

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ
по дисциплине «Введение в нереляционные СУБД»
Тема: Информационная система для IT компании

Студентка гр. 9304	_____	Аксёнова Е.А.
Студент гр. 9304	_____	Арутюнян В.В.
Студент гр. 9304	_____	Ламбин А.В.
Преподаватель	_____	Заславский М.М.

Санкт-Петербург
2022

ЗАДАНИЕ НА ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

Студентка Аксёнова Е.А.

Студент Арутюнян В.В.

Студент Ламбин А.В.

Группа 9304

Тема работы: Информационная система для IT компании

Содержание пояснительной записки:

- Аннотация
- Содержание
- Введение
- Качественные требования к решению
- Сценарии использования
- Модель данных
- Разработанное приложение
- Заключение
- Список использованных источников

Предполагаемый объем пояснительной записки:

Не менее 20 страниц.

Дата выдачи задания: 01.09.2022

Дата сдачи реферата: 22.12.2022

Дата защиты реферата: 22.12.2022

Студентка	_____	Аксёнова Е.А.
Студент	_____	Арутюнян В.В.
Студент	_____	Ламбин А.В.
Преподаватель	_____	Заславский М.М.

АННОТАЦИЯ

В данной работе разработана информационная система для IT компании, позволяющая работникам списывать отработанное время с указанием занятий, просматривать информацию о сотрудниках, формировать заказы необходимых документов. В системе используется следующий стек технологий: язык TypeScript, библиотека React, менеджер состояний Redux и библиотеку компонентов Material UI для frontend-части; язык Kotlin, фреймворк Spring Boot, интеграцию с БД Spring Data MongoDB и базу данных MongoDB для backend-части; Docker Compose для сборки проекта.

SUMMARY

In this work, an information system has been developed for an IT company that allows employees to write off hours worked with an indication of occupations, view information about employees, and form orders for the necessary documents. The system uses the following technology stack: the TypeScript language, the React library, the Redux state manager, and the Material UI component library for the frontend part; Kotlin language, Spring Boot framework, Spring Data MongoDB integration with database and MongoDB database for the backend part; Docker Compose to build the project.

СОДЕРЖАНИЕ

Введение	5
1. Качественные требования к решению	6
1.1. Текущие требования	6
2. Сценарии использования	7
2.1. Макет пользовательского интерфейса	7
2.2. Сценарии использования	7
2.3. Преобладающие операции	15
3. Модель данных	17
3.1. Нереляционная модель данных	17
3.2. Реляционный аналог модели данных	22
3.3. Сравнение моделей	27
4. Разработанное приложение	28
4.1. Описание приложения	28
4.2. Используемые технологии	28
Заключение	29
Список использованных источников	30
Приложение А. Макет пользовательского интерфейса	31
Приложение Б. Документация по сборке и развёртыванию приложения	32
Приложение В. Инструкция для пользователя	33
Приложение Г. Снимки экрана приложения	34
Приложение Д. Запросы к нереляционной базе данных	37
Приложение Е. Запросы к реляционной базе данных	43

ВВЕДЕНИЕ

Для автоматизации процессов ИТ компании используют разрозненные приложения, что увеличивает время, затрачиваемое на работу с ними. Создание единого места для позволит уменьшить время, затрачиваемое на работу с программным обеспечением управления проектами. Поэтому разработка системы, объединяющей функции необходимых приложений, является актуальной.

Цель работы – разработать ИС для ИТ компании, объединяющую некоторые необходимые функции.

Основные задачи:

1. Сформулировать основные сценарии использования приложения.
2. Разработать макет пользовательского интерфейса.
3. Разработать схему базы данных.
4. Подготовить прототип приложения.

В рамках данной работы необходимо реализовать web-приложение, предоставляющее несколько модулей: модуль Watcher для списания и просмотра отработанного времени с указанием занятий, модуль Person для поиска и просмотра информации о сотрудниках, модуль Document для формирования запросов на документы.

1. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

1.1. Текущие требования

Текущие требования к решению выглядят следующим образом:

1. Имеется страница авторизации. Пользователь вводит данные для входа, происходит аутентификация и в зависимости от роли пользователь получает определенные права.
2. Есть панель с различными модулями, предоставляющими определенный функционал. Например, Watcher – модуль, позволяющий списывать отработанное время с указанием занятий.
3. Для модулей будет предусмотрена возможность импортирования и экспортирования данных.
4. Модули предоставляют следующие возможности:
 - a. Watcher – модуль, позволяющий списывать отработанное время с указанием занятий. Также позволяет указывать и просматривать период больничных и отпусков. Для определенных ролей будет доступен просмотр этой информации о других сотрудниках.
 - b. Person – модуль, позволяющий просматривать общую информацию о сотрудниках (должность, часы работы, контактная информация и т.д.). Некоторым ролям будет предоставлен доступ на изменение информации о сотрудниках, добавления новых, а также просмотр конфиденциальных полей (проекты, размеры заработной платы и т.д.).
 - c. Document – модуль, позволяющий формировать отчёты о заработных платах, отработанных днях, доступности отпуска (накопленные дни).
5. Данные синтетические.

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2.1. Макет пользовательского интерфейса

Разработан макет пользовательского интерфейса, представленный в приложении А.

2.2. Сценарии использования

Сценарий использования «Авторизация»:

- Действующее лицо: Пользователь со стандартными правами.
- Начальные условия: Пользователь зашёл на сайт веб-приложения.
- Основной сценарий:
 1. Пользователь вводит логин и пароль в поля «Логин» и «Пароль» соответственно.
 2. Пользователь нажимает кнопку «Войти».
 3. В случае корректно введенных данных пользователь попадает на начальную страницу.
- Альтернативный сценарий:
 1. В случае некорректно введенных данных система отображает сообщение об этом.

Сценарий использования «Выход из аккаунта»:

- Действующее лицо: Пользователь со стандартными правами.
- Начальные условия: Пользователь авторизован.
- Основной сценарий:
 1. Пользователь нажимает на своё имя или аватар в шапке страницы.
 2. На странице отображается кнопка «Выход».
 3. В случае нажатия кнопки «Выход» пользователь попадает на страницу авторизации.
- Альтернативный сценарий:

1. При появлении кнопки в случае нажатия вне кнопки она пропадает.

Сценарий использования «Добавление задачи»:

- Действующее лицо: Пользователь со стандартными правами.
- Начальные условия: Пользователь авторизован.
- Основной сценарий:
 1. Пользователь нажимает на «Watcher» в шапке страницы.
 2. Отображается страница модуля Watcher.
 3. Пользователь нажимает на кнопку «+» у виджета с выбранным им днём.
 4. Открывается модальное окно с формой.
 5. Пользователь вводит необходимые данные.
 6. Пользователь нажимает кнопку «Ок».
 7. Система добавляет новую задачу, закрывает модальное окно и отображает списанное время в соответствующем виджете.
- Альтернативный сценарий:
 1. При нажатии пользователем кнопки «Отмена» новая задача добавлена не будет, модальное окно закроется.

Сценарий использования «Просмотр задач»:

- Действующее лицо: Пользователь со стандартными правами.
- Начальные условия: Пользователь авторизован.
- Основной сценарий:
 1. Пользователь нажимает на «Watcher» в шапке страницы.
 2. Отображается страница модуля Watcher.
 3. Пользователь нажимает на виджет с выбранным им днём.
 4. Открывается модальное окно со всеми имеющимися задачами, выполненными в этот день.

Сценарий использования «Изменение задачи»:

- Действующее лицо: Пользователь со стандартными правами.

- Начальные условия: Выполнен сценарий «Просмотр задач».
- Основной сценарий:
 1. У необходимой задачи пользователь нажимает на кнопку с шестерёнкой.
 2. Отображается модальное окно с формой и заполненными текущими данными.
 3. После изменений пользователь нажимает кнопку «Ок».
 4. Данные перезаписываются, модальное окно закрывается, списанное время пересчитывается.
- Альтернативный сценарий:
 1. При нажатии пользователем кнопки «Отмена» задача изменена не будет, модальное окно закроется.

Сценарий использования «Удаление задачи»:

- Действующее лицо: Пользователь со стандартными правами.
- Начальные условия: Выполнен сценарий «Просмотр задач».
- Основной сценарий:
 1. У необходимой задачи пользователь нажимает на кнопку с корзиной.
 2. Отображается модальное окно с предупреждением об удалении задачи.
 3. Пользователь нажимает кнопку «Ок».
 4. Данные удаляются, модальное окно закрывается, списанное время пересчитывается.
- Альтернативный сценарий:
 1. При нажатии пользователем кнопки «Отмена» данные удалены не будут, модальное окно закроется.

Сценарий использования «Выбор недели»:

- Действующее лицо: Пользователь со стандартными правами.
- Начальные условия: Пользователь находится в модуле Watcher.

