL: https://www.yaklass.ru/p/biologia/9-klass/osnovy-evoliutcionnogo-ucheniia-246743/poniatie-vida-printcipy-sistematiki-246744/re-32dcb240-dd92-4865-bee7-96de4ee12880#:~:text=%D0%91%D0%BB%D0%B8%D0%B7%D0%BA%D0%B8%D0%B5%20(%D1%80%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5)%20%D0%B2%D0%B8%D0%B4%D1%8B%20%D0%B6%D0%B8%D0%B2%D0%BE%D1%82%D0%BD%D1%8B%D1%85%20%D0%BE%D0%B1%D1%8A%D0%B5%D0%B4%D0%B8%D0%BD%D1%8F%D1%8E%D1%82%D1%81%D1%8F,%2C%20%D0%B0%20%D1%82%D0%B8%D0%BF%D1%8B%20%E2%80%94%20%D0%B2%20%D1%86%D0%B0%D1%80%D1%81%D1%82%D0%B2%D0%BE (дата обращения 01.06.2023).

# ПРИЛОЖЕНИЕ А Созданные таблицы БД

Код созданных таблиц:

CREATE TABLE auth.users (

login text NOT NULL,

pass text NOT NULL,

role name NOT NULL,

avatar text,

CONSTRAINT users\_pass\_check CHECK ((length(pass) < 512)),

CONSTRAINT users\_role\_check CHECK ((length((role)::text) < 512))

);

CREATE TABLE public.age (

id integer NOT NULL,

name character varying(20)

);

CREATE SEQUENCE public.age\_id\_seq

START WITH 0

INCREMENT BY 1

MINVALUE 0

NO MAXVALUE

CACHE 1;

CREATE TABLE public.collection (

id integer NOT NULL,

"CatalogueNumber" text,

collect\_id text,

kind\_id integer NOT NULL,

subregion\_id integer NOT NULL,

gen\_bank\_id character varying(20),

point public.geography(Point,4326),

vouch\_inst\_id integer,

vouch\_id character varying(20),

rna boolean DEFAULT false NOT NULL,

sex\_id integer,

age\_id integer,

day integer,

month integer,

year integer NOT NULL,

comment text,

geo\_comment text,

file\_url text,

CONSTRAINT genbank\_check CHECK ((((gen\_bank\_id)::text ~ 'OP\d+'::text) OR (gen\_bank\_id IS NULL)))

);

CREATE SEQUENCE public.collection\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

CREATE TABLE public.collector (

id integer NOT NULL,

last\_name character varying(100) NOT NULL,

first\_name character varying(100),

second\_name character varying(100)

);

CREATE SEQUENCE public.collector\_id\_seq

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

CREATE TABLE public.collector\_to\_collection (

collector\_id integer NOT NULL,

collection\_id integer NOT NULL

);

CREATE TABLE public.country (

id integer NOT NULL,

name text NOT NULL

);

CREATE SEQUENCE public.country\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

CREATE TABLE public.family (

id integer NOT NULL,

order\_id integer NOT NULL,

name character varying(80)

);

CREATE SEQUENCE public.family\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

CREATE TABLE public.genus (

id integer NOT NULL,

family\_id integer NOT NULL,

name character varying(80)

);

CREATE SEQUENCE public.genus\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

CREATE TABLE public.kind (

id integer NOT NULL,

genus\_id integer NOT NULL,

name character varying(80)

);

CREATE SEQUENCE public.kind\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

CREATE TABLE public."order" (

id integer NOT NULL,

name character varying(80)

);

CREATE SEQUENCE public.order\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

CREATE TABLE public.region (

id integer NOT NULL,

country\_id integer NOT NULL,

name text

);

CREATE SEQUENCE public.region\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

CREATE TABLE public.sex (

id integer NOT NULL,

name character varying(40) NOT NULL

);

CREATE SEQUENCE public.sex\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

CREATE TABLE public.subregion (

id integer NOT NULL,

region\_id integer NOT NULL,

name text

);

CREATE SEQUENCE public.subregion\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

CREATE TABLE public.voucher\_institute (

id integer NOT NULL,

name text NOT NULL

);

CREATE SEQUENCE public.voucher\_institute\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER TABLE ONLY public.age ALTER COLUMN id SET DEFAULT nextval('public.age\_id\_seq'::regclass);

ALTER TABLE ONLY public.collection ALTER COLUMN id SET DEFAULT nextval('public.collection\_id\_seq'::regclass);

ALTER TABLE ONLY public.collector ALTER COLUMN id SET DEFAULT nextval('public.collector\_id\_seq'::regclass);

--

-- Name: family id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.family ALTER COLUMN id SET DEFAULT nextval('public.family\_id\_seq'::regclass);

--

-- Name: genus id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.genus ALTER COLUMN id SET DEFAULT nextval('public.genus\_id\_seq'::regclass);

--

-- Name: kind id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.kind ALTER COLUMN id SET DEFAULT nextval('public.kind\_id\_seq'::regclass);

--

-- Name: order id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."order" ALTER COLUMN id SET DEFAULT nextval('public.order\_id\_seq'::regclass);

--

-- Name: region id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.region ALTER COLUMN id SET DEFAULT nextval('public.region\_id\_seq'::regclass);

--

-- Name: sex id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.sex ALTER COLUMN id SET DEFAULT nextval('public.sex\_id\_seq'::regclass);

--

-- Name: subregion id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.subregion ALTER COLUMN id SET DEFAULT nextval('public.subregion\_id\_seq'::regclass);

--

-- Name: voucher\_institute id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.voucher\_institute ALTER COLUMN id SET DEFAULT nextval('public.voucher\_institute\_id\_seq'::regclass);

--

-- Name: users users\_pkey; Type: CONSTRAINT; Schema: auth; Owner: postgres

--

ALTER TABLE ONLY auth.users

ADD CONSTRAINT users\_pkey PRIMARY KEY (login);

--

-- Name: age age\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.age

ADD CONSTRAINT age\_pk PRIMARY KEY (id);

--

-- Name: collection collection\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.collection

ADD CONSTRAINT collection\_pk PRIMARY KEY (id);

--

-- Name: collector collector\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.collector

ADD CONSTRAINT collector\_pk PRIMARY KEY (id);

--

-- Name: collector\_to\_collection collector\_to\_collection\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.collector\_to\_collection

ADD CONSTRAINT collector\_to\_collection\_pk PRIMARY KEY (collector\_id, collection\_id);

--

-- Name: country country\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.country

ADD CONSTRAINT country\_pk PRIMARY KEY (id);

--

-- Name: family family\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.family

ADD CONSTRAINT family\_pk PRIMARY KEY (id);

--

-- Name: genus genus\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.genus

ADD CONSTRAINT genus\_pk PRIMARY KEY (id);

--

-- Name: kind kind\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.kind

ADD CONSTRAINT kind\_pk PRIMARY KEY (id);

--

-- Name: order order\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."order"

ADD CONSTRAINT order\_pk PRIMARY KEY (id);

--

-- Name: region region\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.region

ADD CONSTRAINT region\_pk PRIMARY KEY (id);

--

-- Name: sex sex\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.sex

ADD CONSTRAINT sex\_pk PRIMARY KEY (id);

--

-- Name: subregion subregion\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.subregion

ADD CONSTRAINT subregion\_pk PRIMARY KEY (id);

--

-- Name: voucher\_institute voucher\_institute\_pk; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.voucher\_institute

ADD CONSTRAINT voucher\_institute\_pk PRIMARY KEY (id);

--

-- Name: users encrypt\_pass; Type: TRIGGER; Schema: auth; Owner: postgres

--

CREATE TRIGGER encrypt\_pass BEFORE INSERT OR UPDATE ON auth.users FOR EACH ROW EXECUTE FUNCTION auth.encrypt\_pass();

--

-- Name: collection collection\_age\_id\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.collection

ADD CONSTRAINT collection\_age\_id\_fk FOREIGN KEY (age\_id) REFERENCES public.age(id) ON UPDATE CASCADE ON DELETE RESTRICT;

--

-- Name: collection collection\_kind\_null\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.collection

ADD CONSTRAINT collection\_kind\_null\_fk FOREIGN KEY (kind\_id) REFERENCES public.kind(id);

--

-- Name: collection collection\_sex\_id\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.collection

ADD CONSTRAINT collection\_sex\_id\_fk FOREIGN KEY (sex\_id) REFERENCES public.sex(id) ON UPDATE CASCADE ON DELETE RESTRICT;

--

-- Name: collection collection\_subregion\_null\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.collection

ADD CONSTRAINT collection\_subregion\_null\_fk FOREIGN KEY (subregion\_id) REFERENCES public.subregion(id);

--

-- Name: collection collection\_voucher\_institute\_null\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.collection

ADD CONSTRAINT collection\_voucher\_institute\_null\_fk FOREIGN KEY (vouch\_inst\_id) REFERENCES public.voucher\_institute(id);

--

-- Name: collector\_to\_collection collector\_to\_collection\_collection\_null\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.collector\_to\_collection

ADD CONSTRAINT collector\_to\_collection\_collection\_null\_fk FOREIGN KEY (collection\_id) REFERENCES public.collection(id) ON UPDATE CASCADE ON DELETE CASCADE;

--

-- Name: collector\_to\_collection collector\_to\_collection\_collector\_null\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.collector\_to\_collection

ADD CONSTRAINT collector\_to\_collection\_collector\_null\_fk FOREIGN KEY (collector\_id) REFERENCES public.collector(id) ON UPDATE CASCADE ON DELETE CASCADE;

--

-- Name: family family\_order\_null\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.family

ADD CONSTRAINT family\_order\_null\_fk FOREIGN KEY (order\_id) REFERENCES public."order"(id);

--

-- Name: genus genus\_family\_null\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.genus

ADD CONSTRAINT genus\_family\_null\_fk FOREIGN KEY (family\_id) REFERENCES public.family(id);

--

-- Name: region region\_country\_null\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.region

ADD CONSTRAINT region\_country\_null\_fk FOREIGN KEY (country\_id) REFERENCES public.country(id);

--

-- Name: subregion subregion\_region\_null\_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.subregion

ADD CONSTRAINT subregion\_region\_null\_fk FOREIGN KEY (region\_id) REFERENCES public.region(id);

Данные в таблицах:

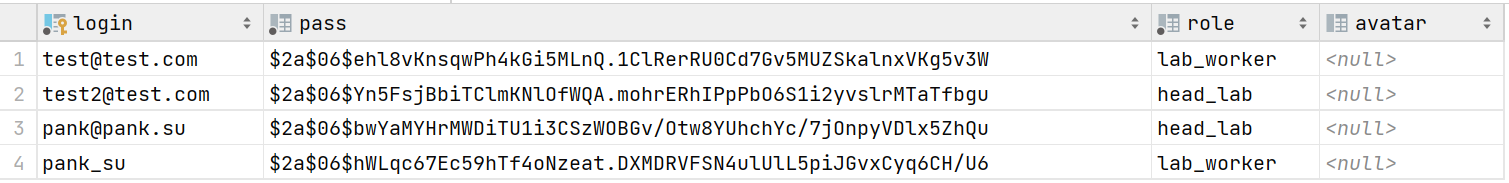


Рисунок A.1 – Таблица users

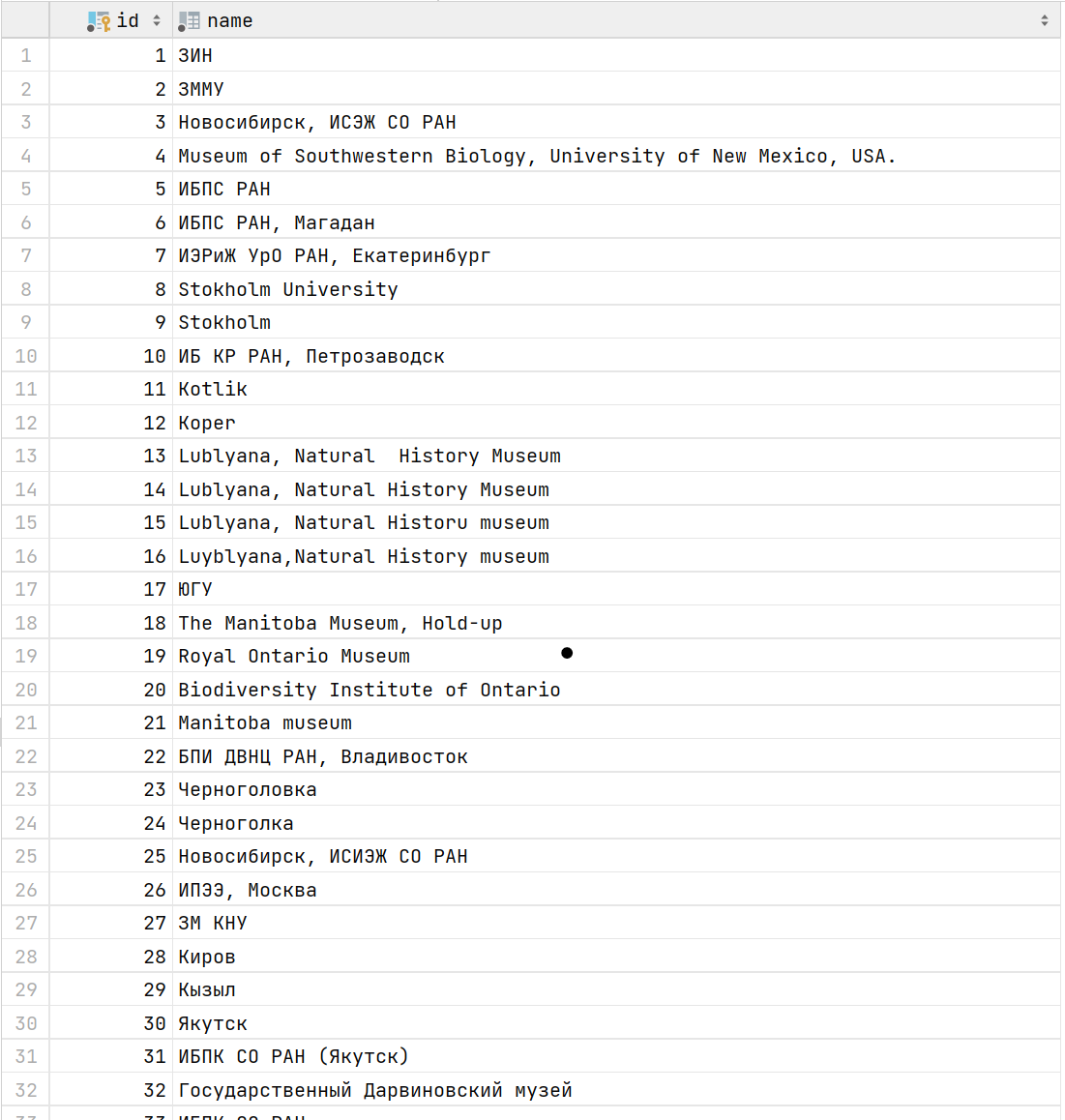


Рисунок A.2 – Таблица voucher\_institute

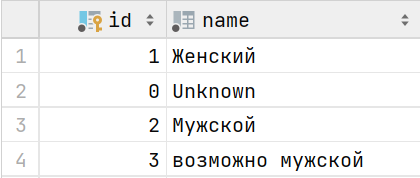


Рисунок A.3 – Таблица sex

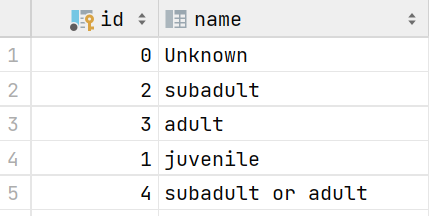


Рисунок A.4 – Таблица age

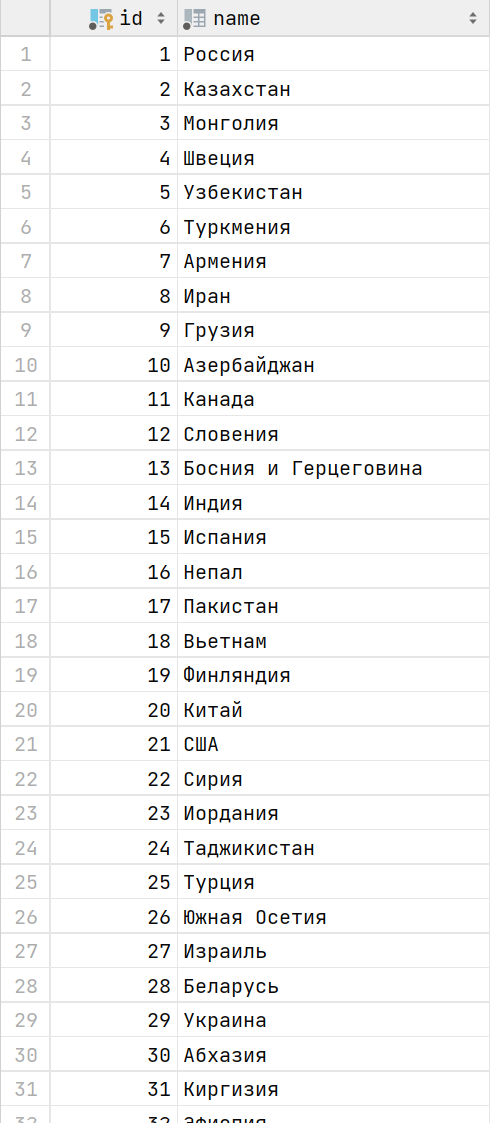


Рисунок A.5 – Таблица country

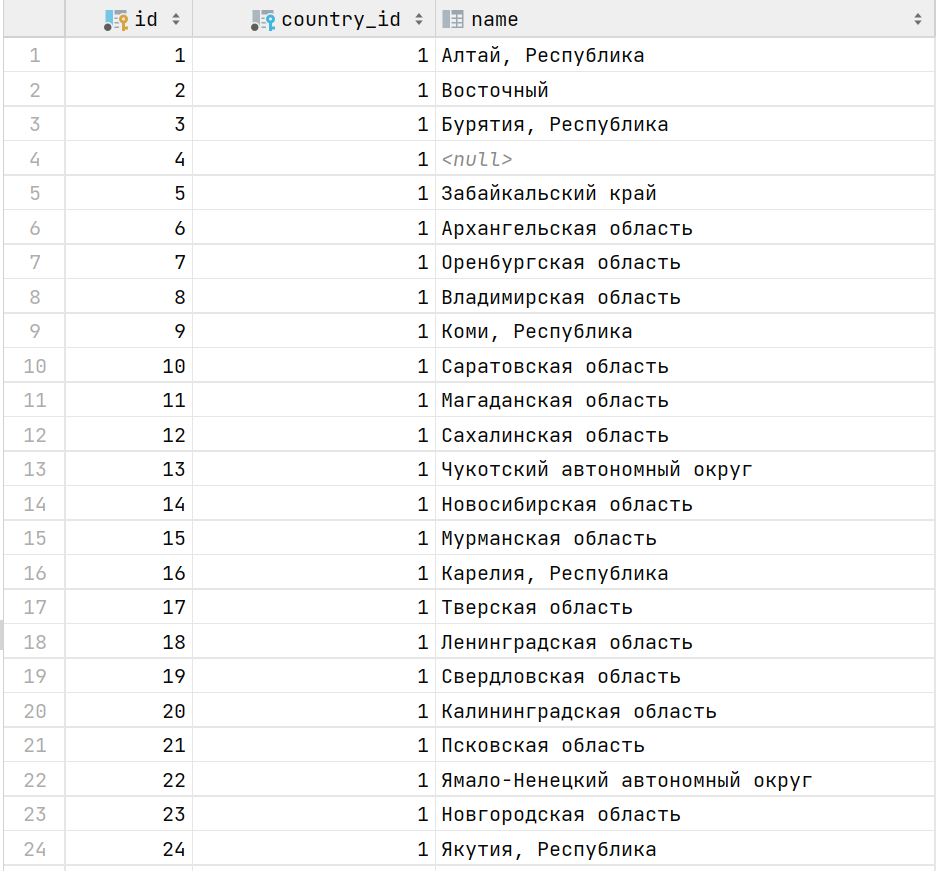


Рисунок A.6 – Таблица region

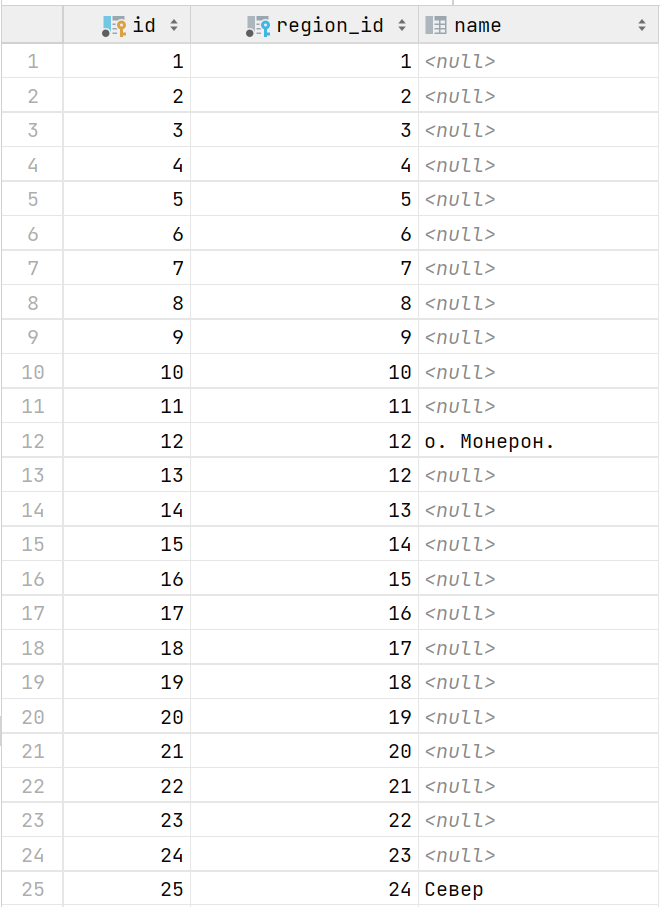


Рисунок A.7 – Таблица subregion



Рисунок A.8 – Таблица order

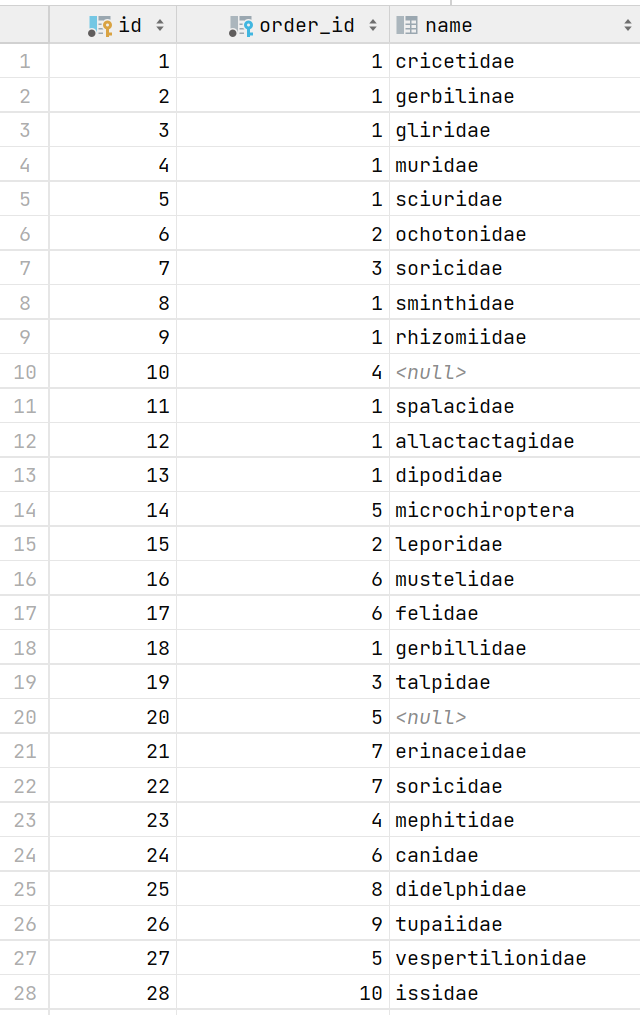


Рисунок A.9 – Таблица family

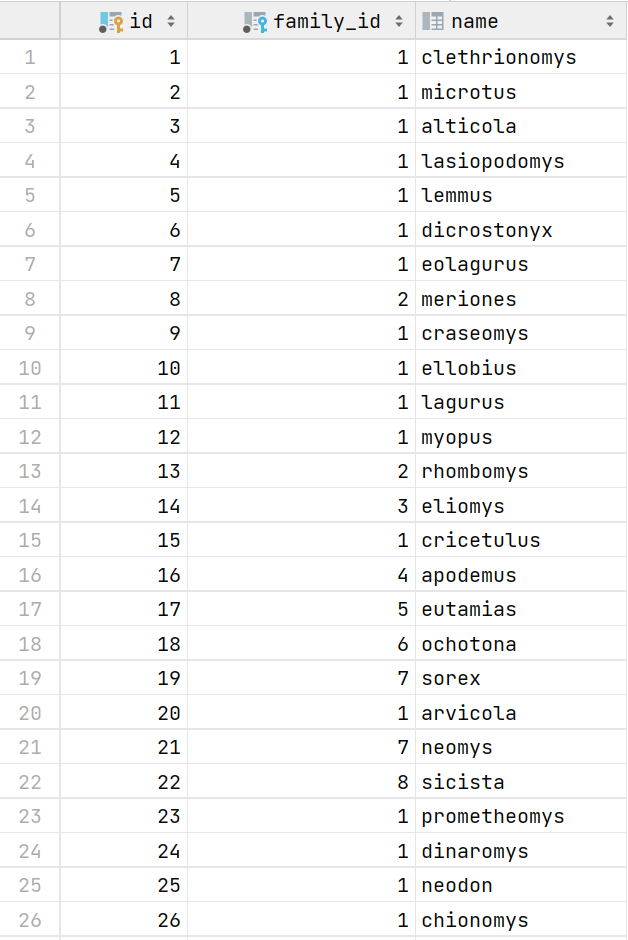


Рисунок A.10 – Таблица genus

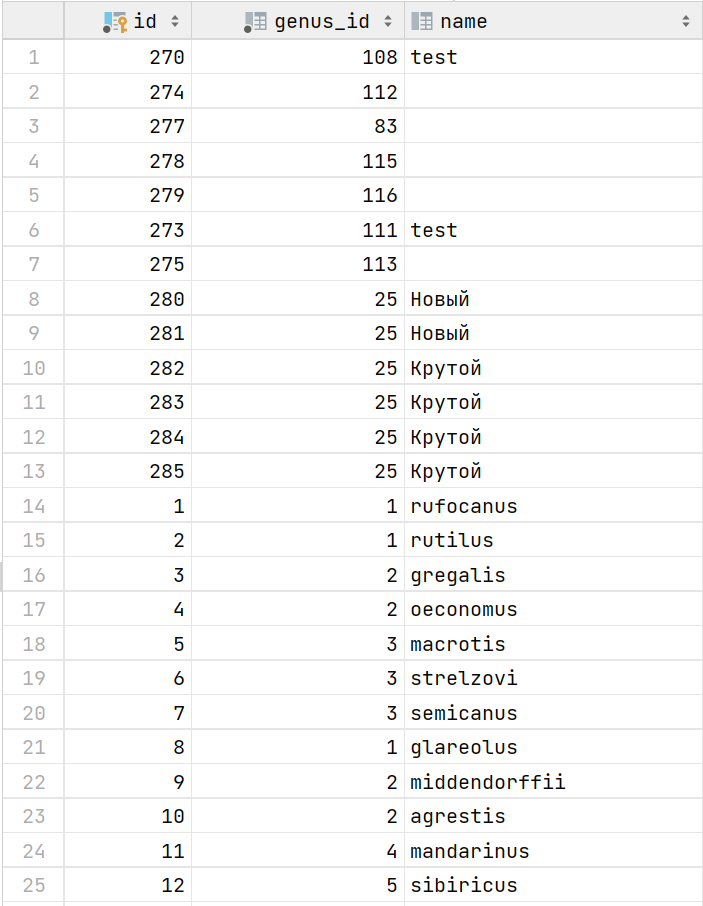


Рисунок A.11 – Таблица kind

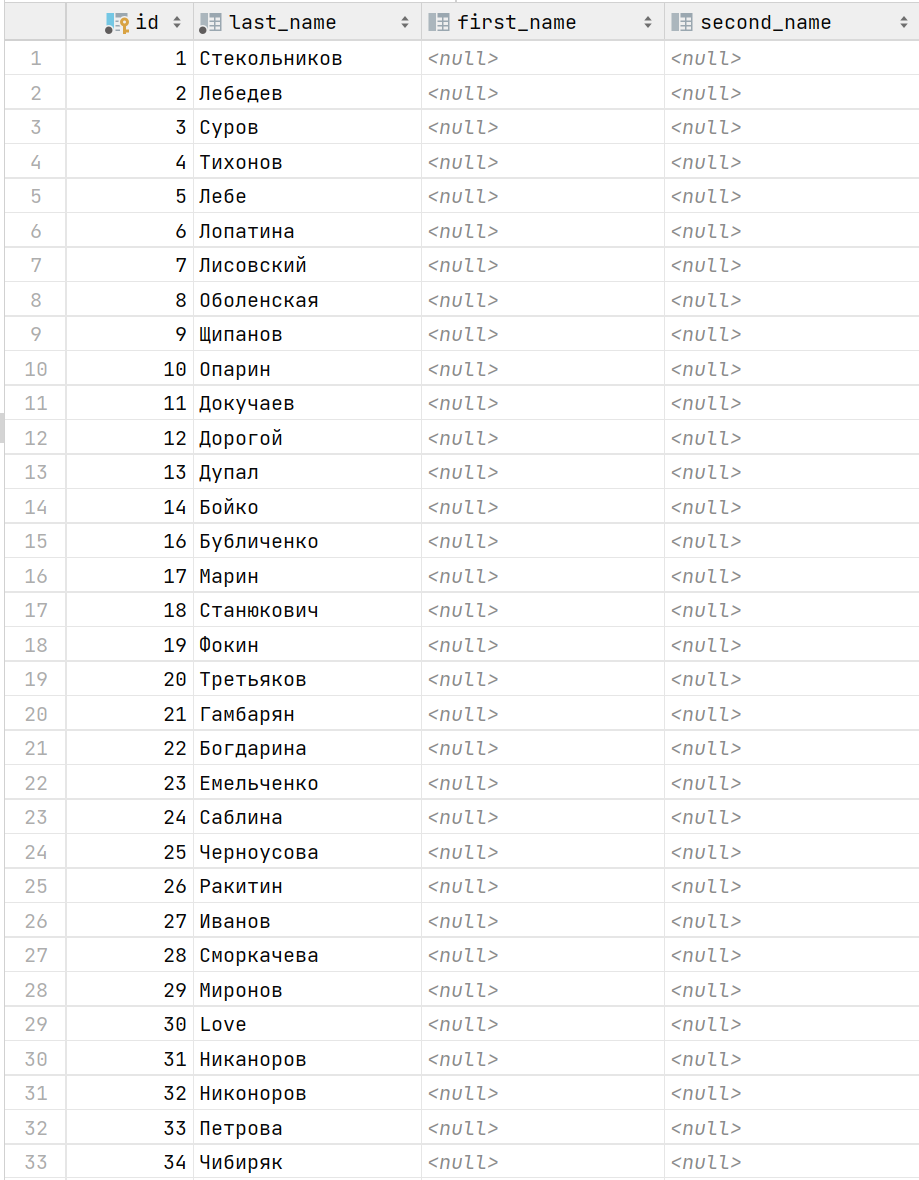


Рисунок A.12 – Таблица collector

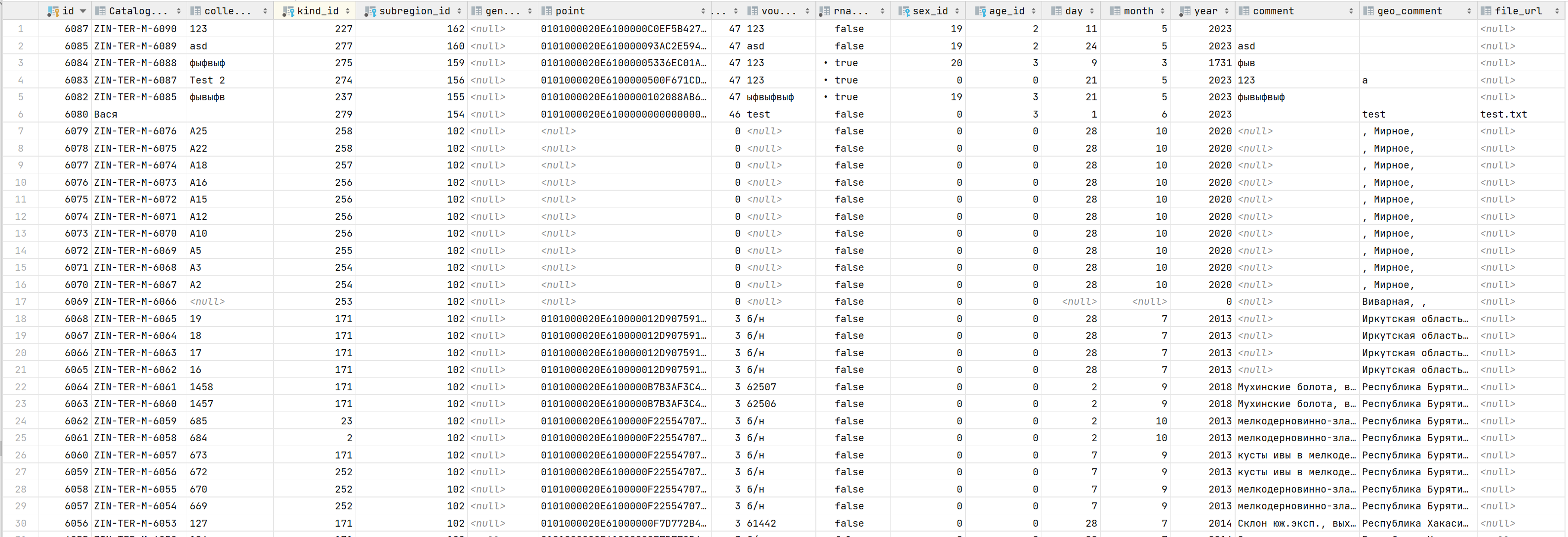


Рисунок A.13 – Таблица collection

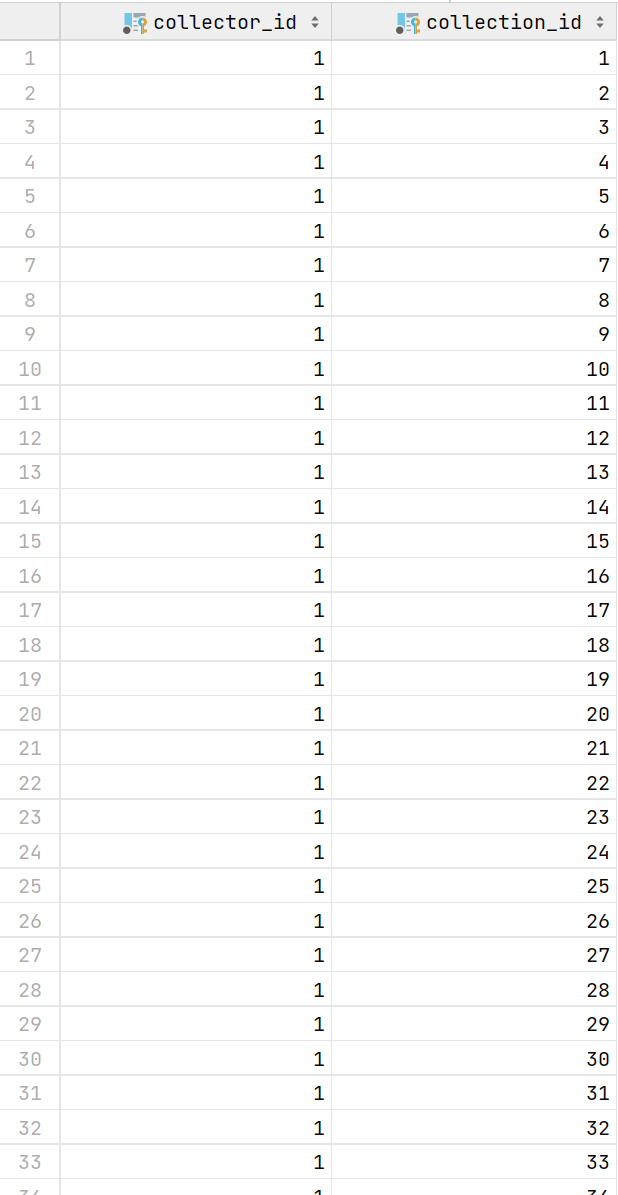


Рисунок A.14 – Таблица collector\_to\_collection

# ПРИЛОЖЕНИЕ Б Разработанные хранимые процедуры, триггеры и представления

CREATE OR REPLACE FUNCTION get\_age\_id(age\_name varchar(20))