– Основной сценарий:

1. При необходимости перейти на предыдущую неделю пользователь нажимает на стрелку влево. Переход к п.8.
2. При необходимости перейти на следующую неделю пользователь нажимает на стрелку вправо. Переход к п.8.
3. При необходимости вернуться на текущую неделю пользователь нажимает на кнопку «Текущая неделя». Переход к п.8.
4. В остальных случаях пользователь нажимает на поле с датами.
5. Появляется модальное окно с календарём.
6. При необходимости выбрать другой месяц пользователь перемещается с помощью стрелок (вправо – следующий месяц, влево – предыдущий месяц).
7. Пользователь выбирает необходимую неделю и нажимает на любой её день.
8. На странице отображается выбранная неделя.

– Альтернативный сценарий:

1. При появлении календаря в случае нажатия вне него календарь пропадает.

Сценарий использования «Просмотр дней отпуска»:

- Действующее лицо: Пользователь со стандартными правами.
- Начальные условия: Пользователь находится в модуле Watcher.
- Основной сценарий:

1. Пользователь нажимает на кнопку «Отпуск».
2. Открывается календарь с отмеченными днями отпуска.
3. Пользователь может переключаться по месяцам с помощью стрелок (вправо – следующий месяц, влево – предыдущий месяц).

Сценарий использования «Просмотр больничных дней»:

- Действующее лицо: Пользователь со стандартными правами.
- Начальные условия: Пользователь находится в модуле Watcher.
- Основной сценарий:
 1. Пользователь нажимает на кнопку «Больничный».
 2. Открывается календарь с отмеченными больничными днями.
 3. Пользователь может переключаться по месяцам с помощью стрелок (вправо – следующий месяц, влево – предыдущий месяц).

Сценарий использования «Изменение своих контактов»:

- Действующее лицо: Пользователь со стандартными правами.
- Начальные условия: Пользователь авторизован.
- Основной сценарий:
 1. Пользователь нажимает на «Person» в шапке страницы.
 2. Отображается страница модуля Person.
 3. Пользователь нажимает на кнопку «...» около необходимого поля («Телефон» или «Email»).
 4. Открывается модальное окно с полем для записи новых данных.
 5. Пользователь вводит данные и нажимает кнопку «Ок».
 6. Система обновляет данные, модальное окно закрывается.
- Альтернативный сценарий:
 1. Пользователь нажимает кнопку «Отмена», модальное окно закрывается.

Сценарий использования «Поиск сотрудника»:

- Действующее лицо: Пользователь со стандартными правами.
- Начальные условия: Пользователь находится в модуле Person.
- Основной сценарий:
 1. Пользователь вводит в строку поиска имя искомого сотрудника или его часть.

2. Отображается список сотрудников, удовлетворяющих данному запросу.
3. Пользователь нажимает на необходимого сотрудника.
4. Открывается страница с неконфиденциальными данными этого сотрудника.

Сценарий использования «Расширенный поиск сотрудника»:

- Действующее лицо: Пользователь со стандартными правами.
- Начальные условия: Пользователь находится в модуле Person.
- Основной сценарий:
 1. Пользователь нажимает на кнопку с настройками.
 2. Открывается страница с полями, по которым можно вести поиск.
 3. Пользователь вводит необходимые параметры.
 4. Система отображает найденных сотрудников.

Сценарий использования «Заказ справки»:

- Действующее лицо: Пользователь со стандартными правами.
- Начальные условия: Пользователь авторизован.
- Основной сценарий:
 1. Пользователь нажимает на «Document» в шапке страницы.
 2. Отображается страница модуля Document.
 3. Пользователь выбирает из списка необходимую ему справку.
 4. Пользователь нажимает кнопку «Заказать».
 5. Система добавляет запрос в очередь.

Сценарий использования «Просмотр заказанных справок»:

- Действующее лицо: Пользователь со стандартными правами.
- Начальные условия: Пользователь авторизован.
- Основной сценарий:
 1. Пользователь нажимает на «Document» в шапке страницы.
 2. Отображается страница модуля Document.

3. В списке отображаются все ранее заказанные справки.

Сценарий использования «Просмотр списанных часов другим пользователем»:

- Действующее лицо: Пользователь с расширенными правами.
- Начальные условия: Пользователь находится в модуле Watcher.
- Основной сценарий:
 1. Пользователь нажимает на кнопку с человечком внизу страницы.
 2. Отображается список сотрудников, у которых есть возможность посмотреть списанные часы.
 3. При необходимости пользователь может воспользоваться строкой поиска.

Сценарий использования «Добавление сотрудника»:

- Действующее лицо: Пользователь с расширенными правами.
- Начальные условия: Пользователь находится в модуле Person.
- Основной сценарий:
 1. Пользователь нажимает на кнопку «+» внизу страницы.
 2. Отображается модальное окно с формой записи нового сотрудника.
 3. Заполнив поля, пользователь нажимает кнопку «Ок».
 4. Система добавляет пользователя, модальное окно закрывается.
- Альтернативный сценарий:
 1. При заполнении данных пользователь нажимает кнопку «Отмена», модальное окно закрывается.

Сценарий использования «Изменение информации о сотруднике»:

- Действующее лицо: Пользователь с расширенными правами.
- Начальные условия: Пользователь находится в модуле Person.
- Основной сценарий:

1. Пользователь нажимает на кнопку «...» около необходимого свойства.
 2. Отображается модальное окно с полем для изменения.
 3. Пользователь нажимает кнопку «Ок».
 4. Система изменяет данные, модальное окно закрывается.
- Альтернативный сценарий:
1. Пользователь нажимает кнопку «Отмена», модальное окно закрывается.

Сценарий использования «Удаление сотрудника»:

- Действующее лицо: Пользователь с расширенными правами.
- Начальные условия: Пользователь находится в модуле Person.
- Основной сценарий:
1. Пользователь нажимает на кнопку с корзиной.
 2. Отображается модальное окно с предупреждением об удалении.
 3. Пользователь нажимает кнопку «Ок».
 4. Система удаляет пользователя, модальное окно закрывается.
- Альтернативный сценарий:
1. Пользователь нажимает кнопку «Отмена», модальное окно закрывается.

Сценарий использования «Просмотр всех заказанных справок»:

- Действующее лицо: Пользователь с расширенными правами.
- Начальные условия: Пользователь находится в модуле Document.
- Основной сценарий:
1. Пользователь нажимает на кнопку с колокольчиком внизу страницы.
 2. Открывается страница со всеми заказанными справками.
 3. При необходимости пользователь может отсортировать и настроить выборку по всем справкам.

Сценарий использования «Изменение статуса заказа»:

- Действующее лицо: Пользователь с расширенными правами.
- Начальные условия: Выполнен сценарий «Просмотр всех заказанных справок».
- Основной сценарий:
 1. Для конкретной справки пользователь выбирает статус из списка всех имеющихся статусов.
 2. Система меняет статус заказа.

Сценарий использования «Массовый импорт / экспорт данных»:

- Действующее лицо: Пользователь с расширенными правами.
- Начальные условия: Пользователь находится в любом из модулей.
- Основной сценарий:
 1. Пользователь нажимает на соответствующую кнопку в правой нижней части страницы.
 2. Веб-приложение предлагает выбрать файл / местоположение файла для импорта / экспорта.
 3. Система импортирует / экспортирует все данные, относящиеся к конкретному модулю.

2.3. Преобладающие операции

В модуле Watcher преобладают операции на чтение, так как в среднем новые записи добавляются одним пользователем, а читаются несколькими: непосредственно самим пользователем, менеджером и HR.

Модуль Person предназначен для поиска работников и просмотра информации о них. В данном случае, будут использованы операции чтения данных. Также данный модуль позволяет менять контактную информацию о себе, менять любые данные (если имеются права на это), добавлять новых пользователей. Однако таких запросов должно быть много меньше, поэтому преобладающей операцией будет чтение данных.

В модуле Document преобладают операции на чтение, так как независимо от создания / изменения документа пользователь может многократно читать данные, проверяя изменение статуса.

3. МОДЕЛЬ ДАННЫХ

3.1. Нереляционная модель данных

Разработана схема нереляционной базы данных (рисунок 1).

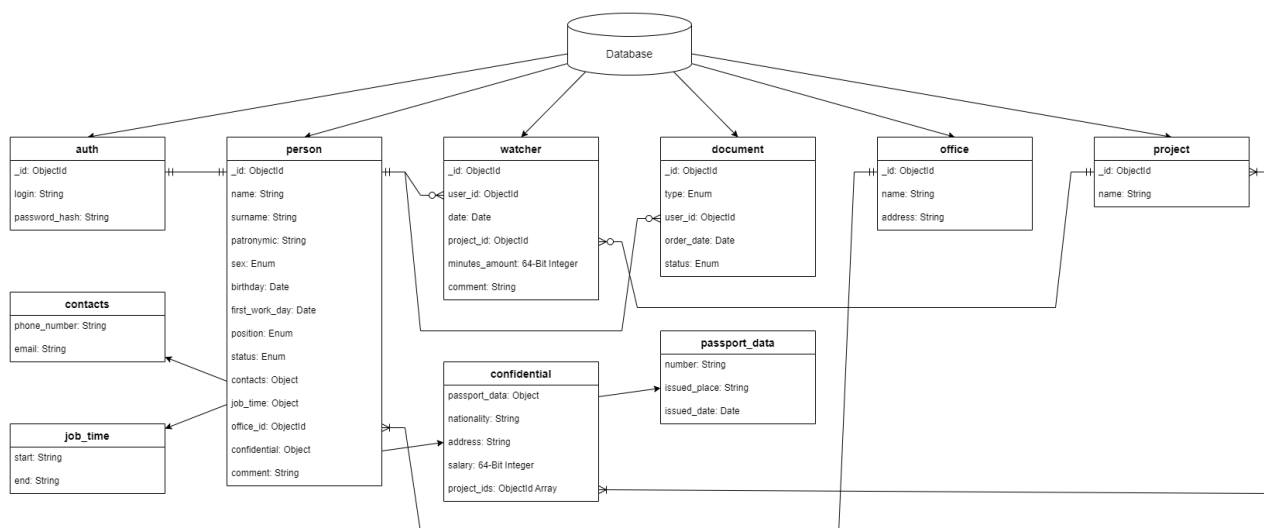


Рисунок 1 – Графическое представление нереляционной базы данных

Нижу приведены описания коллекций. Поскольку строки в MongoDB представлены в кодировке UTF-8, то латинские буквы и символы кодируются 1 байтом, а кириллические буквы – 2 байтами.