RETURNS integer

AS

$$

DECLARE

age\_id integer;

BEGIN

SELECT id INTO age\_id FROM age WHERE $1 = age.name;

IF age\_id IS NULL THEN

INSERT INTO age(name) VALUES ($1);

SELECT id INTO age\_id FROM age WHERE $1 = age.name;

END IF;

RETURN age\_id;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION get\_sex\_id(sex\_name varchar(40))

RETURNS integer

AS

$$

DECLARE

sex\_id integer;

BEGIN

SELECT id INTO sex\_id FROM sex WHERE $1 = sex.name;

IF sex\_id IS NULL THEN

INSERT INTO sex(name) VALUES ($1);

SELECT id INTO sex\_id FROM sex WHERE $1 = sex.name;

END IF;

RETURN sex\_id;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION get\_order\_id(order\_name varchar(80))

RETURNS integer

AS

$$

DECLARE

order\_id integer;

BEGIN

SELECT id INTO order\_id FROM "order" WHERE ($1 = "order".name OR ($1 is NULL AND "order".name is null));

IF order\_id IS NULL THEN

INSERT INTO "order"(name) VALUES ($1);

SELECT get\_order\_id($1) INTO order\_id;

END IF;

RETURN order\_id;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION get\_family\_id(name varchar(80), order\_id integer)

RETURNS integer

AS

$$

DECLARE

family\_id integer;

BEGIN

SELECT id

INTO family\_id

FROM family

WHERE ($1 = family.name OR ($1 is NULL AND family.name is null))

AND $2 = family.order\_id;

IF family\_id IS NULL THEN

INSERT INTO family(name, order\_id) VALUES ($1, $2);

SELECT get\_family\_id($1, $2) INTO family\_id;

END IF;

RETURN family\_id;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION get\_genus\_id(name varchar(80), family\_id integer)

RETURNS integer

AS

$$

DECLARE

genus\_id integer;

BEGIN

SELECT id

INTO genus\_id

FROM genus

WHERE ($1 = genus.name OR ($1 is NULL AND genus.name is null))

AND $2 = genus.family\_id;

IF genus\_id IS NULL THEN

INSERT INTO genus(name, family\_id) VALUES ($1, $2);

SELECT get\_genus\_id($1, $2) INTO genus\_id;

END IF;

RETURN genus\_id;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION get\_kind\_id(name varchar(80), genus\_id integer)

RETURNS integer

AS

$$

DECLARE

kind\_id integer;

BEGIN

SELECT id

INTO kind\_id

FROM kind

WHERE ($1 = kind.name OR ($1 is NULL AND kind.name is null))

AND $2 = kind.genus\_id;

IF kind\_id IS NULL THEN

INSERT INTO kind(name, genus\_id) VALUES ($1, $2);

SELECT get\_kind\_id($1, $2) INTO kind\_id;

END IF;

RETURN kind\_id;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION get\_country\_id(name text)

RETURNS integer

AS

$$

DECLARE

country\_id integer;

BEGIN

SELECT id INTO country\_id FROM country WHERE $1 = country.name;

IF country\_id IS NULL THEN

INSERT INTO country(name) VALUES ($1);

SELECT get\_country\_id($1) INTO country\_id;

END IF;

RETURN country\_id;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION get\_region\_id(name text, country integer)

RETURNS integer

AS

$$

DECLARE

region\_id integer;

BEGIN

SELECT id

INTO region\_id

FROM region

WHERE ($1 = region.name OR ($1 is NULL AND region.name is null))

AND $2 = region.country\_id;

IF region\_id IS NULL THEN

INSERT INTO region(country\_id, name) VALUES ($2, $1);

SELECT get\_region\_id($1, $2) INTO region\_id;

END IF;

RETURN region\_id;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION get\_subregion\_id(name text, region\_id integer)

RETURNS integer

AS

$$

DECLARE

subregion\_id integer;

BEGIN

SELECT id

INTO subregion\_id

FROM subregion

WHERE ($1 = subregion.name OR ($1 is NULL AND subregion.name is null))

AND $2 = subregion.region\_id;

IF subregion\_id IS NULL THEN

INSERT INTO subregion(region\_id, name) VALUES ($2, $1);

SELECT get\_subregion\_id($1, $2) INTO subregion\_id;

END IF;

RETURN subregion\_id;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION get\_vouch\_inst\_id(name text)

RETURNS integer

AS

$$

DECLARE

vouch\_inst\_id integer;

BEGIN

SELECT id INTO vouch\_inst\_id FROM voucher\_institute WHERE $1 = voucher\_institute.name;

IF vouch\_inst\_id IS NULL THEN

INSERT INTO voucher\_institute(name) VALUES ($1);

SELECT get\_vouch\_inst\_id($1) INTO vouch\_inst\_id;

END IF;

RETURN vouch\_inst\_id;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION get\_collector\_id(last\_name varchar(100), first\_name varchar(100) DEFAULT NULL,

second\_name varchar(100) DEFAULT NULL)

RETURNS integer AS

$$

DECLARE

collector\_id integer;

BEGIN

SELECT id

INTO collector\_id

FROM collector

WHERE collector.last\_name = $1

AND (collector.first\_name = $2 or (collector.first\_name is NULL and $2 is NULL))

AND (collector.second\_name = $3 or (collector.second\_name is NULL and $3 is NULL));

IF collector\_id IS NULL THEN

INSERT INTO collector(last\_name, first\_name, second\_name) VALUES ($1, $2, $3);

SELECT get\_collector\_id($1, $2, $3) INTO collector\_id;

END IF;

RETURN collector\_id;

END;

$$ LANGUAGE plpgsql;

-- Добавление в коллекции

CREATE OR REPLACE FUNCTION add\_collection(collect\_id text default null, "order" varchar(80) default null,

family varchar(80) default null, genus varchar(80) default null,

kind varchar(80) default null, age varchar(20) DEFAULT 'Unknown', sex text DEFAULT 'Unknown', vauch\_inst text DEFAULT NULL, vauch\_id text default null,

point geography(point, 4326) default null, country text default null, region text default null, subregion text default null,

geocomment text default null, date\_collect date default null, comment text default null,

collectors text[][3] DEFAULT '{}', rna bool default false)

RETURNS void

LANGUAGE plpgsql

AS

$$

DECLARE

kind\_id\_ integer;

subregion\_id\_ integer DEFAULT 4;

collection\_id\_ integer;

collector text[];

collector\_id\_ integer;

BEGIN

kind\_id\_ := get\_kind\_id($5, (get\_genus\_id($4, (get\_family\_id($3, (get\_order\_id($2)))))));

IF ($11 IS NOT NULL) THEN

subregion\_id\_ := get\_subregion\_id($13, (get\_region\_id($12, (get\_country\_id($11)))));

END IF;

collection\_id\_ := (SELECT id FROM collection ORDER BY id DESC LIMIT 1);

INSERT INTO collection("CatalogueNumber", collect\_id, kind\_id, subregion\_id, point, vouch\_inst\_id, vouch\_id, sex\_id,

age\_id, day, month, year, comment, geo\_comment, rna)

VALUES (concat('ZIN-TER-M-', collection\_id\_), $1, kind\_id\_, subregion\_id\_, $10, get\_vouch\_inst\_id($8), $9,

get\_sex\_id($7), get\_age\_id($6),

extract(day from date\_collect), extract(month from date\_collect), extract(year from date\_collect), $16, $14,

$18);

collection\_id\_ := collection\_id\_ + 1;

IF (array\_length($17, 1) IS NULL) THEN RETURN;

END IF;

FOREACH collector SLICE 1 IN ARRAY $17

LOOP

collector\_id\_ := get\_collector\_id(collector[1]);

INSERT INTO collector\_to\_collection(collector\_id, collection\_id) VALUES (collector\_id\_, collection\_id\_);

END LOOP;

END

$$;

;

-- Удаление из коллекции по ID

CREATE OR REPLACE FUNCTION public.remove\_collection\_by\_id(col\_id int)

RETURNS void

LANGUAGE plpgsql AS

$$

BEGIN

DELETE FROM collection WHERE col\_id = id;

END;

$$;

-- Обновление записи

CREATE OR REPLACE FUNCTION update\_collection\_by\_id(col\_id int, collect\_id text DEFAULT null,

"order" varchar(80) DEFAULT null,

family varchar(80) DEFAULT null, genus varchar(80) DEFAULT null,

kind varchar(80) DEFAULT null, age varchar(20) DEFAULT 'Unknown',

sex text DEFAULT 'Unknown', vauch\_inst text DEFAULT null,

vauch\_id text DEFAULT null,

point geography(point, 4326) DEFAULT null, country text DEFAULT null,

region text DEFAULT null,

subregion text DEFAULT null,

geocomment text DEFAULT null, date\_collect date DEFAULT null,

comment text DEFAULT null,

collectors text[][3] DEFAULT '{}', rna bool default false)

RETURNS void

LANGUAGE plpgsql AS

$$

DECLARE

kind\_id\_ integer;

subregion\_id\_ integer DEFAULT 4;

collector text[];

collector\_id\_ int;

BEGIN

kind\_id\_ := get\_kind\_id($6, (get\_genus\_id($5, (get\_family\_id($4, (get\_order\_id($3)))))));

IF ($12 IS NOT NULL) THEN

subregion\_id\_ := get\_subregion\_id($14, (get\_region\_id($13, (get\_country\_id($12)))));

END IF;

UPDATE collection

SET kind\_id = kind\_id\_,

"CatalogueNumber" = concat('ZIN-TER-M-', col\_id),

subregion\_id = subregion\_id\_,

collect\_id = $2,

age\_id = get\_age\_id(age),

sex\_id = get\_sex\_id(sex),

vouch\_inst\_id = get\_vouch\_inst\_id(vauch\_inst),

vouch\_id = $10,

point = $11,

geo\_comment = geocomment,

day = extract(day from date\_collect),

month = extract(month from date\_collect),

year = extract(year from date\_collect),

comment = $17,

rna = $19

WHERE id = col\_id;

DELETE FROM collector\_to\_collection WHERE collection\_id = col\_id;

IF (array\_length($18, 1) IS NULL) THEN

RETURN;

END IF;

FOREACH collector SLICE 1 IN ARRAY $18

LOOP

collector\_id\_ := get\_collector\_id(collector[1]);

INSERT INTO collector\_to\_collection(collector\_id, collection\_id) VALUES (collector\_id\_, col\_id);

END LOOP;

END

$$;

;

create schema auth;

-- проверка существует ли роль

create or replace function

auth.check\_role\_exists() returns trigger as

$$

begin

if not exists (select 1 from pg\_roles as r where r.rolname = new.role) then

raise foreign\_key\_violation using message =

'unknown database role: ' || new.role;

-- return null;

end if;

return new;

end

$$ language plpgsql;

drop trigger if exists ensure\_user\_role\_exists on auth.users;

create constraint trigger ensure\_user\_role\_exists

after insert or update

on auth.users

for each row

execute procedure auth.check\_role\_exists();

create extension if not exists pgcrypto;

-- триггер хеширования паролей

create or replace function

auth.encrypt\_pass() returns trigger as

$$

begin

if tg\_op = 'INSERT' or new.pass <> old.pass then

new.pass = crypt(new.pass, gen\_salt('bf'));

end if;

return new;

end

$$ language plpgsql;

drop trigger if exists encrypt\_pass on auth.users;

create trigger encrypt\_pass

before insert or update

on auth.users

for each row

execute procedure auth.encrypt\_pass();

-- получение роли(фактически авторизация)

create or replace function

auth.user\_role(login text, pass text) returns name

language plpgsql

as

$$

begin

return (select role

from auth.users

where users.login = user\_role.login

and users.pass = crypt(user\_role.pass, users.pass));

end;

$$;

-- тип token - токен для входа пользователей

CREATE TYPE auth.jwt\_token AS

(

token text

);

create extension if not exists pgjwt;

ALTER DATABASE lab\_base SET "app.jwt\_secret" TO 'Q5He86xPvYscMiZxQw29gy8YkbD7a4aMDH1hQFP';

-- функция авторизации, для анонимного пользователя нужно только выполненные функций

create or replace function

public.login(login text, pass text) returns auth.jwt\_token as

$$

declare

\_role name;

result auth.jwt\_token;

begin

-- check email and password

select auth.user\_role(login, pass) into \_role;

if \_role is null then

raise invalid\_password using message = 'invalid user or password';

end if;

-- НЕ КОМУ НЕ СООБЩАТЬ КОД, НЕ ХРАНИТЬ ЕГО В ОТКРЫТЫХ ПЕРЕМЕННЫХ

select sign(

row\_to\_json(r), current\_setting('app.jwt\_secret')

) as token

from (select \_role as role,

$1 as login,

extract(epoch from now())::integer + 60 \* 60 \* 24 as exp) r

into result;

return result;

end;

$$ language plpgsql security definer;

CREATE TYPE user\_info AS

(

login text,

avatar\_url text,

role text

);

CREATE OR REPLACE FUNCTION get\_user\_info() RETURNS user\_info

LANGUAGE plpgsql AS

$$

DECLARE login\_ text := current\_setting('request.jwt.claims', true)::json->>'login';

DECLARE role text := current\_setting('request.jwt.claims', true)::json->>'role';

DECLARE result user\_info;

BEGIN

SELECT avatar INTO result.avatar\_url FROM auth.users WHERE login\_ = login;

result.role := role;

result.login := login\_;

return result;

END

$$ SECURITY DEFINER;

CREATE OR REPLACE FUNCTION add\_topology("order" varchar(80), family varchar(80) DEFAULT null, genus varchar(80) DEFAULT null,

kind varchar(80) DEFAULT null) RETURNS text AS

$$

BEGIN

PERFORM (SELECT get\_kind\_id(kind, get\_genus\_id(genus, get\_family\_id(family, get\_order\_id("order")))));

RETURN 'ok';

END

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION check\_auth() RETURNS text AS

$$

DECLARE

login\_ text := current\_setting('request.jwt.claims', true)::json ->> 'login';

role\_ text := current\_setting('request.jwt.claims', true)::json ->> 'role';

BEGIN

IF EXISTS(SELECT \* FROM auth.users u WHERE u.role = role\_ AND u.login = login\_) THEN

RETURN 'ok';

END IF;

RAISE sqlstate 'PT403' using message = 'Вы неавторизованы!',

DETAIL = 'Обновите токен',

HINT = 'Перезайдите в аккаунт или попробуйте обновить токен';

END

$$ LANGUAGE plpgsql SECURITY DEFINER ;

CREATE OR REPLACE VIEW basic\_view as

SELECT collection.id AS ID,

"CatalogueNumber",

collect\_id,

order\_.name as Отряд,

family.name as Семейство,

genus.name as Род,

kind.name as Вид,

age.name as Возраст,

sex.name as Пол,

vi.name as "Вауч. институт",

vouch\_id as "Ваучерный ID",

ST\_Y(point::geometry) as Latitude,

ST\_X(point::geometry) as Longtitude,

country.name as Страна,

region.name as Регион,

subregion.name as Субрегион,

geo\_comment as Геокомментарий,

CASE

WHEN year = 0 THEN null

WHEN day IS NULL AND month IS NULL THEN year::text

WHEN day IS NULL THEN concat\_ws('.', month, year)

ELSE concat\_ws('.', day, month, year)

END AS Дата,

rna as RNA,

comment AS "Комментарий",

string\_agg(concat(c.last\_name,

CASE WHEN c.first\_name IS NOT NULL THEN concat(' ', "left"(c.first\_name, 1), '.') ELSE '' END,

CASE WHEN c.second\_name IS NOT NULL THEN concat(' ', "left"(c.second\_name, 1), '.') ELSE '' END),

', ') AS "Коллекторы",

file\_url IS NOT NULL as "Файл"

FROM collection

JOIN kind on kind.id = collection.kind\_id

JOIN genus on genus.id = kind.genus\_id

JOIN family on family.id = genus.family\_id

JOIN "order" order\_ on order\_.id = family.order\_id

JOIN age on age.id = collection.age\_id

JOIN sex on collection.sex\_id = sex.id

JOIN voucher\_institute vi on vi.id = collection.vouch\_inst\_id

JOIN subregion on collection.subregion\_id = subregion.id

JOIN region on subregion.region\_id = region.id

JOIN country on region.country\_id = country.id

LEFT JOIN collector\_to\_collection ctc on collection.id = ctc.collection\_id

LEFT JOIN collector c on ctc.collector\_id = c.id -- Если нет автора, то запись должна быть

GROUP BY collection.id, order\_.name, family.name, genus.name, kind.name, age.name, sex.name, vi.name, country.name,

region.name, subregion.name

ORDER BY collection.id;

# ПРИЛОЖЕНИЕ В Листинг клиентского приложения

main.dart

import 'package:flutter/foundation.dart' show kIsWeb;

import 'package:flutter/material.dart';

import 'package:lab\_gui\_flutter/my\_app\_state.dart';

import 'package:lab\_gui\_flutter/screens/add\_collector\_dialog.dart';

import 'package:lab\_gui\_flutter/screens/add\_topology\_dialog\_old.dart';

import 'package:lab\_gui\_flutter/screens/auth.dart';

import 'package:lab\_gui\_flutter/screens/collection\_page.dart';

import 'package:lab\_gui\_flutter/screens/collectors\_page.dart';

import 'package:lab\_gui\_flutter/web\_settings.dart';

import 'package:provider/provider.dart';

import 'package:side\_sheet\_material3/side\_sheet\_material3.dart';

import 'package:flutter\_localizations/flutter\_localizations.dart';

import 'package:syncfusion\_localizations/syncfusion\_localizations.dart';

import 'color\_schemes.g.dart';

import 'screens/add\_item\_collection\_dialog.dart';

import 'screens/topology\_page.dart';

void main() {

if (kIsWeb) {

webSet();

}

runApp(const MainApp());

}

class MainApp extends StatelessWidget {

const MainApp({super.key});

@override

Widget build(BuildContext context) {

final appState = MyAppState();

appState.start(context: context);

return ChangeNotifierProvider(

create: (context) => appState,

child: MaterialApp(

theme: ThemeData(useMaterial3: true, colorScheme: lightColorScheme),

restorationScopeId: 'app',

darkTheme: ThemeData(useMaterial3: true, colorScheme: darkColorScheme),

home: const MainPage(),

localizationsDelegates: const [

GlobalMaterialLocalizations.delegate,

GlobalWidgetsLocalizations.delegate,

GlobalCupertinoLocalizations.delegate,

SfGlobalLocalizations.delegate

],

supportedLocales: const [Locale('ru')],

locale: const Locale('ru'),

),

);

}

}

class MainPage extends StatefulWidget {

const MainPage({super.key});

@override

State<MainPage> createState() => \_MainPageState();

}

class \_MainPageState extends State<MainPage> {

var selectedIndex = 0;

static const defaultElavation = 1.0;

var railVisible = true;

@override

Widget build(BuildContext context) {

var appState = context.watch<MyAppState>();

var theme = Theme.of(context);

var surfaceContainer = ElevationOverlay.applySurfaceTint(

theme.colorScheme.surface,

theme.colorScheme.surfaceTint,

defaultElavation);

Widget page;

Widget dialog = const Dialog();

switch (selectedIndex) {

case 0:

page = const CollectionPage();

dialog = const AddCollectionItemDialog(

isUpdate: false,

);

break;

case 1:

page = const TopologyPage(

selectableMode: false,

);

dialog = const AddTopologyDialog();

break;

case 2:

page = const CollectorsPage(selectableMode: false);

dialog = const AddCollector(isUpdate: false);

break;

default:

throw UnimplementedError("page not found");

}

return Scaffold(

appBar: AppBar(

backgroundColor: surfaceContainer,

leading: Container(

margin: const EdgeInsets.only(left: 13),

child: IconButton(

icon: const Icon(Icons.menu),

onPressed: () {

setState(() {

railVisible = !railVisible;

});

},

)),

centerTitle: true,

actions: <Widget>[

IconButton(

onPressed: () {

showModalSideSheet(context,

body: AuthScreen(theme: theme),

header: "Авторизация",

addActions: false,

addDivider: false,

addBackIconButton: true,

addCloseIconButton: true);

},

icon: const Icon(Icons.account\_circle))

],

title: const Text("Лаборатория геномики и палеогеномики")),

body: Row(

children: [

SafeArea(

child: Visibility(

visible: railVisible,

child: NavigationRail(

leading: appState.isAuth

? FloatingActionButton(

onPressed: () {

showDialog(

context: context,

builder: (context) {

return dialog;

});

},

child: const Icon(Icons.add),

)

: Container(),

selectedIndex: selectedIndex,

backgroundColor: surfaceContainer,

onDestinationSelected: (value) {

setState(() {

selectedIndex = value;

});

},

extended: false,

labelType: NavigationRailLabelType.all,

groupAlignment: 0,

destinations: const [

NavigationRailDestination(

icon: Icon(Icons.table\_rows\_outlined),

label: Text("Коллекция")),

NavigationRailDestination(

icon: Icon(Icons.account\_tree\_outlined),

label: Text("Топология")),

NavigationRailDestination(

icon: Icon(Icons.person\_4\_outlined),

label: Text("Коллекторы"))

],

))),

Expanded(

child: Container(

height: double.infinity,

color: surfaceContainer,

child: page,

),

)

],

));

}

}

my\_app\_state.dart

import 'package:flutter/material.dart';

import 'package:flutter\_fancy\_tree\_view/flutter\_fancy\_tree\_view.dart';

import 'package:lab\_gui\_flutter/models/collection\_data\_source.dart';

import 'package:lab\_gui\_flutter/models/collector\_data\_source.dart';

import 'package:shared\_preferences/shared\_preferences.dart';

import 'package:syncfusion\_flutter\_datagrid/datagrid.dart';

import 'models/base\_model.dart';

import 'models/collection\_item.dart';

import 'models/collector.dart';

import 'models/jwt.dart';

import 'repository.dart';

sealed class TableState {}

class Error extends TableState {}

class Loading extends TableState {}

class Loaded extends TableState {}

final BaseModel FATHER =

BaseModel(id: 0, name: "father", type: BaseModelsTypes.father);

class MyAppState extends ChangeNotifier {

// Авторизация

var isAuth = false;

String? token;

final DataGridController collectionController = DataGridController();

final DataGridController collectorController = DataGridController();

List<CollectionItem> collection = List.empty(growable: true);

List<Collector> collectors = List.empty(growable: true);

late CollectionDataSource collectionDataSource;

late CollectorDataSource collectorDataSource;

// Добавление изменение

BaseModel? selectedBaseModel;

List<Collector> selectedCollectors = List.empty(growable: true);

void start({required BuildContext context}) {

collectionDataSource =

CollectionDataSource(collectionItems: collection, context: context);

collectorDataSource = CollectorDataSource(collectors, context);

autoUpdate();

checkToken();

childrenTopologyMap[FATHER] = [];

treeController = TreeController(

roots: childrenProvider(FATHER), childrenProvider: childrenProvider);

loadOrders();

}

Future<void> resetSelected() async {

selectedBaseModel = null;

selectedCollectors = List.empty(growable: true);

}

/// Получение коллекторов по [id] записи коллекции

Future<void> setSelectedCollectorsById(int id) async {

selectedCollectors = await getCollectorsByColItemId(id);

notifyListeners();

}

/// Получение топологии по [id] записи коллекции

Future<void> setTopologyByColId(int id) async {

var colItem = await getCollectionItemById(id);

selectedBaseModel = await getBaseModelByNames(colItem.order ?? "",

colItem.family ?? "", colItem.genus ?? "", colItem.species ?? "");

notifyListeners();

}

/// Установка [selectedBaseModel]

void setSelectedBaseModel(BaseModel? baseModel) {

selectedBaseModel = baseModel;

notifyListeners();

}

/// Передача выбранных строк ([collectorsRows]) из таблицы Коллекторы

Future<void> setSelectedCollectors(List<DataGridRow> collectorsRows) async {

var collectors = await getCollectors();

selectedCollectors.clear();

for (var el in collectorsRows) {

selectedCollectors.add(collectors

.firstWhere((element) => element.id == el.getCells().first.value));

}

notifyListeners();

}

/// Проверка токена на сервере

Future<void> checkToken() async {

final SharedPreferences prefs = await SharedPreferences.getInstance();

token = prefs.getString("token");

if (token != null && token != "") {

try {

checkAuth(token!);

isAuth = true;

} on Exception {

await logout();

}

}

notifyListeners();

}

/// Авторизация пользователя по [login\_] и [password]

Future<void> auth(String login\_, String password) async {

Jwt jwt = await login(login\_, password);

token = jwt.token;

final prefs = await SharedPreferences.getInstance();

await prefs.setString("token", token!);

isAuth = true;

notifyListeners();

}

/// Выход пользователя

// TODO добавить на сервер чёрный список токенов

Future<void> logout() async {

token = null;

isAuth = false;

final prefs = await SharedPreferences.getInstance();

await prefs.setString("token", "");

notifyListeners();

}

TableState state = Loading();

// Для обновления всех таблиц

bool get isRestart => state == Loading();

Future<void> restartNow() async {

state = Loading();

notifyListeners();

await Future.delayed(const Duration(seconds: 1));

resetSelected();

try {

collection = await getCollection();

collectors = await getCollectors();

} on Exception catch (e) {

print(e);

state = Error();

notifyListeners();

return;

}

collectorDataSource.collectors = collectors;

collectionDataSource.updateCollectionItems(collection);

collectionDataSource.notifyListeners();

collectorDataSource.notifyListeners();

finishRestart();

}

Future<void> finishRestart() async {

state = Loaded();

notifyListeners();

}

Future<void> autoUpdate() async {

while (true) {

restartNow();

checkToken();

await Future.delayed(const Duration(minutes: 15));

}

}

Iterable<BaseModel> childrenProvider(BaseModel baseModel) {

return childrenTopologyMap[baseModel] ?? const Iterable.empty();

}

late final TreeController<BaseModel> treeController;

var loadingModels = <BaseModel>[];

BaseModel? selectableBaseModel;

Future<void> loadOrders() async {

var orders = await getOrders();

childrenTopologyMap[FATHER] = orders

.where((element) =>

element.name != null && element.name!.trim().isNotEmpty)

.toList();

treeController.roots = childrenProvider(FATHER);

}

Map<BaseModel, List<BaseModel>> childrenTopologyMap = {};

}

repository.dart

import 'dart:convert';

import 'package:http/http.dart' as http;

import 'package:lab\_gui\_flutter/models/base\_model.dart';

import 'package:lab\_gui\_flutter/models/collection\_dto.dart';

import 'package:lab\_gui\_flutter/models/collection\_item.dart';

import 'package:lab\_gui\_flutter/models/jwt.dart';

import 'package:lab\_gui\_flutter/my\_app\_state.dart';

import 'models/collector.dart';

import 'models/user.dart';

import 'models/voucher\_institute.dart';

const URL = "84.204.46.23:3000";

Future<Jwt> login(String login, String password) async {

var url = Uri.http(URL, 'rpc/login');

final response =

await http.post(url, body: {'login': login, 'pass': password});

if (response.statusCode == 200) {

print(response.body);

return Jwt.fromJson(response.body);

} else {

throw Exception("Bad login or password");

}

}

Future<bool> testRequest(String jwt) async {

var url = Uri.http(URL, "rpc/test");

final response = await http.post(url);

if (response.statusCode == 200) {

return true;

} else {

throw Exception("Bad login or password");

}

}

Future<bool> checkAuth(String jwt) async {

var url = Uri.http(URL, "rpc/check\_auth");

final response =

await http.post(url, headers: {"Authorization": "Bearer $jwt"});

print(response.body);

if (response.statusCode == 200) {

return true;

} else {

throw Exception("Bad login or password");

}

}

Future<List<CollectionItem>> getCollection() async {

var url = Uri.http(URL, 'basic\_view');

final response = await http.get(url);

if (response.statusCode == 200) {

Iterable l = json.decode(response.body);

List<CollectionItem> collection = List<CollectionItem>.from(

l.map((model) => CollectionItem.fromJson(model)));

return collection;

} else {

throw Exception("Network not found.");

}

}

Future<List<BaseModel>> getOrders() async {

var url = Uri.http(URL, 'order');

final response = await http.get(url);

if (response.statusCode == 200) {

Iterable l = json.decode(response.body);

try {

List<BaseModel> orders = List<BaseModel>.from(

l.map((model) => BaseModel.fromJson(model, BaseModelsTypes.order)));

return orders;

} catch (e) {

print(e.toString());

}

return [];

} else {

throw Exception("Network not found.");

}

}

Future<List<BaseModel>> getFamiliesById(BaseModel order) async {

var url = Uri.http(URL, 'family', {"order\_id": "eq.${order.id}"});

final response = await http.get(url);

if (response.statusCode == 200) {

Iterable l = json.decode(response.body);

List<BaseModel> orders = List<BaseModel>.from(l.map((model) =>

BaseModel.fromJson(model, BaseModelsTypes.family, parent: order)));

return orders;

} else {

throw Exception("Network not found.");

}

}

Future<List<BaseModel>> getGenusesById(BaseModel family) async {

var url = Uri.http(URL, 'genus', {"family\_id": "eq.${family.id}"});

final response = await http.get(url);

if (response.statusCode == 200) {

Iterable l = json.decode(response.body);

List<BaseModel> orders = List<BaseModel>.from(l.map((model) =>

BaseModel.fromJson(model, BaseModelsTypes.genus, parent: family)));

return orders;

} else {

throw Exception("Network not found.");

}

}

Future<List<BaseModel>> getKindsById(BaseModel genus) async {

var url = Uri.http(URL, 'kind', {"genus\_id": "eq.${genus.id}"});

final response = await http.get(url);

if (response.statusCode == 200) {

Iterable l = json.decode(response.body);

List<BaseModel> orders = List<BaseModel>.from(l.map((model) =>

BaseModel.fromJson(model, BaseModelsTypes.kind, parent: genus)));

return orders;

} else {

throw Exception("Network not found.");

}

}

Future<List<Collector>> getCollectors() async {

var url = Uri.http(URL, 'collector');

final response = await http.get(url);

if (response.statusCode == 200) {

Iterable l = json.decode(response.body);

List<Collector> orders = List<Collector>.from(

l.map((collector) => Collector.fromJson(collector)));

return orders;

} else {

throw Exception("Network not found.");

}

}

Future<int> getLastIdCollection() async {

var url = Uri.http(

URL, 'collection', {"select": "id", "limit": "1", "order": "id.desc"});

final response = await http.get(url);

if (response.statusCode == 200) {

Iterable l = json.decode(response.body);

return l.first["id"] as int;

} else {

throw Exception("Network not found.");

}

}

Future<User> getUserInfoByToken(String token) async {

var url = Uri.http(URL, "rpc/get\_user\_info");

final response =

await http.get(url, headers: {"Authorization": "Bearer $token"});

if (response.statusCode == 200) {

return User.fromJson(response.body);

} else {

throw Exception("Token is expired");

}

}

Future<List<VoucherInstitute>> getVoucherInstitute() async {

var url = Uri.http(URL, 'voucher\_institute');

final response = await http.get(url);

if (response.statusCode == 200) {

Iterable l = json.decode(response.body);

List<VoucherInstitute> vis = List<VoucherInstitute>.from(

l.map((vi) => VoucherInstitute.fromJson(vi)));

return vis;

} else {

throw Exception("Network not found.");

}

}

Future<void> addCollection({

required String age,

required String sex,

required String vauchInst,

required String vauchId,

required String dateCollect,

required List<List<String>> collectors,

required String token,

String? country,

String? region,

String? subregion,

String? geocomment,

String? comment,

String? point,

bool rna = false,

String? collectId,

String? order,

String? family,

String? genus,

String? kind,

}) async {

final url = Uri.http(URL, "rpc/add\_collection");

final body = {

'age': age,

'sex': sex,

'vauch\_inst': vauchInst,

'vauch\_id': vauchId,

'date\_collect': dateCollect,

'collectors':

'{${collectors.map((collector) => '{"${collector[0]}", "${collector[1]}", "${collector[2]}"}').join(', ')}}',

'rna': rna.toString(),

};

if (country != null) body['country'] = country;

if (region != null) body['region'] = region;

if (subregion != null) body['subregion'] = subregion;

if (geocomment != null) body['geocomment'] = geocomment;

if (comment != null) body['comment'] = comment;

if (point != null) body['point'] = point;

if (collectId != null) body["collect\_id"] = collectId;

if (order != null) body["order"] = order;

if (family != null) body["family"] = family;

if (genus != null) body["genus"] = genus;

if (kind != null) body["kind"] = kind;

print(body);

final response = await http

.post(url, body: body, headers: {"Authorization": "Bearer $token"});

print(response.body);

print(response.statusCode);

}

Future<void> updateCollection({

required int col\_id,

required String age,

required String sex,

required String vauchInst,

required String vauchId,

required String dateCollect,

required List<List<String>> collectors,

required String token,

String? country,

String? region,

String? subregion,

String? geocomment,

String? comment,

String? point,

bool rna = false,

String? collectId,

String? order,

String? family,

String? genus,

String? kind,

}) async {

final url = Uri.http(URL, "rpc/update\_collection\_by\_id");

final body = {

'col\_id': col\_id.toString(),

'age': age,

'sex': sex,

'vauch\_inst': vauchInst,

'vauch\_id': vauchId,

'date\_collect': dateCollect,

'collectors':

'{${collectors.map((collector) => '{"${collector[0]}", "${collector[1]}", "${collector[2]}"}').join(', ')}}',

'rna': rna.toString(),

};

if (country != null) body['country'] = country;

if (region != null) body['region'] = region;

if (subregion != null) body['subregion'] = subregion;

if (geocomment != null) body['geocomment'] = geocomment;

if (comment != null) body['comment'] = comment;

if (point != null) body['point'] = point;

if (collectId != null) body["collect\_id"] = collectId;

if (order != null) body["order"] = order;

if (family != null) body["family"] = family;

if (genus != null) body["genus"] = genus;

if (kind != null) body["kind"] = kind;

final response = await http

.post(url, body: body, headers: {"Authorization": "Bearer $token"});

print(response.body);