Коллекция *auth*, содержащая информацию для аутентификации (один документ занимает 59 байт):

- *_id: ObjectId* – идентификатор пользователя (12 байт);
- *login: String* – логин пользователя (до 15 символов или 15 байт);
- *password_hash: String* – хэш-код пароля (32 символов или 32 байт).

Ниже представлен пример документа коллекции *auth*:

```
{
  _id: ObjectId("6359c5e317df7f8644c2cea2"),
  login: "your_aksioma",
  password_hash: "8BeNwoHhivE2zbcRhHnX9vqcAZWP1MnV"
}
```

Коллекция *person*, содержащая информацию о работнике (один документ занимает 2352 байт):

- *_id: ObjectId* – идентификатор пользователя (12 байт);
- *name: String* – имя (до 40 символов или 80 байт);

- *surname: String* – фамилия (до 40 символов или 80 байт);
- *patronymic: String* – отчество (до 40 символов или 80 байт);
- *sex: Enum* – пол пользователя (6 байт);
- *birthday: Date* – дата рождения (8 байт);
- *first_work_date: Date* – дата первого рабочего дня (8 байт);
- *position: Enum* – должность (25 байт);
- *status: Enum* – статус (11 байт);
- *contacts:*
 - *phone_number: String* – номер телефона (до 20 символов или 20 байт);
 - *email: String* – электронная почта (до 100 символов или 100 байт);
- *job_time:*
 - *start: String* – начало рабочего дня (5 символов или 5 байт);
 - *end: String* – конец рабочего дня (5 символов или 5 байт);
- *office_id: ObjectId* – идентификатор офиса (12 байт);
- *confidential:*
 - *passport_data:*
 - *number: String* – серия и номер паспорта (до 20 символов или 40 байт);
 - *issued_place: String* – место выдачи (до 100 символов или 200 байт);
 - *issued_date: Date* – дата выдачи (8 байт);
 - *nationality: String* – гражданство (до 50 символов или 100 байт);
 - *address: String* – адрес пользователя (до 200 символов или 400 байт);
 - *salary: 64-Bit Integer* – заработная плата (8 байт);

- *project_ids: ObjectId Array* – идентификаторы проектов (12n байт, где $n \leq 10$);
- *comment: String* – комментарий (до 512 символов или 1 килобайта).

Ниже представлен пример документа коллекции *person*:

```
{
  _id: ObjectId("6359c5e317df7f8644c2cea2"),
  name: "Екатерина",
  surname: "Аксёнова",
  patronymic: "Александровна",
  sex: "FEMALE",
  birthday: Date("2001-11-14"),
  first_work_date: Date("2022-04-14"),
  position: "JUNIOR_BACKEND_DEVELOPER",
  status: "WORKING",
  contacts: {
    phone_number: "79112886438",
    email: "kate.kate6575@gmail.com"
  },
  job_time: {
    start: "11:00",
    end: "20:00"
  },
  officeId: ObjectId("6359c11c2416d454722f826f"),
  confidential: {
    passport_data: {
      number: "BM5653789",
      issued_place: "Октябрьский РОВД",
      issued_date: Date("2015-10-14")
    },
    nationality: "BELARUS",
    address: "Улица Торжковская, дом 15",
    salary: Long("1000000"),
    project_ids: [ObjectId("6359c20fea6757775ceca286")]
  },
  comment: "Устала"
}
```

Коллекция *watcher*, содержащая информацию об активности (один документ занимает 1264 байт):

- *_id: ObjectId* – идентификатор записи (12 байт);
- *user_id: ObjectId* – идентификатор пользователя (12 байт);
- *date: Date* – дата активности (8 байт);
- *project_id: ObjectId* – название активности (12 байт);
- *minutes_amount: 64-Bit Integer* – затраченное время (8 байт);
- *comment: String* – описание (до 512 символов или 1 килобайта).

Ниже представлен пример документа коллекции *watcher*:

```
{
  _id: ObjectId("6359d0baf1339ed8b904fb00"),
  user_id: ObjectId("6359c5e317df7f8644c2cea2"),
  date: Date("2022-10-22"),
  project_id: ObjectId("6359c20fea6757775ceca286"),
  minutes_amount: Long("360"),
  comment: "Сделала авторизацию"
}
```

Коллекция *document*, содержащая информацию о заказанном документе (один документ занимает 59 байт):

- *_id: ObjectId* – идентификатор заявки (12 байт);
- *type: Enum* – название документа (16 байт);
- *user_id: ObjectId* – идентификатор пользователя (12 байт);
- *order_date: Date* – дата заказа (8 байт);
- *status: Enum* – статус заказа (11 байт).

Ниже представлен пример документа коллекции *document*:

```
{
  _id: ObjectId("6359d3a576a26bfdc3a506e8"),
  user_id: ObjectId("6359c5e317df7f8644c2cea2"),
  type: "INCOME_STATEMENT",
  order_date: Date("2022-10-20"),
  status: "IN_PROGRESS"
}
```

Коллекция *office*, содержащая информацию об офисе (один документ занимает 512 байт):

- *_id: ObjectId* – идентификатор офиса (12 байт);
- *name: String* – название офиса (до 50 символов или 100 байт);
- *address: String* – адрес офиса (до 200 символов или 400 байт).

Ниже представлен пример документа коллекции *office*:

```
{
  _id: ObjectId("6359c11c2416d454722f826f"),
  name: "Офис Витебск",
  address: "Улица Чапаева, дом 34"
}
```

Коллекция *project*, содержащая информацию о проекте (один документ занимает 112 байт):

- *_id: ObjectId* – идентификатор проекта (12 байт);
- *name: String* – название проекта (до 50 символов или 100 байт).

Ниже представлен пример документа коллекции *project*:

```
{
  _id: ObjectId("6359c20fea6757775ceca286"),
  name: "Buhinder"
}
```

Пусть количество записей в коллекции *A* равняется x_A . Весь объём занимаемой памяти равен $V_{actual} = 59 x_{auth} + 2352 x_{person} + 1264 x_{watcher} + 59 x_{document} + 512 x_{office} + 112 x_{project}$.

Количество записей в коллекциях *auth* и *person* одинаково: $x_{person} = x_{auth}$. Коллекции *office* и *project* можно считать словарями, а следовательно, их размеры являются константами. Будем считать, что в среднем на каждого пользователя в день приходится по 2 записи в *Watcher*, тогда объём занимаемой памяти коллекции *watcher* равен $x_{watcher} = 2Tx_{person}$, где T – количество дней. Рассмотрим промежуток времени, равный $T = 65$ дней (примерно, 3 месяца без учёта выходных), тогда $x_{watcher} = 130 x_{person}$. Будем также считать, что в среднем каждый пользователь заказывает 2 справки в течение 3 месяцев, тогда объём занимаемой памяти коллекции *document* равен $x_{document} = 2 x_{person}$. Учитывая всё выше перечисленное, получаем следующее значение объёма памяти: $V_{actual} = 166\,849 x_{person}$.

Избыточными полями являются те поля коллекций, которые могут повторяться в разных документах. В нашей базе данных к таким полям в худшем случае относятся: *person.id*, *person.sex*, *person.position*, *person.status*, *person.job_time.start*, *person.job_time.end*, *person.office_id*, *person.nationality*, *watcher.user_id*, *watcher.date*, *watcher.project_id*, *document.type*, *document.user_id*, *document.status*. Итого, суммарный объём избыточных данных равен $4414 x_{person}$, следовательно, «чистый» объём данных равен $V_{clear} = 162\,435 x_{person}$.

Таким образом, избыточность модели равна

$$redundancy = \frac{V_{actual}}{V_{clear}} \approx 1,027.$$

Анализируя модель данных и полученные результаты, приходим к выводу, что модель растёт с линейной скоростью.

Реализованы запросы к нереляционной модели, с помощью которых реализуются сценарии использования (приложение Д).

3.2. Реляционный аналог модели данных

Разработана схема реляционной базы данных (рисунок 2).