}

Future<CollectionItem> getCollectionItemById(int id) async {

var url = Uri.http(URL, "basic\_view", {"id": "eq.$id"});

final response = await http.get(url);

if (response.statusCode == 200) {

Iterable l = json.decode(response.body);

List<CollectionItem> collection = List<CollectionItem>.from(

l.map((model) => CollectionItem.fromJson(model)));

return collection[0];

}

throw Exception();

}

Future<CollectionDTO> getCollectionDtoById(int id) async {

var url = Uri.http(URL, "collection", {"id": "eq.$id"});

final response = await http.get(url);

if (response.statusCode == 200) {

Iterable l = json.decode(response.body);

List<CollectionDTO> collection = List<CollectionDTO>.from(

l.map((model) => CollectionDTO.fromJson(model)));

return collection[0];

}

throw Exception();

}

Future<Collector> getCollectorById(int id) async {

var url = Uri.http(URL, "collector", {"id": "eq.$id"});

final response = await http.get(url);

print(response.body);

if (response.statusCode == 200) {

Iterable l = json.decode(response.body);

List<Collector> collectors = List<Collector>.from(

l.map((collector) => Collector.fromJson(collector)));

return collectors[0];

}

throw Exception();

}

Future<List<Collector>> getCollectorsByColItemId(int id) async {

var url = Uri.http(URL, "collector\_to\_collection",

{"select": "collector\_id", "collection\_id": "eq.$id"});

final response = await http.get(url);

if (response.statusCode == 200) {

print(response.body);

Iterable l = json.decode(response.body);

List<int> collectorIds =

List<int>.from(l.map((e) => e["collector\_id"] as int));

List<Collector> collectors = List.empty(growable: true);

for (int collector\_id in collectorIds) {

collectors.add(await getCollectorById(collector\_id));

}

return collectors;

}

throw Exception();

}

Future<int> getOrderIdByName(String name) async {

var url = Uri.http(URL, "order", {"select": "id", "name": "eq.$name"});

final response = await http.get(url);

if (response.statusCode == 200) {

Iterable l = json.decode(response.body);

List<int> orderIds = List<int>.from(l.map((e) => e["id"] as int));

return orderIds[0];

}

throw Exception();

}

Future<int> getFamilyIdByName(String name, int orderId) async {

var url = Uri.http(URL, "family",

{"select": "id", "name": "eq.$name", "order\_id": "eq.$orderId"});

final response = await http.get(url);

if (response.statusCode == 200) {

Iterable l = json.decode(response.body);

List<int> familyIds = List<int>.from(l.map((e) => e["id"] as int));

return familyIds[0];

}

throw Exception();

}

Future<int> getGenusIdByName(String name, int familyId) async {

var url = Uri.http(URL, "genus",

{"select": "id", "name": "eq.$name", "family\_id": "eq.$familyId"});

final response = await http.get(url);

if (response.statusCode == 200) {

Iterable l = json.decode(response.body);

List<int> genusIds = List<int>.from(l.map((e) => e["id"] as int));

return genusIds[0];

}

throw Exception();

}

Future<int> getKindIdByName(String name, int genusId) async {

var url = Uri.http(URL, "kind",

{"select": "id", "name": "eq.$name", "genus\_id": "eq.$genusId"});

final response = await http.get(url);

if (response.statusCode == 200) {

Iterable l = json.decode(response.body);

List<int> kindIds = List<int>.from(l.map((e) => e["id"] as int));

return kindIds[0];

}

throw Exception();

}

Future<BaseModel> getBaseModelByNames(String orderName, String familyName,

String genusName, String kindName) async {

var orderId = await getOrderIdByName(orderName);

var familyId = await getFamilyIdByName(familyName, orderId);

var genusId = await getGenusIdByName(genusName, familyId);

var kindId = await getKindIdByName(kindName, genusId);

return (await getKindsById((await getGenusesById((await getFamiliesById(

(await getOrders())

.firstWhere((element) => element.id == orderId)))

.firstWhere((element) => element.id == familyId)))

.firstWhere((element) => element.id == genusId)))

.firstWhere((element) => element.id == kindId);

}

Future<int> getLastCollectorId() async {

var url = Uri.http(

URL, 'collector', {"select": "id", "limit": "1", "order": "id.desc"});

final response = await http.get(url);

if (response.statusCode == 200) {

Iterable l = json.decode(response.body);

return l.first["id"] as int;

} else {

throw Exception("Network not found.");

}

}

Future<void> addCollector(

String lastName, String firstName, String secondName, String token) async {

var url = Uri.http(URL, 'collector');

final body = {

'last\_name': lastName,

'first\_name': firstName,

'second\_name': secondName

};

final response = await http

.post(url, body: body, headers: {"Authorization": "Bearer $token"});

print(response.body);

}

Future<void> updateCollector(String lastName, String firstName,

String secondName, String token, int id) async {

var url = Uri.http(URL, 'collector', {"id": "eq.$id"});

final body = {

'last\_name': lastName,

'first\_name': firstName,

'second\_name': secondName

};

final response = await http

.patch(url, body: body, headers: {"Authorization": "Bearer $token"});

print(response.body);

}

final topology = [

BaseModelsTypes.order,

BaseModelsTypes.family,

BaseModelsTypes.genus,

BaseModelsTypes.kind

];

Future<void> addBaseModel(

BaseModel baseModel,

String name,

String token,

) async {

final headers = {"Authorization": "Bearer $token"};

var structure = {BaseModelsTypes.values[baseModel.type.index + 1].name: name};

var father = baseModel;

while ((father.type) != BaseModelsTypes.father) {

structure[father.type.name] = father.name ?? "";

father = father.parent ?? FATHER;

}

var url = Uri.http(URL, "rpc/add\_topology");

final response = await http.post(url, body: structure, headers: headers);

print(response.body);

}

Future<void> updateBaseModel(

BaseModel baseModel,

String newName,

String token,

) async {

final headers = {"Authorization": "Bearer $token"};

var url = Uri.http(URL, baseModel.type.name, {"id": "eq.${baseModel.id}"});

final response =

await http.patch(url, body: {"name": newName}, headers: headers);

print(response.body);

}

web\_settings.dart

import "dart:html";

/// Настройки для web версии приложении:

/// - отключение нажатия правой кнопки мыши

void webSet(){

window.document.onContextMenu.listen((evt) => evt.preventDefault());

}

color\_schemes.g.dart

import 'package:flutter/material.dart';

// Цветовые темы

const lightColorScheme = ColorScheme(

brightness: Brightness.light,

primary: Color(0xFF006590),

onPrimary: Color(0xFFFFFFFF),

primaryContainer: Color(0xFFC8E6FF),

onPrimaryContainer: Color(0xFF001E2E),

secondary: Color(0xFF4F606E),

onSecondary: Color(0xFFFFFFFF),

secondaryContainer: Color(0xFFD3E5F5),

onSecondaryContainer: Color(0xFF0B1D29),

tertiary: Color(0xFF63597C),

onTertiary: Color(0xFFFFFFFF),

tertiaryContainer: Color(0xFFE9DDFF),

onTertiaryContainer: Color(0xFF1F1635),

error: Color(0xFFBA1A1A),

errorContainer: Color(0xFFFFDAD6),

onError: Color(0xFFFFFFFF),

onErrorContainer: Color(0xFF410002),

background: Color(0xFFFCFCFF),

onBackground: Color(0xFF191C1E),

outline: Color(0xFF71787E),

onInverseSurface: Color(0xFFF0F0F3),

inverseSurface: Color(0xFF2E3133),

inversePrimary: Color(0xFF88CEFF),

shadow: Color(0xFF000000),

surfaceTint: Color(0xFF006590),

outlineVariant: Color(0xFFC1C7CE),

scrim: Color(0xFF000000),

surface: Color(0xFFF9F9FC),

onSurface: Color(0xFF191C1E),

surfaceVariant: Color(0xFFDDE3EA),

onSurfaceVariant: Color(0xFF41484D),

);

const darkColorScheme = ColorScheme(

brightness: Brightness.dark,

primary: Color(0xFF88CEFF),

onPrimary: Color(0xFF00344D),

primaryContainer: Color(0xFF004C6E),

onPrimaryContainer: Color(0xFFC8E6FF),

secondary: Color(0xFFB7C9D8),

onSecondary: Color(0xFF21323F),

secondaryContainer: Color(0xFF384956),

onSecondaryContainer: Color(0xFFD3E5F5),

tertiary: Color(0xFFCEC0E8),

onTertiary: Color(0xFF352B4B),

tertiaryContainer: Color(0xFF4B4163),

onTertiaryContainer: Color(0xFFE9DDFF),

error: Color(0xFFFFB4AB),

errorContainer: Color(0xFF93000A),

onError: Color(0xFF690005),

onErrorContainer: Color(0xFFFFDAD6),

background: Color(0xFF191C1E),

onBackground: Color(0xFFE2E2E5),

outline: Color(0xFF8B9198),

onInverseSurface: Color(0xFF191C1E),

inverseSurface: Color(0xFFE2E2E5),

inversePrimary: Color(0xFF006590),

shadow: Color(0xFF000000),

surfaceTint: Color(0xFF88CEFF),

outlineVariant: Color(0xFF41484D),

scrim: Color(0xFF000000),

surface: Color(0xFF111416),

onSurface: Color(0xFFC5C6C9),

surfaceVariant: Color(0xFF41484D),

onSurfaceVariant: Color(0xFFC1C7CE),

);

add\_collector\_dialog.dart

import 'package:flutter/material.dart';

import 'package:lab\_gui\_flutter/repository.dart';

import 'package:provider/provider.dart';

import '../my\_app\_state.dart';

class AddCollector extends StatefulWidget {

final bool isUpdate;

final int? updatableId;

const AddCollector({super.key, required this.isUpdate, this.updatableId});

@override

State<AddCollector> createState() => \_AddCollectorState();

}

class \_AddCollectorState extends State<AddCollector> {

final idController = TextEditingController();

final lastNameController = TextEditingController();

final firstNameController = TextEditingController();

final secondNameController = TextEditingController();

var canAdd = false;

Future<void> getInfo() async {

if (!widget.isUpdate) {

idController.text = (await getLastCollectorId()).toString();

return;

}

idController.text = widget.updatableId?.toString() ?? " ";

var collector = await getCollectorById(widget.updatableId ?? 0);

lastNameController.text = collector.lastName ?? "";

firstNameController.text = collector.firstName ?? "";

secondNameController.text = collector.secondName ?? "";

setState(() {

canAdd = true;

});

}

@override

void initState() {

getInfo();

super.initState();

}

@override

Widget build(BuildContext context) {

var appState = context.watch<MyAppState>();

return AlertDialog(

title: const Text("Добавление нового коллектора"),

icon: const Icon(Icons.add),

content: Column(mainAxisSize: MainAxisSize.min, children: [

TextField(

decoration:

const InputDecoration(border: OutlineInputBorder(), label: Text("ID")),

enabled: false,

controller: idController,

),

const SizedBox(

height: 20,

),

TextField(

decoration: const InputDecoration(

border: OutlineInputBorder(), label: Text("Фамилия")),

controller: lastNameController,

onChanged: (value) {

setState(() {

canAdd = value.trim().isNotEmpty;

});

},

),

const SizedBox(

height: 20,

),

TextField(

decoration:

const InputDecoration(border: OutlineInputBorder(), label: Text("Имя")),

controller: firstNameController,

),

const SizedBox(

height: 20,

),

TextField(

decoration: const InputDecoration(

border: OutlineInputBorder(), label: Text("Отчетство")),

controller: secondNameController,

)

]),

actions: [

FilledButton(

onPressed: canAdd

? () async {if (!widget.isUpdate) {

addCollector(

lastNameController.text,

firstNameController.text,

secondNameController.text,

appState.token!);

} else{

updateCollector(

lastNameController.text,

firstNameController.text,

secondNameController.text,

appState.token!, widget.updatableId ?? -1);

}

Navigator.pop(context);

}

: null,

child: Text(widget.isUpdate ? "Изменить" : "Добавить"))

],

);

}

}

add\_item\_collection\_dialog.dart

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'package:flutter\_map/flutter\_map.dart';

import 'package:lab\_gui\_flutter/my\_app\_state.dart';

import 'package:lab\_gui\_flutter/repository.dart';

import 'package:intl/intl.dart';

import 'package:lab\_gui\_flutter/screens/collectors\_page.dart';

import 'package:lab\_gui\_flutter/screens/topology\_page.dart';

import 'package:latlong2/latlong.dart';

import 'package:provider/provider.dart';

import 'package:osm\_nominatim/osm\_nominatim.dart';

import 'package:responsive\_grid/responsive\_grid.dart';

import '../models/voucher\_institute.dart';

class AddCollectionItemDialog extends StatefulWidget {

const AddCollectionItemDialog(

{super.key, required this.isUpdate, this.updatableId});

final bool isUpdate;

final int? updatableId;

@override

State<AddCollectionItemDialog> createState() =>

\_AddCollectionItemDialogState();

}

enum Gender { Unknown, Male, Female }

enum Age { adult, subadult, juvenil, Unknown }

enum MapMode { point, polygon, notSet }

class \_AddCollectionItemDialogState extends State<AddCollectionItemDialog> {

bool isUpdate = false;

var \_gender = Gender.Unknown;

var \_age = Age.Unknown;

var \_isRna = false;

List<VoucherInstitute> vauchInstitutes = [];

var menuIsVisible = false;

final idController = TextEditingController();

final numberController = TextEditingController();

final collectIdController = TextEditingController();

final dateController = TextEditingController();

final vauchController = TextEditingController();

final vauchIDController = TextEditingController();

final longtitudeController = TextEditingController();

final latitudeController = TextEditingController();

final countryController = TextEditingController();

final regionController = TextEditingController();

final subRegionController = TextEditingController();

final geoCommentController = TextEditingController();

final commentController = TextEditingController();

final dateFormat = DateFormat("dd.MM.yyyy");

// Работа с картой

MapMode mapMode = MapMode.notSet;

LatLng point = LatLng(59.938284, 30.302509);

final \_textFieldKey = GlobalKey(); // FLUTTER багу два года, что за дела

static String \_displayStringForOption(VoucherInstitute option) => option.name;

Future<void> getInfo() async {

// Получение идентифакторов и другой посредственной информации

vauchInstitutes = await getVoucherInstitute();

if (!widget.isUpdate) {

var id = await getLastIdCollection() + 1;

idController.text = id.toString();

numberController.text = "ZIN-TER-M-${id}";

return;

}

var id = widget.updatableId!;

idController.text = id.toString();

var collectionItem = await getCollectionItemById(id);

numberController.text = collectionItem.catalogueNumber ?? "";

vauchController.text = collectionItem.scientificInstitute ?? "";

vauchIDController.text = collectionItem.voucherId ?? "";

latitudeController.text = collectionItem.latitude?.toString() ?? "";

longtitudeController.text = collectionItem.longitude?.toString() ?? "";

countryController.text = collectionItem.country ?? "";

regionController.text = collectionItem.region ?? "";

subRegionController.text = collectionItem.subregion ?? "";

geoCommentController.text = collectionItem.geoComment ?? "";

commentController.text = collectionItem.comment ?? "";

\_isRna = collectionItem.rna ?? false;

setState(() {

point = LatLng(collectionItem.latitude ?? point.latitude,

collectionItem.longitude ?? point.longitude);

if (collectionItem.latitude != null && collectionItem.longitude != null) {

mapMode = MapMode.point;

mapController.move(point, 10);

}

});

var collectionDTO = await getCollectionDtoById(id);

try {

setState(() {

\_age = Age.values[collectionDTO.ageId ?? 0];

\_gender = Gender.values[collectionDTO.sexId ?? 0];

});

} on RangeError {

// Если года неправильные

}

}

final mapController = MapController();

Future<void> setPointState() async {

setState(() {

mapMode = MapMode.point;

point = LatLng(

double.parse(latitudeController.text),

double.parse(longtitudeController.text),

);

});

}

Future<void> getCountryInfo() async {

final reverseSearchResult = await Nominatim.reverseSearch(

lat: point.latitude, lon: point.longitude, language: "ru");

countryController.text = reverseSearchResult.address?["country"];

regionController.text = reverseSearchResult.address?["region"] ?? "";

subRegionController.text = reverseSearchResult.address?["county"] ?? "";

}

Future<void> updateItem(MyAppState appState) async {

String? pointStr;

if (longtitudeController.text.trim() != "" ||

latitudeController.text.trim() != "") {

pointStr = "Point(${point.latitude} ${point.longitude})";

}

final topology = appState.selectedBaseModel?.getFullTopology();

String? order;

String? family;

String? genus;

String? kind;

try {

order = topology?[0];

family = topology?[1];

genus = topology?[2];

kind = topology?[3];

} on RangeError {

// cool block

}

List<List<String>> collectors = List.empty(growable: true);

for (var collector in appState.selectedCollectors) {

collectors.add([

collector.lastName ?? "",

collector.firstName ?? "",

collector.secondName ?? ""

]);

}

await updateCollection(

col\_id: widget.updatableId!,

collectId: collectIdController.text.trim().isEmpty

? null

: collectIdController.text.trim(),

order: order,

family: family,

genus: genus,

kind: kind,

age: \_age.name,

sex: \_gender.name,

vauchInst: vauchController.text,

vauchId: vauchIDController.text,

point: pointStr,

country: countryController.text.trim().isEmpty

? null

: countryController.text.trim(),

region: regionController.text.trim().isEmpty

? null

: regionController.text.trim(),

subregion: subRegionController.text.trim().isEmpty

? null

: subRegionController.text.trim(),

geocomment: geoCommentController.text.trim().isEmpty

? null

: geoCommentController.text.trim(),

dateCollect: dateController.text,

comment: commentController.text.trim().isEmpty

? null

: commentController.text.trim(),

collectors: collectors,

token: appState.token!,

rna: \_isRna);

appState.restartNow();

}

Future<void> addNewItem(MyAppState appState) async {

String? pointStr;

if (longtitudeController.text.trim() != "" ||

latitudeController.text.trim() != "") {

pointStr = "Point(${point.latitude} ${point.longitude})";

}

final topology = appState.selectedBaseModel?.getFullTopology();

String? order;

String? family;

String? genus;

String? kind;

try {

order = topology?[0];

family = topology?[1];

genus = topology?[2];

kind = topology?[3];

} on RangeError {

// cool block

}

List<List<String>> collectors = List.empty(growable: true);

for (var collector in appState.selectedCollectors) {

collectors.add([

collector.lastName ?? "",

collector.firstName ?? "",

collector.secondName ?? ""

]);

}

await addCollection(

collectId: collectIdController.text.trim().isEmpty

? null

: collectIdController.text.trim(),

order: order,

family: family,

genus: genus,

kind: kind,

age: \_age.name,

sex: \_gender.name,

vauchInst: vauchController.text,

vauchId: vauchIDController.text,

point: pointStr,

country: countryController.text.trim().isEmpty

? null

: countryController.text.trim(),

region: regionController.text.trim().isEmpty

? null

: regionController.text.trim(),

subregion: subRegionController.text.trim().isEmpty

? null

: subRegionController.text.trim(),

geocomment: geoCommentController.text.trim().isEmpty

? null

: geoCommentController.text.trim(),

dateCollect: dateController.text,

comment: commentController.text.trim().isEmpty

? null

: commentController.text.trim(),

collectors: collectors,

token: appState.token!,

rna: \_isRna);

appState.restartNow();

}

@override

void initState() {

getInfo();

dateController.text = dateFormat.format(DateTime.now());

super.initState();

}

@override

Widget build(BuildContext context) {

var appState = context.watch<MyAppState>();

var theme = Theme.of(context);

final titleTextStyle =

TextStyle(color: theme.colorScheme.onSurfaceVariant, fontSize: 22);

return Dialog(

child: Container(

margin: const EdgeInsets.only(top: 14),

child: SingleChildScrollView(

child: Column(

mainAxisSize: MainAxisSize.min,

children: [

Center(

child:

Text("Добавление новой записи", style: titleTextStyle),

),

ResponsiveGridRow(

children: [

ResponsiveGridCol(

lg: 3,

md: 6,

sm: 12,

child: firstColumnInput(theme, context, appState),

),

ResponsiveGridCol(

lg: 3,

md: 6,

sm: 12,

child: secondColumnInput(context, theme)),

ResponsiveGridCol(lg: 6, md: 12, child: inputGeo()),

ResponsiveGridCol(

lg: 12, child: addButton(appState, context))

],

),

],

),

)));

}

Container addButton(MyAppState appState, BuildContext context) {

return Container(

padding: const EdgeInsets.all(14),

child: Align(

alignment: Alignment.centerRight,

child: FilledButton.icon(

onPressed: () {

if (widget.isUpdate) {

updateItem(appState);

Navigator.pop(context);

return;

}

addNewItem(appState);

Navigator.pop(context);

},

icon: const Icon(Icons.add),

label: Text(widget.isUpdate ? "Обновить" : "Добавить")),

),

);

}

Container inputGeo() {

return Container(

padding: const EdgeInsets.only(left: 14, right: 14),

child: ListView(

shrinkWrap: true,

children: [

AspectRatio(

aspectRatio: 5 / 3,

child: ClipRRect(

borderRadius: BorderRadius.circular(50),

child: FlutterMap(

options: MapOptions(

center: point,

maxZoom: 15,

onSecondaryTap: (tapPosition, point) {

latitudeController.text = point.latitude.toString();

longtitudeController.text = point.longitude.toString();

setPointState();

getCountryInfo();

},

),

mapController: mapController,

children: [

TileLayer(

urlTemplate:

'https://tile.openstreetmap.org/{z}/{x}/{y}.png',

userAgentPackageName: 'su.pank.su',

),

MarkerLayer(

markers: mapMode == MapMode.point

? [

Marker(

point: point,

builder: (context) => const FlutterLogo())

]

: [],

),

]),

)),

const SizedBox(

height: 19,

),

Row(

children: [

Expanded(

child: TextField(

keyboardType: TextInputType.number,

controller: latitudeController,

inputFormatters: [

FilteringTextInputFormatter.allow(

RegExp(r'(^-?\d\*\.?\d\*)'))

],

onEditingComplete: () {

if (countryController.text.isNotEmpty ||

regionController.text.isNotEmpty ||

subRegionController.text.isNotEmpty ||

latitudeController.text.isEmpty) {

return;

}

getCountryInfo();

},

onChanged: (value) {

if (countryController.text.isNotEmpty ||

regionController.text.isNotEmpty ||

subRegionController.text.isNotEmpty ||

longtitudeController.text.isEmpty) {

return;

}

setPointState();

},

decoration: const InputDecoration(

border: OutlineInputBorder(), labelText: 'Широта'))),

const SizedBox(

width: 22,

),

Expanded(

child: TextField(

keyboardType: TextInputType.number,

controller: longtitudeController,

onEditingComplete: () {

if (countryController.text.isNotEmpty ||

regionController.text.isNotEmpty ||

subRegionController.text.isNotEmpty ||

latitudeController.text.isEmpty) {

return;

}

getCountryInfo();

},

onChanged: (val) {

if (countryController.text.isNotEmpty ||

regionController.text.isNotEmpty ||

subRegionController.text.isNotEmpty ||

latitudeController.text.isEmpty) {

return;

}

setPointState();

},

inputFormatters: [

FilteringTextInputFormatter.allow(

RegExp(r'(^-?\d\*\.?\d\*)'))

],

decoration: const InputDecoration(

border: OutlineInputBorder(), labelText: 'Долгота')))

],

),

const SizedBox(

height: 16,

),

Row(

children: [

Expanded(

child: TextField(

controller: countryController,

decoration: const InputDecoration(

border: OutlineInputBorder(), labelText: 'Страна'))),

const SizedBox(

width: 22,

),

Expanded(

child: TextField(

controller: regionController,

decoration: const InputDecoration(

border: OutlineInputBorder(), labelText: 'Регион'))),

const SizedBox(

width: 22,

),

Expanded(

child: TextField(

controller: subRegionController,

decoration: const InputDecoration(

border: OutlineInputBorder(),

labelText: 'Субрегион')))

],

),

const SizedBox(

height: 16,

),

TextField(

controller: geoCommentController,

decoration: const InputDecoration(

border: OutlineInputBorder(),

labelText: 'Комментарий к геопозиции'))

],

),

);

}

// Нужно чуть-чуть исправить, некритично (ListView и сам по себе Column)

Container firstColumnInput(

ThemeData theme, BuildContext context, MyAppState appState) {

return Container(

padding: const EdgeInsets.only(left: 14, right: 14),

child: ListView(shrinkWrap: true, children: [

Column(

children: [

TextField(

controller: idController,

enabled: false,

decoration: const InputDecoration(

border: OutlineInputBorder(), labelText: 'ID'),

),

const SizedBox(

height: 4,

),

Container(

margin: const EdgeInsets.only(left: 16, right: 16),

child: Text("Число генерируется самостоятельно",

style: theme.textTheme.bodySmall

?.apply(color: theme.colorScheme.outline))),

const SizedBox(

height: 26,

),

TextField(

controller: numberController,

enabled: false,

decoration: const InputDecoration(

border: OutlineInputBorder(),

labelText: 'Номер в каталоге'),

),

const SizedBox(

height: 4,

),

Container(

margin: const EdgeInsets.only(left: 16, right: 16),

child: Text("Число генерируется самостоятельно по ID",

style: theme.textTheme.bodySmall

?.apply(color: theme.colorScheme.outline))),

const SizedBox(

height: 26,

),

TextField(

controller: collectIdController,

decoration: const InputDecoration(

border: OutlineInputBorder(), labelText: 'Collect ID'),

),

const SizedBox(

height: 26,

),

SizedBox(

width: double.infinity,

child: FilledButton.tonal(

onPressed: () {

showDialog(

context: context,

builder: (context) {

return const Dialog(

child: TopologyPage(

selectableMode: true,

));

});

},

child: const Text(

"Выбрать топологию",

))),

SizedBox(

height: 73,

child: Text(

appState.selectedBaseModel?.getFullTopology().join(" ") ??

"")),

SizedBox(

width: double.infinity,

child: FilledButton.tonal(

onPressed: () {

showDialog(

context: context,

builder: ((context) {

return const Dialog(

child: CollectorsPage(

selectableMode: true,

),

);

}));

},

child: const Text("Выбрать коллекторов"))),

SizedBox(

height: 92,

child: Text(appState.selectedCollectors

.map((e) => e.lastName)

.join(", ")))

],

),

]));

}

Container secondColumnInput(BuildContext context, ThemeData theme) {

return Container(

padding: const EdgeInsets.only(left: 14, right: 14),

child: ListView(

shrinkWrap: true,

children: [

TextField(

controller: dateController,

readOnly: true,

decoration: InputDecoration(

border: const OutlineInputBorder(),

labelText: 'Дата сбора',

suffixIcon: IconButton(

onPressed: () async {

DateTime? date = await showDatePicker(

context: context,

initialDate: DateTime.now(),

firstDate: DateTime(1700),

lastDate: DateTime.now(),

locale: const Locale('ru'),

initialEntryMode: DatePickerEntryMode.input);

if (date != null) {

dateController.text = dateFormat.format(date);

}

},

icon: const Icon(Icons.today))),

),

const SizedBox(

height: 22,

),

RawAutocomplete<VoucherInstitute>(

displayStringForOption: \_displayStringForOption,

optionsBuilder: (value) {

if (!menuIsVisible) {

return const Iterable.empty();

}

if (value.text.isNotEmpty) {

return vauchInstitutes.where((element) => element.name

.toLowerCase()

.contains(value.text.toLowerCase()));

}

return vauchInstitutes;

},

fieldViewBuilder: (BuildContext context,

TextEditingController fieldTextEditingController,

FocusNode fieldFocusNode,

VoidCallback onFieldSubmitted) {

return TextField(

key: \_textFieldKey,

focusNode: fieldFocusNode,

controller: fieldTextEditingController,

onChanged: (value) {

setState(() {

menuIsVisible = value.isNotEmpty;

});

},

decoration: InputDecoration(

border: const OutlineInputBorder(),

labelText: 'Вауч. инст',

suffixIcon: IconButton(

onPressed: () {

setState(() {

menuIsVisible = !menuIsVisible;

});

},

icon: const Icon(Icons.keyboard\_arrow\_down))),

);

},

// Существует баг с размером autocomplete, это попытка его исправить

optionsViewBuilder: (context, onSelected, options) {

final textFieldBox = \_textFieldKey.currentContext!

.findRenderObject() as RenderBox;

final textFieldWidth = textFieldBox.size.width;

return Material(

shape: const RoundedRectangleBorder(

borderRadius:

BorderRadius.vertical(bottom: Radius.circular(4.0)),

),

// Баг флаттера

child: SizedBox(

width: 200,

child: ListView(

padding: const EdgeInsets.all(8.0),

children: options

.map((VoucherInstitute option) => GestureDetector(

onTap: () {

onSelected(option);

},

child: ListTile(

title: Text(option.name),

),

))

.toList(),

)));

}),

const SizedBox(

height: 22,

),

TextField(

controller: vauchIDController,

decoration: const InputDecoration(

border: OutlineInputBorder(), labelText: 'Вауч. код')),

const SizedBox(

height: 22,

),

SizedBox(

width: double.infinity,

child: Text(

"Пол",

style: theme.textTheme.titleSmall?.apply(fontWeightDelta: 2),

textAlign: TextAlign.left,

)),

Column(

children: [

// Если можно обойтись без лишних циклов, то обойдёмся

ListTile(

title: const Text("Неизвестный"),

dense: true,

visualDensity: const VisualDensity(vertical: -3),

leading: Radio(

value: Gender.Unknown,

groupValue: \_gender,

onChanged: (value) {

setState(() {

\_gender = value!;

});

},

),

),

ListTile(

title: const Text("Мужской"),

dense: true,

visualDensity: const VisualDensity(vertical: -3),

leading: Radio(

value: Gender.Male,

groupValue: \_gender,

onChanged: (value) {

setState(() {

\_gender = value!;

});

},

),

),

ListTile(

title: const Text("Женский"),

dense: true,

visualDensity: const VisualDensity(vertical: -3),

leading: Radio(

value: Gender.Female,

groupValue: \_gender,

onChanged: (value) {

setState(() {

\_gender = value!;

});

},

),

)

],

),

SizedBox(

width: double.infinity,

child: Text(

"Возраст",

style: theme.textTheme.titleSmall?.apply(fontWeightDelta: 2),

textAlign: TextAlign.left,

)),

Column(

children: [

ListTile(

title: const Text("Неизвестный"),

dense: true,

visualDensity: const VisualDensity(vertical: -3),

leading: Radio(

value: Age.Unknown,

groupValue: \_age,

onChanged: (value) {

setState(() {

\_age = value!;

});

},

),

),

ListTile(

title: const Text("adult"),

dense: true,

visualDensity: const VisualDensity(vertical: -3),

leading: Radio(

value: Age.adult,

groupValue: \_age,

onChanged: (value) {

setState(() {

\_age = value!;

});

},

),

),

ListTile(

title: const Text("subadult"),

dense: true,

visualDensity: const VisualDensity(vertical: -3),

leading: Radio(

value: Age.subadult,

groupValue: \_age,

onChanged: (value) {

setState(() {

\_age = value!;

});

},

),

),

ListTile(

title: const Text("juvenil"),

dense: true,

visualDensity: const VisualDensity(vertical: -3),

leading: Radio(

value: Age.juvenil,

groupValue: \_age,

onChanged: (value) {

setState(() {

\_age = value!;

});

},

),

)

],

)

],

),

);

}

}

auth.dart

import 'package:flutter/material.dart';

import 'package:lab\_gui\_flutter/my\_app\_state.dart';

import 'package:lab\_gui\_flutter/repository.dart';

import 'package:provider/provider.dart';

import '../models/user.dart';

class AuthScreen extends StatefulWidget {

const AuthScreen({

super.key,

required this.theme,

});

final ThemeData theme;

@override

State<AuthScreen> createState() => \_AuthScreenState();

}

class \_AuthScreenState extends State<AuthScreen> {

final loginController = TextEditingController();

final passwordController = TextEditingController();

var isLoadingNow = false;

String? jwt;

Future<void> authNow(MyAppState appState) async {

setState(() {

isLoadingNow = true;

});

try {

await appState.auth(loginController.text, passwordController.text);

} on Exception {

isLoadingNow = false;

showDialog(

context: context,

builder: (context) {

return AlertDialog(

title: const Text("Ошибка входа"),

content: const Text("Неверный логин или пароль"),

actions: [

FilledButton(

onPressed: () {

Navigator.of(context).pop();

},

child: const Text("Понятно"))

],

);

});

}

setState(() {

isLoadingNow = false;

});

}

@override

void dispose() {

loginController.dispose();

passwordController.dispose();

super.dispose();

}

@override

Widget build(BuildContext context) {

var appState = context.watch<MyAppState>();

final loginComponent = LoginComponent(

theme: widget.theme,

loginController: loginController,

passwordController: passwordController,

authNow: authNow);

if (isLoadingNow) {

return const CircularProgressIndicator();

}

if (!appState.isAuth) {

return loginComponent;

}

return ProfileInfoComponent(

appState: appState,

);

}

}

class LoginComponent extends StatelessWidget {

const LoginComponent({

super.key,

required this.theme,

required this.loginController,

required this.passwordController,

required this.authNow,

});

final ThemeData theme;

final TextEditingController loginController;

final TextEditingController passwordController;

final Future<void> Function(MyAppState) authNow;

@override

Widget build(BuildContext context) {

var appState = context.watch<MyAppState>();

return Container(

margin: const EdgeInsets.only(left: 75, right: 75),

child: Column(children: [

const Text("Введите ваш логин и пароль для входа в систему"),

const SizedBox(

height: 23,

),

TextField(

controller: loginController,

decoration: const InputDecoration(

border: OutlineInputBorder(),

labelText: 'Логин',

),

),

const SizedBox(

height: 23,

),

TextField(

controller: passwordController,

obscureText: true,

decoration: const InputDecoration(

border: OutlineInputBorder(), labelText: 'Пароль'),

),

const SizedBox(

height: 4,

),

Container(

margin: const EdgeInsets.only(left: 16, right: 16),

child: Text(

"Пароль, от вашего аккаунта вам может поменять или выдать администратор",

style: theme.textTheme.bodySmall

?.apply(color: theme.colorScheme.onSurfaceVariant))),

const SizedBox(

height: 24,

),

Align(

alignment: Alignment.centerRight,

child: FilledButton(

onPressed: () {

authNow(appState);

},

child: const Text("Войти в аккаунт")),

),

]),

);

}

}

class ProfileInfoComponent extends StatefulWidget {

const ProfileInfoComponent({super.key, required this.appState});

final MyAppState appState;

@override

State<ProfileInfoComponent> createState() => \_ProfileInfoComponentState();

}

class \_ProfileInfoComponentState extends State<ProfileInfoComponent> {

User? user;

var isLoading = true;

Future<void> getUserInfo(String token) async {

try {

user = await getUserInfoByToken(token);

} on Exception {

widget.appState.logout();

}

setState(() {

isLoading = false;

});

}

@override

void initState() {

getUserInfo(widget.appState.token!);

super.initState();

}

@override

Widget build(BuildContext context) {

var appState = context.watch<MyAppState>();

var theme = Theme.of(context);

if (isLoading == true) {

return const Center(

child: CircularProgressIndicator(),

);

}

return Container(

margin: const EdgeInsets.only(left: 40, right: 28),

width: double.infinity,

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

mainAxisSize: MainAxisSize.max,

children: [

Container(

alignment: Alignment.topCenter,

width: 188,

height: 188,

child: ClipRRect(

borderRadius: BorderRadius.circular(1000),

child: Image(

image: NetworkImage(user?.avatarUrl ??