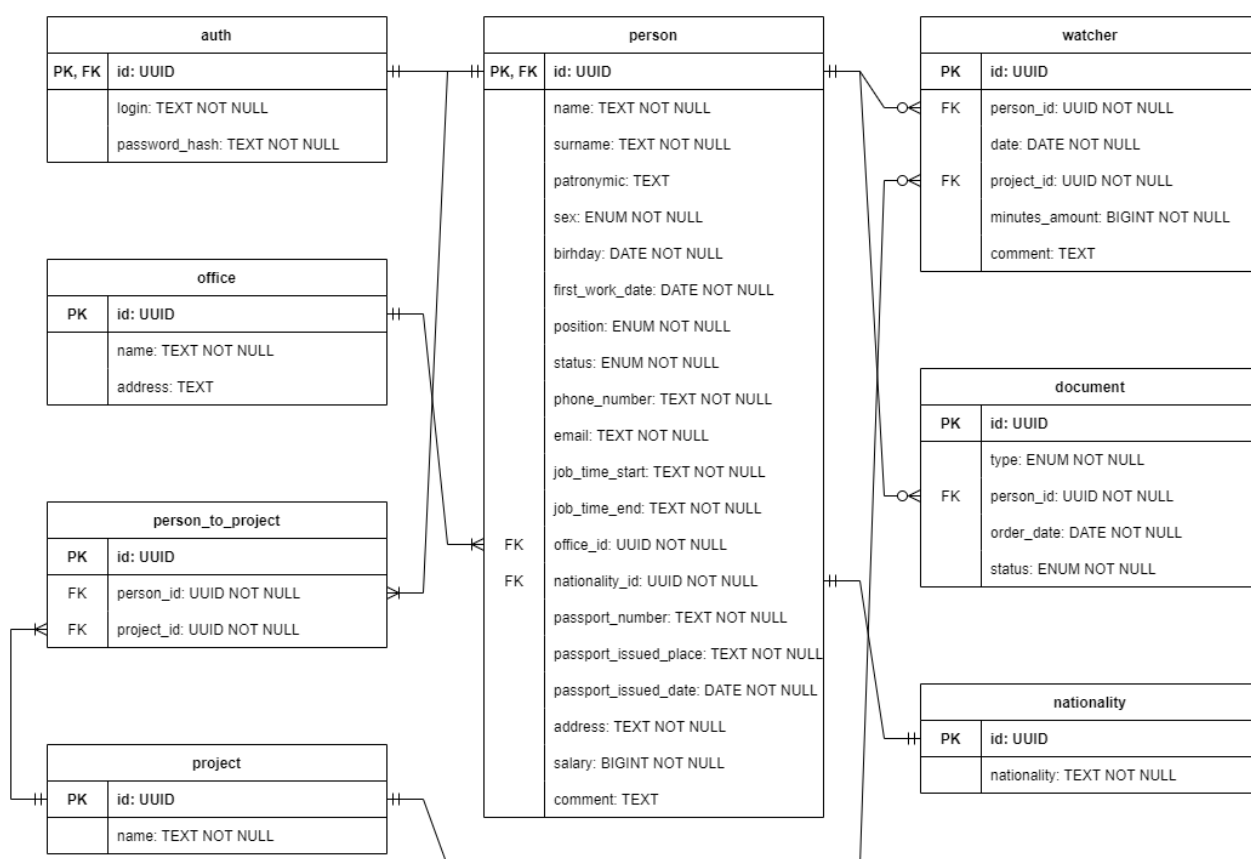


Рисунок 2 – Графическое представление реляционной базы данных

Нижу приведены описания таблиц. Будем считать, поскольку строки в SQL представлены в кодировке Unicode, то все символы кодируются 2 байтами.

Таблица *auth*, содержащая информацию для аутентификации (одна запись занимает 110 байт):

- *id: UUID* – идентификатор пользователя (16 байт);
- *login: TEXT* – логин пользователя (до 15 символов или 30 байт);
- *password_hash: TEXT* – хэш-код пароля (32 символов или 64 байт).

Ниже представлен пример записи таблицы *auth*:

```
{
  id: "78a1f37d-9043-4783-a5f5-3a353d9bfff49",
  login: "your_aksioma",
  password_hash: "8BeNwoHhivE2zbcrrhHnX9vqcAZWP1MnV"
}
```

Таблица *person*, содержащая информацию о работнике (одна запись занимает 2328 байт):

- *id: UUID* – идентификатор пользователя (16 байт);
- *name: TEXT* – имя (до 40 символов или 80 байт);
- *surname: TEXT* – фамилия (до 40 символов или 80 байт);
- *patronymic: TEXT* – отчество (до 40 символов или 80 байт);
- *sex: ENUM* – пол пользователя (12 байт);
- *birthday: DATE* – дата рождения (8 байт);
- *first_work_date: DATE* – дата первого рабочего дня (8 байт);
- *position: ENUM* – должность (50 байт);
- *status: ENUM* – статус (22 байт);
- *phone_number: TEXT* – номер телефона (до 20 символов или 40 байт);
- *email: TEXT* – электронная почта (до 100 символов или 200 байт);
- *job_time_start: TEXT* – начало рабочего дня (5 символов или 10 байт);
- *job_time_end: TEXT* – конец рабочего дня (5 символов или 10 байт);
- *office_id: UUID* – идентификатор офиса (16 байт);
- *nationality_id: UUID* – гражданство (до 50 символов или 100 байт);
- *passport_number: TEXT* – серия и номер паспорта (до 20 символов или 40 байт);
- *passport_issued_place: TEXT* – место выдачи (до 100 символов или 200 байт);
- *passport_issued_date: DATE* – идентификатор гражданства (16 байт);

- *address: TEXT* – адрес пользователя (до 200 символов или 400 байт);
- *salary: BIGINT* – заработная плата (8 байт);
- *comment: TEXT* – комментарий (до 512 символов или 1 килобайта).

Ниже представлен пример записи таблицы *person*:

```
{
  id: "78a1f37d-9043-4783-a5f5-3a353d9bff49",
  name: "Екатерина",
  surname: "Аксёнова",
  patronymic: "Александровна",
  sex: "FEMALE",
  birthday: "2001-11-14",
  first_work_date: "2022-04-14",
  position: "JUNIOR_BACKEND_DEVELOPER",
  status: "WORKING",
  phone_number: "79112886438",
  email: "kate.kate6575@gmail.com",
  job_time_start: "11:00",
  job_time_end: "20:00",
  office_id: "c74c02eb-2411-4dfd-bf29-d2d9a65219ca",
  nationality_id: "ccfe6bb0-b1e3-4210-9ef9-fc57746599df",
  passport_number: "BM5653789",
  passport_issued_place: "Октябрьский РОВД",
  passport_date: "2015-10-14",
  address: "Улица Торжковская, дом 15",
  salary: 1000000,
  comment: "Устала"
}
```

Таблица *office*, содержащая информацию об офисе (одна запись занимает 516 байт):

- *id: UUID* – идентификатор офиса (16 байт);
- *name: TEXT* – название офиса (до 50 символов или 100 байт);
- *address: TEXT* – адрес офиса (до 200 символов или 400 байт).

Ниже представлен пример записи таблицы *office*:

```
{
  id: "c74c02eb-2411-4dfd-bf29-d2d9a65219ca",
  name: "Офис Витебск",
  address: "Улица Чапаева, дом 34"
}
```

Таблица *person_to_project*, сопоставляющая пользователя и его проекты (одна запись занимает 48 байт):

- *id: UUID* – идентификатор (16 байт);
- *person_id: UUID* – идентификатор пользователя (16 байт);
- *project_id: UUID* – идентификатор проекта (16 байт).

Ниже представлен пример записи таблицы *person_to_project*:

```
{
  id: "094032ae-587f-48a7-962d-6f817d6b8f49",
  person_id: "78a1f37d-9043-4783-a5f5-3a353d9bfff49",
  project_id: "a0092694-19de-4dc8-8d54-ac75594920b2"
}
```

Таблица *project*, содержащая информацию о проекте (одна запись занимает 116 байт):

- *id: UUID* – идентификатор проекта (16 байт);
- *name: TEXT* – название проекта (до 50 символов или 100 байт).

Ниже представлен пример записи таблицы *project*:

```
{
  id: "a0092694-19de-4dc8-8d54-ac75594920b2",
  name: "Buhinder"
}
```

Таблица *nationality*, содержащая информацию о гражданствах (одна запись занимает 116 байт):

- *id: UUID* – идентификатор гражданства (16 байт);
- *nationality: TEXT* – название гражданства (до 50 символов или 100 байт).

Ниже представлен пример записи таблицы *nationality*:

```
{
  id: "ccfe6bb0-b1e3-4210-9ef9-fc57746599df",
  nationality: "BELARUS"
}
```

Таблица *watcher*, содержащая информацию об активности (одна запись занимает 1088 байт):

- *id: UUID* – идентификатор записи (16 байт);
- *person_id: UUID* – идентификатор пользователя (16 байт);
- *date: DATE* – дата активности (8 байт);
- *project_id: UUID* – идентификатор активности (16 байт);
- *minutes_amount: BIGINT* – затраченное время (8 байт);
- *comment: TEXT* – описание (до 512 символов или 1 килобайта).