"https://sun9-3.userapi.com/impg/GhqbifLL9RXukQi9AJ6SObwsLr-rQ2rDfYWLkg/qyxTl5xOlYU.jpg?size=188x188&quality=96&sign=1749242e0d43b4eaf90e28a74cec3cd9&type=album"))),

),

Text(

user?.login ?? "Unknown",

style: theme.textTheme.headlineLarge,

),

// Text("Имя", style: theme.textTheme.headlineLarge),

Text(

user?.role ?? "Unknown",

style: theme.textTheme.labelLarge

?.apply(color: theme.colorScheme.surfaceTint),

),

Align(

alignment: Alignment.centerRight,

child: FilledButton(

onPressed: () async {

appState.logout();

},

child: const Text("Выйти из аккаунта")),

)

]),

);

}

}

collection\_page.dart

import 'package:context\_menus/context\_menus.dart';

import 'package:flutter/material.dart';

import 'package:lab\_gui\_flutter/screens/error\_indicator.dart';

import 'package:provider/provider.dart';

import 'package:syncfusion\_flutter\_datagrid/datagrid.dart';

import '../my\_app\_state.dart';

import 'loading\_indicator.dart';

import 'package:syncfusion\_flutter\_core/theme.dart';

class CollectionPage extends StatefulWidget {

const CollectionPage({super.key});

@override

State<CollectionPage> createState() => \_CollectionPageState();

}

class \_CollectionPageState extends State<CollectionPage> {

late Map<String, double> columnWidthsCollection = {

'id': double.nan,

'catalogueNumber': double.nan,

'collectId': double.nan,

'order': double.nan,

'family': double.nan,

'genus': double.nan,

'species': double.nan,

'age': double.nan,

'gender': double.nan,

'scientificInstitute': double.nan,

'voucherId': double.nan,

'latitude': double.nan,

'longitude': double.nan,

'country': double.nan,

'region': double.nan,

'subregion': double.nan,

'geoComment': double.nan,

'date': double.nan,

'rna': double.nan,

'comment': double.nan,

'stringAgg': double.nan,

};

@override

void initState() {

super.initState();

}

@override

Widget build(BuildContext context) {

final columns = [

GridColumn(

columnName: 'id',

width: columnWidthsCollection['id']!,

minimumWidth: 100,

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerRight,

child: const Text('ID'),

),

),

GridColumn(

columnName: 'catalogueNumber',

width: columnWidthsCollection['catalogueNumber']!,

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerLeft,

child: const Text('Номер в каталоге'),

),

),

GridColumn(

columnName: 'collectId',

width: columnWidthsCollection['collectId']!,

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerLeft,

child: const Text('Collect ID'),

),

),

GridColumn(

columnName: 'order',

width: columnWidthsCollection['order']!,

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerLeft,

child: const Text('Отряд'),

),

),

GridColumn(

columnName: 'family',

width: columnWidthsCollection['family']!,

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerLeft,

child: const Text('Семейство'),

),

),

GridColumn(

columnName: 'genus',

width: columnWidthsCollection['family']!,

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerLeft,

child: const Text('Род'),

),

),

GridColumn(

columnName: 'species',

width: columnWidthsCollection['species']!,

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerLeft,

child: const Text('Вид'),

),

),

GridColumn(

columnName: 'age',

width: columnWidthsCollection['age']!,

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerLeft,

child: const Text('Возраст'),

),

),

GridColumn(

columnName: 'gender',

width: columnWidthsCollection['gender']!,

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerLeft,

child: const Text('Пол'),

),

),

GridColumn(

columnName: 'scientificInstitute',

width: columnWidthsCollection['scientificInstitute']!,

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerLeft,

child: const Text('Ваучерный институт'),

),

),

GridColumn(

columnName: 'voucherId',

width: columnWidthsCollection['voucherId']!,

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerLeft,

child: const Text('Вауч. ID'),

),

),

GridColumn(

columnName: 'latitude',

width: columnWidthsCollection['latitude']!,

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerRight,

child: const Text('Latitude'),

),

),

GridColumn(

columnName: 'longitude',

width: columnWidthsCollection['longitude']!,

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerRight,

child: const Text('Longitude'),

),

),

GridColumn(

columnName: 'country',

width: columnWidthsCollection['country']!,

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerLeft,

child: const Text('Страна'),

),

),

GridColumn(

columnName: 'region',

width: columnWidthsCollection['region']!,

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerLeft,

child: const Text('Регион'),

),

),

GridColumn(

columnName: 'subregion',

width: columnWidthsCollection['subregion']!,

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerLeft,

child: const Text('Суб. регион'),

),

),

GridColumn(

columnName: 'geoComment',

width: columnWidthsCollection['geoComment']!,

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerLeft,

child: const Text('Гео-комментарий'),

),

),

GridColumn(

columnName: 'date',

width: columnWidthsCollection['date']!,

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerLeft,

child: const Text('Дата'),

),

),

GridColumn(

columnName: 'rna',

width: columnWidthsCollection['rna']!,

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerLeft,

child: const Text('RNA'),

),

),

GridColumn(

columnName: 'comment',

width: columnWidthsCollection['comment']!,

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerLeft,

child: const Text('Комментарий'),

),

),

GridColumn(

columnName: 'collectors',

width: columnWidthsCollection['stringAgg']!,

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerLeft,

child: const Text('Авторы'),

),

)

];

var appState = context.watch<MyAppState>();

var theme = Theme.of(context);

appState.collectionDataSource.context = context;

switch (appState.state) {

case Error():

return Center(child: ErrorIndicator(

buttonFNC: () {

appState.restartNow();

},

));

case Loading():

return const LoadingIndicator();

default:

break;

}

// Здесь необходим LayoutBuilder,

// потому что иначе flutter не видит обновлений и не хочет обновлять таблицу

// при изменении размеров окна

return LayoutBuilder(builder: (context, constraints) {

return ContextMenuOverlay(

child: ClipRRect(

borderRadius: BorderRadius.circular(0),

child: SfDataGridTheme(

data: SfDataGridThemeData(

headerColor: theme.colorScheme.primaryContainer,

selectionColor: theme.colorScheme.secondaryContainer),

child: SfDataGrid(

controller: appState.collectionController,

frozenColumnsCount: 1,

columnWidthMode: ColumnWidthMode.auto,

columnWidthCalculationRange:

ColumnWidthCalculationRange.visibleRows,

selectionMode: SelectionMode.multiple,

allowColumnsResizing: true,

allowFiltering: true,

allowSorting: true,

allowMultiColumnSorting: true,

columnResizeMode: ColumnResizeMode.onResizeEnd,

isScrollbarAlwaysShown: true,

columns: columns,

onColumnResizeUpdate: (ColumnResizeUpdateDetails details) {

if (details.width < 30) {

return false;

}

setState(() {

columnWidthsCollection[details.column.columnName] =

details.width;

});

return true;

},

source: appState.collectionDataSource,

))));

});

}

}

collectors\_page.dart

import 'package:context\_menus/context\_menus.dart';

import 'package:flutter/material.dart';

import 'package:lab\_gui\_flutter/models/collector.dart';

import 'package:lab\_gui\_flutter/screens/loading\_indicator.dart';

import 'package:provider/provider.dart';

import 'package:syncfusion\_flutter\_core/theme.dart';

import 'package:syncfusion\_flutter\_datagrid/datagrid.dart';

import '../my\_app\_state.dart';

import 'error\_indicator.dart';

class CollectorsPage extends StatefulWidget {

const CollectorsPage({super.key, required this.selectableMode});

final bool selectableMode;

@override

State<CollectorsPage> createState() => \_CollectorsPageState();

}

class \_CollectorsPageState extends State<CollectorsPage> {

List<Collector> selectedCollectors = List.empty(growable: true);

final List<GridColumn> columns = <GridColumn>[

GridColumn(

columnName: 'id',

maximumWidth: 100,

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerRight,

child: const Text('ID'),

),

),

GridColumn(

columnName: 'lastName',

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerLeft,

child: const Text('Фамилия'),

),

),

GridColumn(

columnName: 'firstName',

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerLeft,

child: const Text('Имя'),

),

),

GridColumn(

columnName: 'secondName',

label: Container(

padding: const EdgeInsets.all(16.0),

alignment: Alignment.centerLeft,

child: const Text('Отчетство'),

),

),

];

@override

Widget build(BuildContext context) {

var appState = context.watch<MyAppState>();

var theme = Theme.of(context);

appState.collectorDataSource.context = context;

switch (appState.state) {

case Error():

return Center(child: ErrorIndicator(

buttonFNC: () {

appState.restartNow();

},

));

case Loading():

return const LoadingIndicator();

default:

break;

}

return Stack(children: [

LayoutBuilder(builder: (context, constraints) {

return ContextMenuOverlay(

child: SfDataGridTheme(

data: SfDataGridThemeData(

headerColor: theme.colorScheme.primaryContainer,

selectionColor: theme.colorScheme.secondaryContainer),

child: SfDataGrid(

columnWidthMode: ColumnWidthMode.fill,

allowFiltering: true,

allowSorting: true,

allowMultiColumnSorting: true,

controller: appState.collectorController,

selectionMode: SelectionMode.multiple,

source: appState.collectorDataSource,

columns: columns)));

}),

widget.selectableMode

? Container(

margin: const EdgeInsets.all(70),

child: Align(

alignment: Alignment.bottomRight,

child: FilledButton(

onPressed: true

? () {

appState.setSelectedCollectors(

appState.collectorController.selectedRows);

Navigator.pop(context);

}

: null,

child: const Text("Подтвердить"))))

: const Text("")

]);

}

}

error\_indicator.dart

import 'package:flutter/material.dart';

class ErrorIndicator extends StatefulWidget {

// Действие при нажатии на кнопку повторения попытки

final void Function() buttonFNC;

const ErrorIndicator({super.key, required this.buttonFNC});

@override

State<ErrorIndicator> createState() => \_ErrorIndicatorState();

}

class \_ErrorIndicatorState extends State<ErrorIndicator> {

@override

Widget build(BuildContext context) {

return Column(

mainAxisSize: MainAxisSize.min,

children: [

const SizedBox(

width: 300,

child: Text(

"Подключение к серверу отсутствует, проверьте ваше подключение и повторите попытку.",

textAlign: TextAlign.center,

)),

const SizedBox(

height: 10,

),

FilledButton(

onPressed: widget.buttonFNC, child: const Text("Повторить попытку"))

],

);

}

}

loading\_indicator.dart

import 'package:flutter/material.dart';

class LoadingIndicator extends StatelessWidget {

const LoadingIndicator({

super.key,

});

@override

Widget build(BuildContext context) {

return const Center(

child: Column(mainAxisAlignment: MainAxisAlignment.center, children: [

CircularProgressIndicator(),

SizedBox(

height: 10,

),

Text("Подождите, происходит загрузка данных...")

]));

}

}

topology\_page.dart

import 'package:context\_menus/context\_menus.dart';

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'package:flutter\_fancy\_tree\_view/flutter\_fancy\_tree\_view.dart';

import 'package:lab\_gui\_flutter/my\_app\_state.dart';

import 'package:lab\_gui\_flutter/screens/add\_topology\_dialog\_old.dart';

import 'package:provider/provider.dart';

import '../models/base\_model.dart';

import '../repository.dart';

class TopologyPage extends StatefulWidget {

const TopologyPage({super.key, required this.selectableMode});

final bool selectableMode;

@override

State<TopologyPage> createState() => \_TopologyPageState();

}

class \_TopologyPageState extends State<TopologyPage> {

final topology = {

BaseModelsTypes.father: getOrders,

BaseModelsTypes.order: getFamiliesById,

BaseModelsTypes.family: getGenusesById,

BaseModelsTypes.genus: getKindsById,

};

@override

void initState() {

super.initState();

}

bool isAdding = false;

BaseModel? addedParent;

bool isEditing = false;

late BaseModel editedBaseModel;

TextEditingController \_textEditingController = TextEditingController();

Widget getLeading(BaseModel baseModel) {

var appState = context.watch<MyAppState>();

if (appState.loadingModels.contains(baseModel)) {

return const CircularProgressIndicator();

}

late final VoidCallback? onPressed;

late final bool? isOpen;

final List<BaseModel>? children = appState.childrenTopologyMap[baseModel];

if (baseModel.name != null &&

baseModel.type != BaseModelsTypes.kind &&

children == null) {

isOpen = false;

onPressed = () async {

appState.loadingModels.add(baseModel);

appState.notifyListeners();

var list = await topology[baseModel.type]!(baseModel);

appState.childrenTopologyMap[baseModel] = list

.where((element) =>

element.name != null && element.name!.trim().isNotEmpty)

.toList();

appState.loadingModels.remove(baseModel);

appState.treeController.expand(baseModel);

};

} else if (baseModel.type == BaseModelsTypes.kind ||

baseModel.name == null) {

isOpen = null;

onPressed = null;

} else if (children!.isEmpty) {

appState.childrenTopologyMap[baseModel] = [

BaseModel(

id: -1,

name: null,

type: BaseModelsTypes.values[baseModel.type.index + 1],

parent: baseModel)

];

isOpen = appState.treeController.getExpansionState(baseModel);

onPressed = () => appState.treeController.toggleExpansion(baseModel);

} else {

isOpen = appState.treeController.getExpansionState(baseModel);

onPressed = () => appState.treeController.toggleExpansion(baseModel);

}

return FolderButton(

isOpen: isOpen,

onPressed: onPressed,

);

}

bool firstSelect = false;

@override

Widget build(BuildContext context) {

var appState = context.watch<MyAppState>();

if (widget.selectableMode && !firstSelect) {

setState(() {

appState.selectableBaseModel = appState.selectedBaseModel;

firstSelect = true;

});

}

return Stack(children: [

ContextMenuOverlay(

child: AnimatedTreeView(

treeController: appState.treeController,

nodeBuilder: (context, entry) {

var parent = entry.node.type == BaseModelsTypes.order

? FATHER

: entry.node.parent;

String title =

baseModelToTopologyName[entry.node.type]?.toLowerCase() ??

"";

if (entry.node.id == -1) {

return addingButton(entry, parent, title, context, appState);

}

if (isEditing && editedBaseModel == entry.node) {

return Row(children: [

SizedBox(

width: entry.level \* 40,

),

SizedBox(

width: 250,

child: TextField(

controller: \_textEditingController,

decoration: InputDecoration(

border: const OutlineInputBorder(),

label: Text("Название ${title}а"))),

),

IconButton(

onPressed: () async {

if (\_textEditingController.text.trim() == "") {

showDialog(

context: context,

builder: (context) {

return const AlertDialog(

title: Text('Ошибка'),

content: Text(

"Поле пустое, пожалуйста, внимательно вводите даннные."),

);

});

return;

}

await updateBaseModel(entry.node,

\_textEditingController.text, appState.token!);

await reloadFather(parent, appState);

},

icon: const Icon(Icons.done))

]);

}

return ContextMenuRegion(

contextMenu: GenericContextMenu(buttonConfigs: [

ContextMenuButtonConfig("Изменить", onPressed: () {

setState(() {

\_textEditingController.text = entry.node.name!;

isEditing = true;

editedBaseModel = entry.node;

});

})

]),

child: Column(

mainAxisAlignment: MainAxisAlignment.start,

children: [

TreeIndentation(

entry: entry,

child: Row(

children: [

getLeading(entry.node),

widget.selectableMode

? Container(

color: appState.selectableBaseModel !=

null &&

appState.selectableBaseModel! ==

entry.node

? Theme.of(context)

.colorScheme

.primaryContainer

: Colors.transparent,

child: GestureDetector(

child: Text(entry.node.name ?? ""),

onTap: () {

setState(() {

appState.selectableBaseModel =

entry.node;

});

},

))

: Text(entry.node.name ?? "")

],

)),

appState.isAuth &&

appState.childrenTopologyMap[parent]!

.indexOf(entry.node) ==

appState.childrenTopologyMap[parent]!