Ниже представлен пример записи таблицы *watcher*:

```
{
  id: "43bb4c5c-fe1d-483d-90f4-23b4adc3505f",
  person_id: "78a1f37d-9043-4783-a5f5-3a353d9bfff49",
  date: "2022-10-22",
  project_id: "a0092694-19de-4dc8-8d54-ac75594920b2",
  minutes_amount: 360,
  comment: "Сделала авторизацию"
}
```

Таблица *document*, содержащая информацию о заказанном документе (одна запись занимает 94 байт):

- *id: UUID* – идентификатор заявки (16 байт);
- *type: ENUM* – название документа (32 байт);
- *person_id: UUID* – идентификатор пользователя (16 байт);
- *order_date: DATE* – дата заказа (8 байт);
- *status: ENUM* – статус заказа (22 байт).

Ниже представлен пример записи таблицы *document*:

```
{
  id: "cdbeeadc-ad3e-4b90-92e6-2fd3f44d8c13",
  person_id: "78a1f37d-9043-4783-a5f5-3a353d9bfff49",
  type: "INCOME_STATEMENT",
  order_date: "2022-10-20",
  status: "IN_PROGRESS"
}
```

Пусть количество записей в коллекции *A* равняется x_A . Весь объём занимаемой памяти равен $V_{actual} = 110 x_{auth} + 2328 x_{person} + 516 x_{office} + 48 x_{person_to_project} + 116 x_{project} + 116 x_{nationality} + 1088 x_{watcher} + 94 x_{document}$.

Аналогично расчётам оценки удельного объёма информации в нереляционной базе данных заключаем, что $x_{auth} = x_{person}$, $x_{office} = \text{const}$, $x_{project} = \text{const}$, $x_{watcher} = 130 x_{person}$, $x_{document} = 2 x_{person}$. Аналогично коллекциям *office* и *project* коллекцию *nationality* можно считать словарём: $x_{nationality} = \text{const}$. Поскольку количество проектов у человека не превышает 10, то $x_{person_to_project} = 10 x_{person}$. Учитывая всё выше перечисленное, получаем следующее значение объёма памяти: $V_{actual} = 144\,546 x_{person}$.

Избыточными полями являются те поля таблиц, которые могут повторяться в разных записях. В нашей базе данных к таким полям в худшем случае относятся: *person.id*, *person.sex*, *person.position*, *person.status*, *person.job_time_start*, *person.job_time_end*, *person.office_id*, *person.nationality_id*, *watcher.person_id*, *watcher.date*, *watcher.project_id*, *document.type*, *document.person_id*, *document.status*. Итого, суммарный объём избыточных данных равен $5524 x_{person}$, следовательно, «чистый» объём данных равен $V_{clear} = 139\,022 x_{person}$.

Таким образом, избыточность модели равна

$$redundancy = \frac{V_{actual}}{V_{clear}} \approx 1,040.$$

Анализируя модель данных и полученные результаты, приходим к выводу, что модель растёт с линейной скоростью.

Реализованы запросы к реляционной модели, с помощью которых реализуются сценарии использования (приложение Е).

3.3. Сравнение моделей

Сравнение реляционной и нереляционной моделей данных представлено в таблице 1.

Таблица 1 – Сравнение моделей

	SQL	NoSQL
Удельный объём информации	144 546 x	166 864 x
Избыточность	1,040	1,027
Среднее количество запросов	2,15	1,20
Количество коллекций	8	6

Исходя из вышеприведённых результатов можно заключить, что реляционная база данных занимает меньше памяти, чем нереляционная, однако избыточность данных у неё больше. Также преимуществом нереляционной БД является меньшее количество обращений к коллекциям.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. Описание приложения

Backend представляет собой 4 микросервиса для авторизации и каждого модуля: Watcher, Person и Document.

Frontend представляет собой web-приложение. Разработанный интерфейс представлен в приложении Г.

Пользователь, пройдя авторизацию, получает определённые права в зависимости от должности. На основании прав имеет те или иные возможности, например, просмотр конфиденциальной информации.

Приложение содержит три модуля: Watcher для списывания отработанного времени с указанием занятий, Person для просмотра определённой информации о сотрудниках, Document для формирования заказов необходимых документов.

Инструкция по сборке и развёртыванию приложения представлена в приложении Б.

Инструкция для пользователей расположена в приложении В.

4.2. Используемые технологии

Для frontend-части используется следующий стек технологий:

- язык программирования TypeScript;
- библиотека React;
- менеджер состояний Redux;
- библиотека компонентов Material UI

Для backend-части используется следующий стек технологий:

- язык программирования Kotlin;
- фреймворк Spring Boot;
- интеграция с БД Spring Data MongoDB;
- база данных MongoDB.

Для сборки проекта используется Docker Compose.

ЗАКЛЮЧЕНИЕ

В ходе данной работы был разработан макет информационной системы для IT компании, на основе которого реализован прототип приложения, содержащий основной функционал. Приложение позволяет списывать время по выполненным задачам, просматривать общую информацию о сотрудниках, оставлять заказы на формирование документов.

В текущей реализации имеется ряд недостатков:

- Передача импортируемых и экспортируемых данных реализовано одним файлом. Необходимо реализовать частичную передачу данных по REST API для снижения нагрузки на сеть, что позволит отправлять больший объём данных.
- Работа со временем ведётся в локальной временной зоне. Необходимо перейти на временной формат с указанием зоны, что поможет избежать ошибок при граничных случаях.
- Отсутствие отдельных баз данных для каждого микросервиса. Необходимо разделить базы данных с введением уникальных авторизационных данных, что позволит увеличить безопасность.
- Отсутствует межсервисное взаимодействие через REST API (Spring Cloud OpenFeign). Необходимо реализовать feign-клиентов для межсервисного взаимодействия.
- Незавершённая клиентская часть. Необходимо исправить визуальную часть, добавить недостающий функционал.
- Недостаточно организованная структура React-приложения. Необходимо реорганизовать компоненты, что позволит упростить дальнейшие разработку и поддержку приложения.

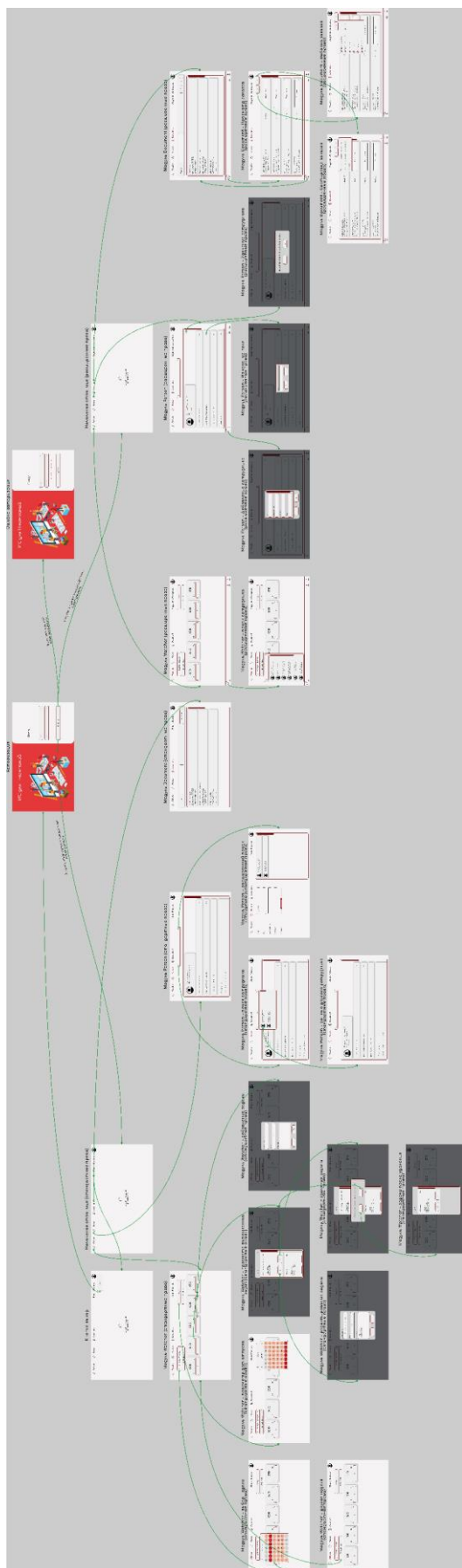
В дальнейшем данное приложение можно расширить, добавив новые модули.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Spring Documentation [Электронный ресурс] URL:
<https://spring.io/guides> (дата обращения: 10.11.2022)
2. Kotlin Documentation [Электронный ресурс] URL:
<https://kotlinlang.org/docs/home.html> (дата обращения: 10.11.2022)
3. MongoDB Documentation [Электронный ресурс] URL:
<https://www.mongodb.com/docs> (дата обращения: 13.10.2022)
4. WebFlux Documentation [Электронный ресурс] URL:
<https://docs.spring.io/spring-framework/docs/current/reference/html/web-reactive.html> (дата обращения: 10.11.2022)
5. Spring Data MongoDB Documentation [Электронный ресурс] URL:
<https://spring.io/projects/spring-data-mongodb> (дата обращения: 10.11.2022)
6. Material UI Documentation [Электронный ресурс]. URL:
<https://mui.com/material-ui/getting-started/overview> (дата обращения 15.11.2022)
7. React Documentation [Электронный ресурс]. URL:
<https://reactjs.org/docs/getting-started.html> (дата обращения 25.10.2022)
8. TypeScript Documentation [Электронный ресурс]. URL:
<https://www.typescriptlang.org/docs> (дата обращения: 15.11.2022)
9. Docker Documentation [Электронный ресурс]. URL:
<https://docs.docker.com> (дата обращения: 28.11.2022)

ПРИЛОЖЕНИЕ А

МАКЕТ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА



ПРИЛОЖЕНИЕ Б

ДОКУМЕНТАЦИЯ ПО СБОРКЕ И РАЗВЁРТЫВАНИЮ ПРИЛОЖЕНИЯ

Инструкция по сборке и развёртыванию приложений:

1. Необходимо клонировать репозиторий проекта по данной ссылке:
<https://github.com/moevm/nosql2h22-it-company>.
2. Для запуска всех сервисов необходимо в директории `/environment` запустить Docker Compose с помощью следующей команды:

```
docker-compose up
```
3. После завершения локального развёртывания приложения необходимо перейти по ссылке: <http://localhost:3000>.