.length -

1

? addingButton(

entry, parent, title, context, appState)

: const SizedBox()

],

));

})),

widget.selectableMode

? Container(

margin: const EdgeInsets.all(70),

child: Align(

alignment: Alignment.bottomRight,

child: FilledButton(

onPressed: appState.selectableBaseModel != null

? () {

appState.setSelectedBaseModel(

appState.selectableBaseModel);

Navigator.pop(context);

}

: null,

child: const Text("Подтвердить"))))

: const Text("")

]);

}

Future<void> reloadFather(BaseModel? parent, MyAppState appState) async {

if (parent != FATHER) appState.treeController.collapse(parent!);

appState.loadingModels.add(parent!);

appState.notifyListeners();

List<BaseModel> list;

if (parent != FATHER) {

list = await topology[parent.type]!(parent);

appState.childrenTopologyMap[parent] = list

.where((element) =>

element.name != null && element.name!.trim().isNotEmpty)

.toList();

} else {

appState.loadOrders();

}

appState.loadingModels.remove(parent);

if (parent != FATHER) appState.treeController.expand(parent);

\_textEditingController.text = "";

setState(() {

isEditing = false;

});

appState.notifyListeners();

}

Column addingButton(TreeEntry<BaseModel> entry, BaseModel? parent,

String title, BuildContext context, MyAppState appState) {

return Column(children: [

Row(children: [

SizedBox(

width: entry.level \* 40,

),

IconButton(

iconSize: 24,

onPressed: () {

setState(() {

isAdding = true;

addedParent = parent;

});

},

icon: const Icon(Icons.add),

alignment: Alignment.topLeft,

),

const SizedBox(

width: 10,

),

LayoutBuilder(builder: (context, constraints) {

if (title == "семейств") {

return Text("Добавить новое ${title}о");

} else {

return Text("Добавить новый ${title}");

}

})

]),

isAdding && addedParent == parent

? Row(children: [

SizedBox(

width: entry.level \* 40,

),

SizedBox(

width: 250,

child: TextField(

controller: \_textEditingController,

decoration: InputDecoration(

border: const OutlineInputBorder(),

label: Text("Название ${title}а"))),

),

IconButton(

onPressed: () async {

if (\_textEditingController.text.trim() == "") {

showDialog(

context: context,

builder: (context) {

return const AlertDialog(

title: Text('Ошибка'),

content: Text(

"Поле пустое, пожалуйста, внимательно вводите даннные."),

);

});

return;

}

await addBaseModel(parent ?? FATHER,

\_textEditingController.text, appState.token!);

await reloadFather(parent, appState);

},

icon: const Icon(Icons.done))

])

: const SizedBox()

]);

}

}

base\_model.dart

enum BaseModelsTypes {

father,

order,

family,

genus,

kind,

}

/// Класс который работает с топологией

/// [id] - идентификатор

/// [name] - название

/// [type] - тип

/// [parent] - отец

class BaseModel {

int id;

String? name;

BaseModelsTypes type;

BaseModel? parent;

BaseModel(

{required this.id, required this.name, required this.type, this.parent});

factory BaseModel.fromJson(Map<String, dynamic> json, BaseModelsTypes type,

{BaseModel? parent}) {

return BaseModel(

id: json['id'] as int,

name: json['name'] as String?,

type: type,

parent: parent);

}

Map<String, dynamic> toJson() {

final Map<String, dynamic> data = <String, dynamic>{};

data['id'] = id;

data['name'] = name;

return data;

}

/// Получение списка топологии

List<String> getFullTopology() {

var first = this;

List<String> topology = List.empty(growable: true);

topology.add(first.name ?? "");

while (first.parent != null) {

first = first.parent!;

topology.add(first.name ?? "");

}

return topology.reversed.toList();

}

@override

String toString() {

return "$id $name";

}

@override

bool operator ==(dynamic other) =>

other != null &&

other is BaseModel &&

other.type == type &&

other.parent == parent &&

other.id == id &&

other.name == name;

}

collection\_data\_source.dart

import 'package:context\_menus/context\_menus.dart';

import 'package:flutter/material.dart';

import 'package:lab\_gui\_flutter/screens/add\_item\_collection\_dialog.dart';

import 'package:provider/provider.dart';

import 'package:syncfusion\_flutter\_datagrid/datagrid.dart';

import '../my\_app\_state.dart';

import 'collection\_item.dart';

import 'package:intl/intl.dart';

/// Источник данных для таблицы

class CollectionDataSource extends DataGridSource {

BuildContext context;

final List<CollectionItem> collectionItems;

final DateFormat formatter = DateFormat('dd.MM.y');

void updateCollectionItems(List<CollectionItem> collectionItems) {

context = context;

\_collectionItems = collectionItems

.map<DataGridRow>((item) => DataGridRow(cells: [

DataGridCell<int?>(columnName: 'id', value: item.id),

DataGridCell<String?>(

columnName: 'catalogueNumber', value: item.catalogueNumber),

DataGridCell<String?>(

columnName: 'collectId', value: item.collectId),

DataGridCell<String?>(columnName: 'order', value: item.order),

DataGridCell<String?>(columnName: 'family', value: item.family),

DataGridCell<String?>(columnName: 'genus', value: item.genus),

DataGridCell<String?>(columnName: 'species', value: item.species),

DataGridCell<String?>(columnName: 'age', value: item.age),

DataGridCell<String?>(columnName: 'gender', value: item.gender),

DataGridCell<String?>(

columnName: 'scientificInstitute',

value: item.scientificInstitute),

DataGridCell<String?>(

columnName: 'voucherId', value: item.voucherId),

DataGridCell<double?>(

columnName: 'latitude', value: item.latitude),

DataGridCell<double?>(

columnName: 'longitude', value: item.longitude),

DataGridCell<String?>(columnName: 'country', value: item.country),

DataGridCell<String?>(columnName: 'region', value: item.region),

DataGridCell<String?>(

columnName: 'subregion', value: item.subregion),

DataGridCell<String?>(

columnName: 'geoComment', value: item.geoComment),

DataGridCell<DateTime?>(columnName: 'date', value: item.date),

DataGridCell<bool?>(columnName: 'rna', value: item.rna),

DataGridCell<String?>(columnName: 'comment', value: item.comment),

DataGridCell<String?>(

columnName: 'collectors', value: item.collectors),

]))

.toList();

}

CollectionDataSource({required this.collectionItems, required this.context}) {

updateCollectionItems(collectionItems);

}

List<DataGridRow> \_collectionItems = [];

@override

List<DataGridRow> get rows => \_collectionItems;

@override

DataGridRowAdapter? buildRow(DataGridRow row) {

var appState = Provider.of<MyAppState>(context,

listen: false); // Проcлушивание не нужно

var collectionItem = appState.collection

.firstWhere((element) => element.id == row.getCells().first.value);

return DataGridRowAdapter(

cells: row.getCells().map<Widget>((dataGridCell) {

return ContextMenuRegion(

contextMenu: GenericContextMenu(buttonConfigs: [

ContextMenuButtonConfig("Изменить",

onPressed: !appState.isAuth

? null

: () {

appState.setSelectedCollectorsById(collectionItem.id);

appState.setTopologyByColId(collectionItem.id);

showDialog(

context: context,

builder: (context) {

return AddCollectionItemDialog(

isUpdate: true,

updatableId: collectionItem.id,

);

});

})

]),

child: LayoutBuilder(

builder: (context, constraints) {

if (dataGridCell.columnName == 'date') {

if (dataGridCell.value == null) return Container();

String formattedString;

switch (collectionItem.dateType) {

case DateType.all:

formattedString = formatter.format(dataGridCell.value);

break;

case DateType.mounthAndYear:

formattedString =

DateFormat("MM.y").format(dataGridCell.value);

case DateType.year:

formattedString =

DateFormat("y").format(dataGridCell.value);

}

return Container(

alignment: Alignment.centerRight,

padding: const EdgeInsets.all(16.0),

child: Text(formattedString),

);

}

if (dataGridCell.columnName == 'rna') {

return Container(

alignment: Alignment.center,

child: Checkbox(

value: dataGridCell.value,

onChanged: null,

),

);

}

if (dataGridCell.columnName == 'id') {

if (collectionItem.hasFile ?? false) {

return Container(

alignment: Alignment.center,

padding: const EdgeInsets.all(16.0),

child: Expanded(

child: Row(

mainAxisAlignment: MainAxisAlignment.spaceBetween,

children: [

const Icon(Icons.file\_present),

Text(dataGridCell.value?.toString() ?? '')

],

),

));

} else {

return Container(

alignment: Alignment.centerRight,

padding: const EdgeInsets.all(16.0),

child: Text(dataGridCell.value?.toString() ?? ''),

);

}

}

return Container(

alignment: (dataGridCell.columnName == 'latitude' ||

dataGridCell.columnName == 'longitude')

? Alignment.centerRight

: Alignment.centerLeft,

padding: const EdgeInsets.all(16.0),

child: Text(dataGridCell.value?.toString() ?? ''),

);

},

));

}).toList());

}

}

collection\_dto.dart

class CollectionDTO {

final int? id;

final String? catalogueNumber;

final String? collectId;

final int? kindId;

final int? subregionId;

final String? genBankId;

final String? point;

final int? vouchInstId;

final String? vouchId;

final bool? rna;

final int? sexId;

final int? ageId;

final int? day;

final int? month;

final int? year;

final String? comment;

final String? geoComment;

CollectionDTO({

this.id,

this.catalogueNumber,

this.collectId,

this.kindId,

this.subregionId,

this.genBankId,

this.point,

this.vouchInstId,

this.vouchId,

this.rna,

this.sexId,

this.ageId,

this.day,

this.month,

this.year,

this.comment,

this.geoComment,

});

factory CollectionDTO.fromJson(Map<String, dynamic> json) {

return CollectionDTO(

id: json['id'] as int?,

catalogueNumber: json['CatalogueNumber'] as String?,

collectId: json['collect\_id'] as String?,

kindId: json['kind\_id'] as int?,

subregionId: json['subregion\_id'] as int?,

genBankId: json['gen\_bank\_id'] as String?,

point: json['point'] as String?,

vouchInstId: json['vouch\_inst\_id'] as int?,

vouchId: json['vouch\_id'] as String?,

rna: json['rna'] as bool?,

sexId: json['sex\_id'] as int?,

ageId: json['age\_id'] as int?,

day: json['day'] as int?,

month: json['month'] as int?,

year: json['year'] as int?,

comment: json['comment'] as String?,

geoComment: json['geo\_comment'] as String?,

);

}

}

collection\_item.dart

import 'package:intl/intl.dart';

enum DateType { all, mounthAndYear, year }

class CollectionItem {

int id;

String? catalogueNumber;

String? collectId;

String? order;

String? family;

String? genus;

String? species;

String? age;

String? gender;

String? scientificInstitute;

String? voucherId;

double? latitude;

double? longitude;

String? country;

String? region;

String? subregion;

String? geoComment;

DateTime? date;

bool? rna;

String? comment;

String? collectors;

bool? hasFile;

DateType dateType;

CollectionItem(

{required this.id,

this.catalogueNumber,

this.collectId,

this.order,

this.family,

this.genus,

this.species,

this.age,

this.gender,

this.scientificInstitute,

this.voucherId,

this.latitude,

this.longitude,

this.country,

this.region,

this.subregion,

this.geoComment,

this.date,

this.rna,

this.comment,

this.collectors,

this.hasFile,

required this.dateType});

factory CollectionItem.fromJson(Map<String, dynamic> json) {

var dateType = DateType.all;

DateTime? date;

try {

date = DateFormat('d.M.y').parse(json['Дата']);

} on Error catch (\_) {

} on FormatException catch (\_) {

dateType = DateType.mounthAndYear;

try {

date = DateFormat('M.y').parse(json['Дата']);

} on Error catch (\_) {

} on FormatException catch (\_) {

dateType = DateType.year;

date = DateFormat('y').parse(json['Дата']);

}

}

return CollectionItem(

id: json['id'] as int,

catalogueNumber: json['CatalogueNumber'] as String?,

collectId: json['collect\_id'] as String?,

order: json['Отряд'] as String?,

family: json['Семейство'] as String?,

genus: json['Род'] as String?,

species: json['Вид'] as String?,

age: json['Возраст'] as String?,

gender: json['Пол'] as String?,

scientificInstitute: json['Вауч. институт'] as String?,

voucherId: json['Ваучерный ID'] as String?,

latitude: (json['latitude'] as num?)?.toDouble(),

longitude: (json['longtitude'] as num?)?.toDouble(),

country: json['Страна'] as String?,

region: json['Регион'] as String?,

subregion: json['Субрегион'] as String?,

geoComment: json['Геокомментарий'] as String?,

date: date,

rna: json['rna'] as bool?,

comment: json['Комментарий'] as String?,

collectors: json['Коллекторы'] as String?,

hasFile: json["Файл"] as bool,

dateType: dateType);

}

}

collector.dart

class Collector {

final int? id;

final String? lastName;

final String? firstName;

final String? secondName;

Collector({

this.id,

this.lastName,

this.firstName,

this.secondName,

});

factory Collector.fromJson(Map<String, dynamic> json) {

return Collector(

id: json['id'] as int?,

lastName: json['last\_name'] as String?,

firstName: json['first\_name'] as String?,

secondName: json['second\_name'] as String?,

);

}

}

collector\_data\_source.dart

import 'package:context\_menus/context\_menus.dart';

import 'package:flutter/material.dart';

import 'package:lab\_gui\_flutter/screens/add\_collector\_dialog.dart';

import 'package:syncfusion\_flutter\_datagrid/datagrid.dart';

import 'collector.dart';

class CollectorDataSource extends DataGridSource {

BuildContext context;

List<Collector> collectors;

CollectorDataSource(this.collectors, this.context);

@override

List<DataGridRow> get rows => collectors.map<DataGridRow>((collector) {

return DataGridRow(

cells: [

DataGridCell<int>(columnName: 'id', value: collector.id),

DataGridCell<String>(

columnName: 'lastName', value: collector.lastName),

DataGridCell<String>(

columnName: 'firstName', value: collector.firstName),

DataGridCell<String>(

columnName: 'secondName', value: collector.secondName),

],

);

}).toList();

@override

DataGridRowAdapter? buildRow(DataGridRow row) {

var collectorId = row.getCells().first.value as int;

return DataGridRowAdapter(

cells: row.getCells().map<Widget>((dataGridCell) {

return ContextMenuRegion(

contextMenu: GenericContextMenu(buttonConfigs: [

ContextMenuButtonConfig("Изменить", onPressed: () {

showDialog(

context: context,

builder: (context) {

return AddCollector(

isUpdate: true, updatableId: collectorId);

});

})

]),

child: Container(

alignment: (dataGridCell.columnName == 'id')

? Alignment.centerRight

: Alignment.centerLeft,

padding: const EdgeInsets.all(16.0),

child: Text((dataGridCell.value ?? "").toString()),

));

}).toList(),

);

}

}

jwt.dart

import 'dart:convert';

import 'package:collection/collection.dart';

class Jwt {

final String? token;

const Jwt({this.token});

@override

String toString() => 'Jwt(token: $token)';

factory Jwt.fromMap(Map<String, dynamic> data) => Jwt(

token: data['token'] as String?,

);

Map<String, dynamic> toMap() => {

'token': token,

};

/// `dart:convert`

///

/// Parses the string and returns the resulting Json object as [Jwt].

factory Jwt.fromJson(String data) {

return Jwt.fromMap(json.decode(data) as Map<String, dynamic>);

}

/// `dart:convert`

///

/// Converts [Jwt] to a JSON string.

String toJson() => json.encode(toMap());

@override

bool operator ==(Object other) {

if (identical(other, this)) return true;

if (other is! Jwt) return false;

final mapEquals = const DeepCollectionEquality().equals;

return mapEquals(other.toMap(), toMap());

}

@override

int get hashCode => token.hashCode;

}

user.dart

import 'dart:convert';

import 'package:collection/collection.dart';

class User {

final String? login;

final dynamic avatarUrl;

final String? role;

const User({this.login, this.avatarUrl, this.role});

@override

String toString() {

return 'User(login: $login, avatarUrl: $avatarUrl, role: $role)';

}

factory User.fromMap(Map<String, dynamic> data) => User(

login: data['login'] as String?,

avatarUrl: data['avatar\_url'] as dynamic,

role: data['role'] as String?,

);

Map<String, dynamic> toMap() => {

'login': login,

'avatar\_url': avatarUrl,

'role': role,

};

/// `dart:convert`

///

/// Parses the string and returns the resulting Json object as [User].

factory User.fromJson(String data) {

return User.fromMap(json.decode(data) as Map<String, dynamic>);

}

/// `dart:convert`

///

/// Converts [User] to a JSON string.

String toJson() => json.encode(toMap());

User copyWith({

String? login,

dynamic avatarUrl,

String? role,

}) {

return User(

login: login ?? this.login,

avatarUrl: avatarUrl ?? this.avatarUrl,

role: role ?? this.role,

);

}

@override

bool operator ==(Object other) {

if (identical(other, this)) return true;

if (other is! User) return false;

final mapEquals = const DeepCollectionEquality().equals;

return mapEquals(other.toMap(), toMap());

}

@override

int get hashCode => login.hashCode ^ avatarUrl.hashCode ^ role.hashCode;

}

voucher\_institute.dart

class VoucherInstitute {

int id;

String name;

VoucherInstitute({required this.id, required this.name});

factory VoucherInstitute.fromJson(Map<String, dynamic> json) {

return VoucherInstitute(

id: json['id'] as int,

name: json['name'] as String,

);

}

}

# ПРИЛОЖЕНИЕ Г Листинг выдачи прав пользователям

-- Настройка web\_anon - роль для гостей

create role web\_anon nologin;

grant usage on schema public to web\_anon;

grant select on all tables in schema public to web\_anon;

grant select on basic\_view to web\_anon;

grant execute on function public.login(text, text) to web\_anon;

-- работник лаборатории, может выполнять процедуры, что позволяет ему изменять базу данных

create user lab\_worker;

grant web\_anon to lab\_worker;

grant USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public to lab\_worker;

grant execute on function add\_collection(text, text, varchar, varchar, varchar, varchar, varchar, text, text, text,

geography, text, text, text, text, date, text, text[], boolean) TO lab\_worker;

grant execute on function update\_collection\_by\_id(int, text, text, varchar, varchar, varchar, varchar, varchar, text, text, text,

geography, text, text, text, text, date, text, text[], boolean) TO lab\_worker;

grant insert, update on all tables in schema public to lab\_worker;

revoke delete on all tables in schema public from lab\_worker;

REVOKE execute on function remove\_collection\_by\_id(int) FROM lab\_worker;

-- зав. лабораторией

create user head\_lab;

grant lab\_worker to head\_lab;

grant execute on all functions in schema public to head\_lab;

grant usage on schema auth to head\_lab;

grant insert, update, delete, select on all tables in schema public to head\_lab;

grant insert, update, delete, select on all tables in schema auth to head\_lab;

-- администратор бд - очень доверенное лицо, которое будет уверено в своих запросах

create user database\_admin;

grant all privileges on schema public to database\_admin;

grant all privileges on schema auth to database\_admin;

-- роль для ролевого менеджера - postgrest, с помощью запросов понимает какого пользователя надо использовать

create role authenticator noinherit login password 'abobapass';

grant web\_anon to authenticator;

grant lab\_worker to authenticator;

grant head\_lab to authenticator;

# ПРИЛОЖЕНИЕ Д Листинг конфигурационного файла

db-uri = "postgres://authenticator:abobapass@localhost:5432/lab\_base"

db-schemas = "public"

db-anon-role = "web\_anon"

jwt-secret = "Q5He86xPvYscMiZxQw29gy8YkbD7a4aMDH1hQFP"