Более подробную информацию можно прочитать в `README.md` в репозитории проекта.

ПРИЛОЖЕНИЕ В

ИНСТРУКЦИЯ ДЛЯ ПОЛЬЗОВАТЕЛЯ

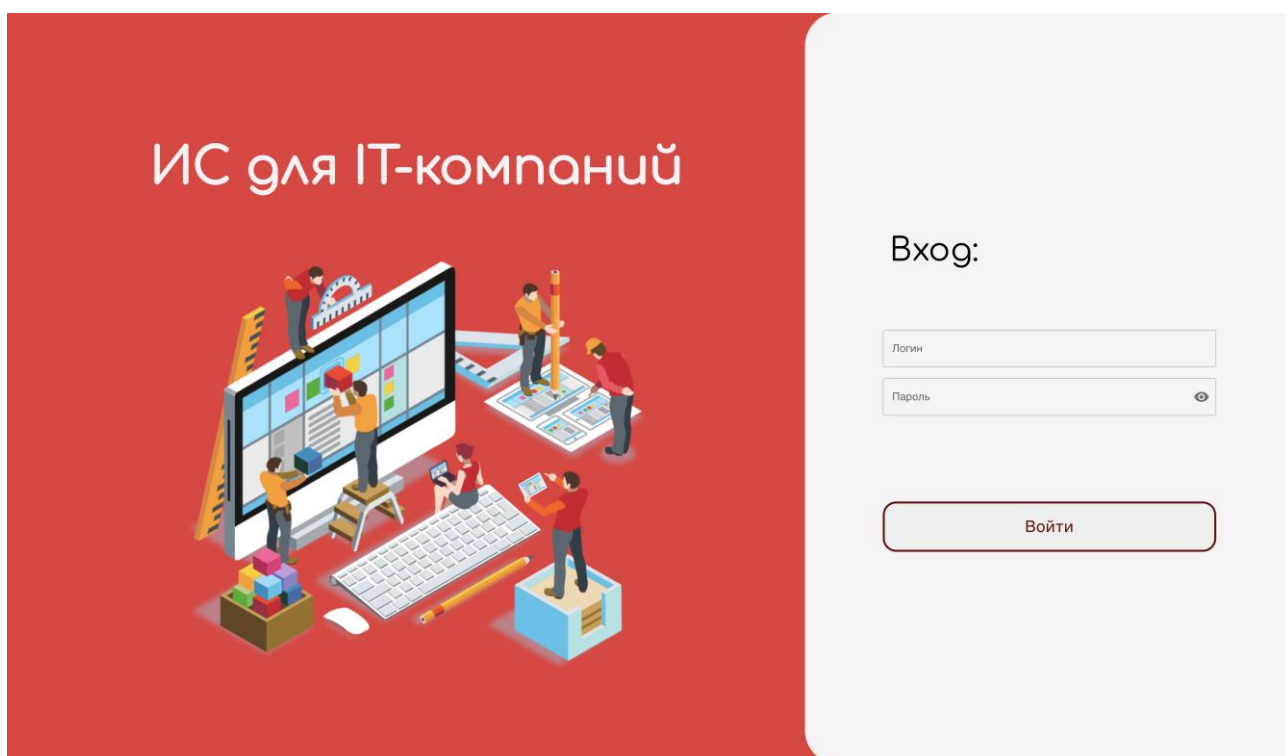
Инструкция для новых пользователей:

1. Для авторизации в приложении необходимо запросить регистрацию у отдела кадров (HR) компании.
2. С помощью полученных логина и пароля авторизируйтесь в приложении.
3. Для перехода между модулями имеется панель в верхней части приложения.
4. Для использования функций, доступных пользователям с расширенными правами, имеется панель в нижней части приложения.

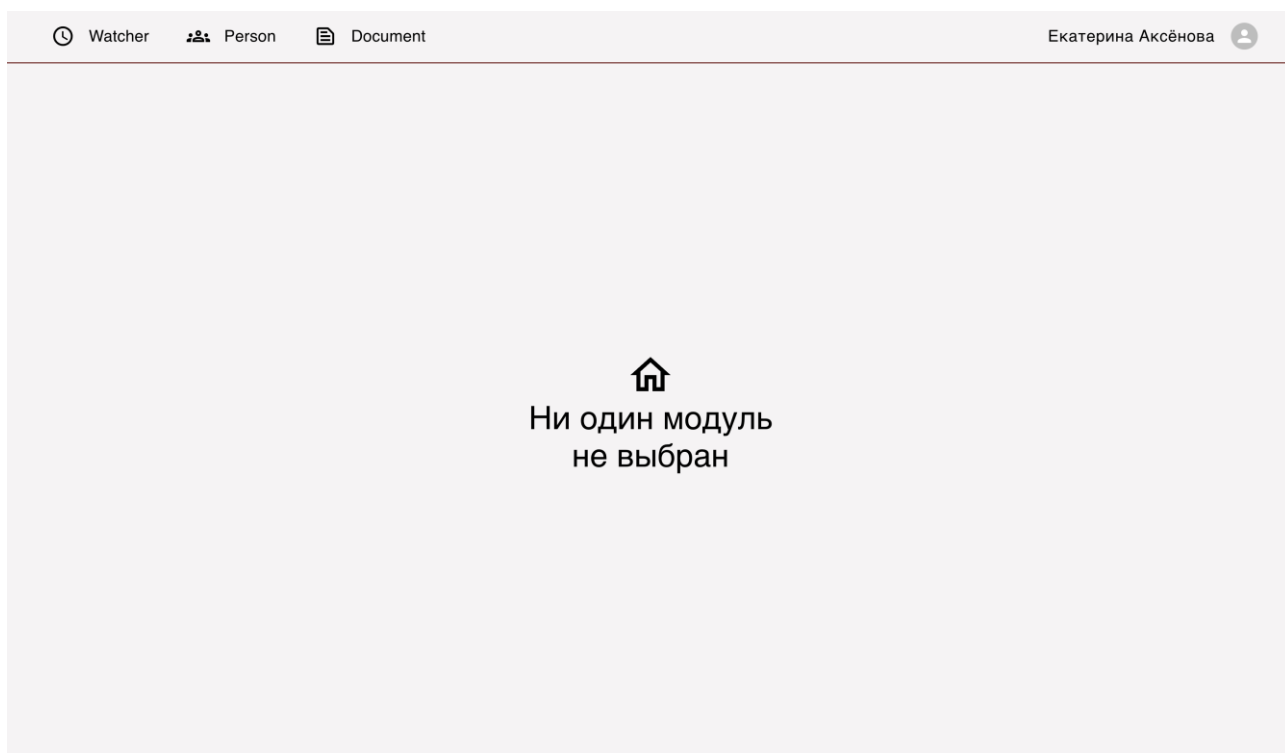
ПРИЛОЖЕНИЕ Г

СНИМКИ ЭКРАНА ПРИЛОЖЕНИЯ

Стартовая страница:



Домашняя страница:



Модуль Watcher:

🕒 Watcher

👤 Person

📄 Document

Екатерина Аксёнова

Текущая неделя
2022-12-12 - 2022-12-18

Дата: 2022-12-12, пн
Часы: 8.00
+
ПОДРОБНЕЕ

Дата: 2022-12-13, вт
Часы: 8.00
+
ПОДРОБНЕЕ

Дата: 2022-12-14, ср
Часы: 8.00
+
ПОДРОБНЕЕ

Дата: 2022-12-15, чт
Часы: 8.00
+
ПОДРОБНЕЕ

Дата: 2022-12-16, пт
Часы: 8.00
+
ПОДРОБНЕЕ

Дата: 2022-12-17, сб
Часы: 8.00
+
ПОДРОБНЕЕ

Дата: 2022-12-18, вс
Часы: 0.00
+
ПОДРОБНЕЕ

↓

↑

Модуль Person:

🕒 Watcher

👤 Person

📄 Document

Екатерина Аксёнова

🔍 map

Мария Иванова

👤 Екатерина Аксёнова, 21
HR
Работает

Телефон: 79112886438

Email: kate.kate6575@gmail.com

Время работы: 11:00 – 20:00

Паспорт: BM5653789, выдан Октябрьский РОВД 2015-10-13

Адрес: Улица Торжковская, дом 15

Зароботная плата: R1000000

Комментарий:
Комментарий для Екатерины

+

↓

↑

Модуль Document:

🕒


Watcher

👤

Person

📄

Document

Екатерина Аксёнова 

Справка:

Отчёт о доходах ▾

[ЗАКАЗАТЬ](#)

Мои справки:

Отчёт о доходах
Дата заказа: 2022-10-14
Дата выдачи: не выдано
Статус: Заказано

Отчёт о доходах
Дата заказа: 2022-09-07
Дата выдачи: не выдано
Статус: В процессе

Отчёт о доходах
Дата заказа: 2022-09-16
Дата выдачи: 2022-12-14
Статус: Готово

Отчёт о доходах
Дата заказа: 2022-11-17
Дата выдачи: не выдано
Статус: Отменено

Отчёт об отработанных днях
Дата заказа: 2022-11-12
Дата выдачи: не выдано
Статус: Заказано





ПРИЛОЖЕНИЕ Д

ЗАПРОСЫ К НЕРЕЛЯЦИОННОЙ БАЗЕ ДАННЫХ

Проверка наличия пользователя в системе:

```
let kateLogin = "your_aksioma";
let katePasswordHash = "8BeNwoHhivE2zbcrhHnX9vqcAZWP1MnV";

db.getCollection("auth")
  .findOne({login: kateLogin, password_hash: katePasswordHash}, {_id: 1});
```

Показ всех дней в неделе с 03.10.2022 по 07.10.2022 в модуле Watcher:

```
let userId = new ObjectId("6359c5e317df7f8644c2cea2");
let firstWeekDate = new Date("2022-10-03");
let lastWeekDate = new Date("2022-10-07");

db.watcher.aggregate(
  {$match: {date: {$gte: firstWeekDate, $lte: lastWeekDate}, user_id: userId}},
  {$group: {_id: "$date", minutes_amount: {$sum: "$minutes_amount"}}}
);
```

Добавление задачи:

```
let userId = new ObjectId("6359c5e317df7f8644c2cea2");
let specificDateForNewRow = new Date("2022-10-25");
let newProjectId = ObjectId("6359c217e6faf17102236f02");
let newComment = "random new comment";
let newMinutes = Long("1020");

db.watcher.insertOne({
  user_id: userId,
  date: specificDateForNewRow,
  project_id: newProjectId,
  comment: newComment,
  minutes_amount: newMinutes
});
```

Просмотр задач за 25.10.2022:

```
let userId = new ObjectId("6359c5e317df7f8644c2cea2");
let specificDateForNewRow = new Date("2022-10-25");
let newProjectId = ObjectId("6359c217e6faf17102236f02");
let newComment = "random new comment";
let newMinutes = Long("1020");

db.watcher.insertOne({
  user_id: userId,
  date: specificDateForNewRow,
  project_id: newProjectId,
  comment: newComment,
  minutes_amount: newMinutes
});
```

Изменение задачи:

```
let watcherId = ObjectId("6359d0baf1339ed8b904fb00");
let newProjectId = ObjectId("6359c217e6faf17102236f02");
let newComment = "random new comment";
let newMinutes = Long("1020");

db.watcher.updateOne(
  { _id: watcherId },
  { $set: { project_id: newProjectId, comment: newComment, minutes_amount: newMinutes } }
);
```

Удаление задачи:

```
let watcherId = ObjectId("6359d0baf1339ed8b904fb00");

db.watcher.deleteOne({ _id: watcherId });
```

Просмотр дней отпуска:

```
let userId = new ObjectId("6359c5e317df7f8644c2cea2");
let firstMonthDate = new Date("2022-10-01");
let lastMonthDate = new Date("2022-10-31");

db.watcher.aggregate([
  { $match: { date: { $gte: firstMonthDate, $lte: lastMonthDate }, user_id: userId },
    { $lookup: { from: "project", localField: "project_id", foreignField: "_id", as:
      "project_info" } },
    { $match: { "project_info.name": "Отпуск" } },
    { $group: { _id: "$date" } },
  ]);
```

Просмотр больничных дней:

```
let userId = new ObjectId("6359c5e317df7f8644c2cea2");
let firstMonthDate = new Date("2022-10-01");
let lastMonthDate = new Date("2022-10-31");

db.watcher.aggregate([
  { $match: { date: { $gte: firstMonthDate, $lte: lastMonthDate }, user_id: userId },
    { $lookup: { from: "project", localField: "project_id", foreignField: "_id", as:
      "project_info" } },
    { $match: { "project_info.name": "Больничный" } },
    { $group: { _id: "$date" } },
  ]);
```

Показ всей информации о пользователе:

```
let userId = new ObjectId("6359c5e317df7f8644c2cea2");

db.person.findOne({ _id: userId });
```

Изменение своих контактов:

```
let userId = new ObjectId("6359c5e317df7f8644c2cea2");
let newPhone = "791112929292";
let newEmail = "muzan@demon.com";

db.person.updateOne(
  {_id: userId},
  {$set: {contacts: {phone_number: newPhone, email: newEmail}}}
);
```

Поиск сотрудника, в имени которого содержится подстрока «а»:

```
let matchExpression = "a";

db.person.find(
  {name: {$regex: matchExpression, $options: "i"}},
  {_id: 1, name: 1, surname: 1, position: 1}
);
```

Расширенный поиск сотрудника женского пола, у которого присутствует подстрока «а» в имени, фамилии, отчестве, а также в должности и электронной почте, у которого возраст от 18 до 30 лет, в номере телефона которого присутствует набор символов «7»:

```
let nameMatchExpression = "a";
let surnameMatchExpression = "a";
let patronymicMatchExpression = "a";
let sexMatch = "FEMALE";
let ageMinMatchExpression = 18;
let ageMaxMatchExpression = 30;
let positionMatchExpression = "a";
let emailMatchExpression = "a";
let phoneMatchExpression = "7";

db.person.aggregate([
  {$addFields: {age: {$dateDiff: {startDate: "$birthday", endDate: new Date(), unit: "year"}}}},
  {
    $match: {
      name: {$regex: nameMatchExpression, $options: "i"},
      surname: {$regex: surnameMatchExpression, $options: "i"},
      patronymic: {$regex: patronymicMatchExpression, $options: "i"},
      sex: {$regex: sexMatch, $options: "i"},
      age: {$gte: ageMinMatchExpression, $lte: ageMaxMatchExpression},
      position: {$regex: positionMatchExpression, $options: "i"},
      "contacts.email": {$regex: emailMatchExpression, $options: "i"},
      "contacts.phone_number": {$regex: phoneMatchExpression, $options: "i"},
    }
  },
  {$unset: "age"},
  {$project: {_id: 1, name: 1, surname: 1, position: 1}}
]);
```

Заказ справки:

```
let userId = new ObjectId("6359c5e317df7f8644c2cea2");
let nowDate = new Date();
let selectedStatus = "IN_PROGRESS";
let selectedType = "INCOME_STATEMENT";

db.document.insertOne({
  order_date: nowDate,
  status: selectedStatus,
  type: selectedType,
  user_id: userId
});
```

Просмотр заказанных справок:

```
let userId = new ObjectId("6359c5e317df7f8644c2cea2");

db.document.find({user_id: userId});
```

Добавление сотрудника:

```
let muzanAuth = {
  _id: ObjectId("635b14aecc60fb4f04fc25d9"),
  login: "muzan_kibutsuji",
  password_hash: "f415df421177820c3a69db701f424efb"
};

db.getCollection("auth").insertOne(muzanAuth);

let muzanPerson = {
  name: "Muzan",
  surname: "Kibutsuji",
  sex: "MALE",
  birthday: new Date("1095-01-01"),
  first_work_date: new Date("1096-01-01"),
  position: "ADMINISTRATION",
  status: "ON_HOLIDAY",
  contacts: {
    phone_number: "1038492382334",
    email: "muzan@demon.com",
  },
  job_time: {
    start: "09:00",
    end: "18:00",
  },
  officeId: ObjectId("6359c18e62901a6a7db55ef7"),
  confidential: {
    passport_data: {
      number: "BM3478689",
      issued_place: "Октябрьский РОВД",
      issued_date: new Date("2015-10-14")
    },
    nationality: "RUSSIA",
    address: "Улица Торжковская, дом 15",
    salary: Long("1000000"),
    project_ids: [ObjectId("6359c217e6faf17102236f02")],
  },
};
```



```
    comment: "Молодой специалист"
  });

db.person.insertOne(muzanPerson);
```

Изменение информации о сотруднике:

```
let muzanId = ObjectId("635b14aecc60fb4f04fc25d9");
let muzanUpdatedPerson = muzanPerson = {
  status: "WORKING",
  contacts: {
    phone_number: "100001001111",
  },
  job_time: {
    start: "10:00",
    end: "19:00",
  },
  confidential: {
    salary: Long("312000000"),
  },
  comment: "Обновленная информация"
};

db.person.updateOne(
  { _id: muzanId },
  { $set: muzanUpdatedPerson }
);
```

Удаление сотрудника:

```
let muzanId = ObjectId("635b14aecc60fb4f04fc25d9");

db.person.deleteOne({ _id: muzanId });
```

Просмотр всех заказанных справок:

```
let userId = new ObjectId("6359c5e317df7f8644c2cea2");

db.document.find({
  user_id: userId,
  status: { $in: ["DONE", "ORDERED"] }
});
```

Изменение статуса заказа на "Отменено":

```
let documentId = ObjectId("6359beb21b389817992710c0");
let newDocumentStatus = "DONE";

db.document.updateOne(
  { _id: documentId },
  { $set: { status: newDocumentStatus } }
);
```

Сценарий использования "Массовый импорт / экспорт данных":

```
db.getCollection("auth").find();

db.person.aggregate([
  {$lookup: {from: "office", localField: "office_id", foreignField: "_id", as:
"office_info"}},
  {$lookup: {from: "project", localField: "confidential.project_ids", foreignField:
"_id", as: "confidential.projects_info"}},
  {$unset: ["office_id", "confidential.project_ids"]}
]);

db.document.find();

db.watcher.aggregate(
  {$lookup: {from: "project", localField: "project_id", foreignField: "_id", as:
"project_info"}},
  {$unset: ["project_id"]}
);
```

ПРИЛОЖЕНИЕ Е

ЗАПРОСЫ К РЕЛЯЦИОННОЙ БАЗЕ ДАННЫХ

Проверка наличия пользователя в системе:

```
WITH my_constants (usr_login, usr_password_hash)
  as (values ('your_aksioma', '8BeNwoHhivE2zbcRhHnX9vqcAZWP1MnV'))

SELECT id
FROM auth
WHERE login = (SELECT usr_login FROM my_constants)
AND password_hash = (SELECT usr_password_hash FROM my_constants);
```

Показ всех дней в неделе с 03.10.2022 по 07.10.2022 в модуле Watcher:

```
WITH my_constants (usr_id, usr_date_start, usr_date_end)
  as (values ('78a1f37d-9043-4783-a5f5-3a353d9bffa9', '2022-10-24', '2022-10-28'))

SELECT date, SUM(minutes_amount)
FROM watcher
WHERE person_id = (SELECT usr_id::uuid FROM my_constants)
AND date BETWEEN (SELECT usr_date_start::date FROM my_constants) AND (SELECT
usr_date_end::date FROM my_constants)
GROUP BY date;
```

Добавление задачи:

```
WITH my_constants (usr_id, usr_date, usr_project_name, usr_time, usr_comment)
  as (values ('78a1f37d-9043-4783-a5f5-3a353d9bffa9', '2022-10-25', 'Buhinder', 360,
'Sделала авторизацию X2'))

INSERT
INTO watcher (person_id, date, project_id, minutes_amount, comment)
SELECT (SELECT usr_id::uuid FROM my_constants),
      (SELECT usr_date::date FROM my_constants),
      project.id,
      (SELECT usr_time FROM my_constants),
      (SELECT usr_comment FROM my_constants)
FROM project
WHERE name = (SELECT usr_project_name FROM my_constants);
```

Просмотр задач за 25.10.2022:

```
WITH my_constants (usr_id, usr_date)
  as (values ('78a1f37d-9043-4783-a5f5-3a353d9bffa9', '2022-10-25'))

SELECT p.name, watcher.minutes_amount, watcher.comment
FROM watcher
LEFT JOIN project p on watcher.project_id = p.id
WHERE person_id = (SELECT usr_id::uuid FROM my_constants)
AND date = (SELECT usr_date::date FROM my_constants);
```

Изменение задачи:

```
WITH my_constants (usr_watcher_id, usr_time, usr_comment)
  as (values ('6aa884fe-d01b-4e30-b64a-c71d16ddb308', 480, 'Создал бд для справок'))

UPDATE watcher
SET minutes_amount = (SELECT usr_time FROM my_constants),
    comment         = (SELECT usr_comment FROM my_constants)
WHERE id = (SELECT usr_watcher_id::uuid FROM my_constants);
```

Удаление задачи:

```
WITH my_constants (usr_watcher_id)
  as (values ('f6221d65-af6e-4c4c-b071-bb85936ae25f'))

DELETE
FROM watcher
WHERE id = (SELECT usr_watcher_id::uuid FROM my_constants);
```

Просмотр дней отпуска:

```
WITH my_constants (usr_activity)
  as (values ('Отпуск'))

SELECT watcher.date
FROM watcher
  JOIN project p on p.id = watcher.project_id
WHERE p.name = (SELECT usr_activity FROM my_constants);
```

Просмотр больничных дней:

```
WITH my_constants (usr_activity)
  as (values ('Больничный'))

SELECT watcher.date
FROM watcher
  JOIN project p on p.id = watcher.project_id
WHERE p.name = (SELECT usr_activity FROM my_constants);
```

Показ всей информации о пользователе:

```
WITH my_constants (usr_id)
  as (values ('78a1f37d-9043-4783-a5f5-3a353d9bff49'))

SELECT *
FROM person
WHERE id = (SELECT usr_id::uuid FROM my_constants);
```

Изменение своих контактов:

```
WITH my_constants (usr_id, usr_number, usr_email)
  as (values ('78a1f37d-9043-4783-a5f5-3a353d9bff49', '79179842361',
    'kate6575@gmail.com'))

UPDATE person
```

```
SET email      = (SELECT usr_email FROM my_constants),
   phone_number = (SELECT phone_number FROM my_constants)
WHERE id = (SELECT usr_id::uuid FROM my_constants);
```

Поиск сотрудника, в имени которого содержится подстрока «а»:

```
WITH my_constants (name_pattern)
  as (values ('%a%'))

SELECT id, name, surname, position
FROM person
WHERE name ILIKE (SELECT name_pattern FROM my_constants);
```

Расширенный поиск сотрудника женского пола, у которого присутствует подстрока «а» в имени, фамилии, отчестве, а также в должности и электронной почте, у которого возраст от 18 до 30 лет, в номере телефона которого присутствует набор символов «7»:

```
WITH my_constants (name_pattern, surname_pattern, patronymic_pattern, age_min, age_max,
sex, position_pattern, email_pattern, number_pattern)
  as (values ('%a%', '%a%', '%a%', 18, 30, 'FEMALE', '%a%', '%a%', '%7%'))

SELECT id, name, surname, position
FROM person
WHERE name ILIKE (SELECT name_pattern FROM my_constants)
  AND surname ILIKE (SELECT surname_pattern FROM my_constants)
  AND patronymic ILIKE (SELECT patronymic_pattern FROM my_constants)
  AND DATE_PART('year', AGE(now(), birthday)) BETWEEN (SELECT age_min FROM
my_constants) AND (SELECT age_max FROM my_constants)
  AND sex = (SELECT sex::sex FROM my_constants)
  AND position::text ILIKE (SELECT position_pattern FROM my_constants)
  AND email ILIKE (SELECT email_pattern FROM my_constants)
  AND phone_number ILIKE (SELECT number_pattern FROM my_constants);
```

Заказ справки:

```
WITH my_constants (usr_id, doc_type)
  as (values ('78a1f37d-9043-4783-a5f5-3a353d9bffa49', 'WORK_STATEMENT'))

INSERT
INTO document(type, person_id, order_date, status)
VALUES ((SELECT doc_type::document_type FROM my_constants), (SELECT usr_id::uuid FROM
my_constants), CURRENT_DATE, 'ORDERED');
```

Просмотр заказанных справок:

```
WITH my_constants (usr_id)
  as (values ('78a1f37d-9043-4783-a5f5-3a353d9bffa49'))

SELECT *
FROM document
WHERE person_id = (SELECT usr_id::uuid FROM my_constants);
```

Добавление сотрудника:

```
WITH my_constants_auth (id, login, password_hash)
  as (values ('c48fadb6-4bed-435b-9d1b-6947ccb77644'::uuid, 'lelen',
'm5C09li7Qk/4ifl7IwPXxY96j/YNQZq4'))

INSERT
INTO auth (id, login, password_hash)
SELECT *
FROM my_constants_auth;

WITH my_constants_person (id, usr_name, usr_surname, usr_patronymic, usr_sex,
  usr_birthday, usr_first_work_day, usr_position, usr_status,
  usr_phone_number, usr_email,
  usr_job_time_start, usr_job_time_end, usr_office_id,
  usr_nationality_id, usr_passport_number,
  usr_passport_issued_place, usr_passport_issued_date,
  usr_address, usr_salary, usr_comment)
  as (values ('c48fadb6-4bed-435b-9d1b-6947ccb77644'::uuid, 'Елена', 'Иванова',
'Алексеевна', 'FEMALE'::sex, '2001-10-14'::date, '2022-07-14'::date,
'JUNIOR_BACKEND_DEVELOPER'::usr_position, 'WORKING'::person_status,
'79117236438', 'lelens@gmail.com', '9:00', '18:00',
'bd711817-57a0-430d-8553-15dd4e3d7de0'::uuid,
'f7d00710-1f66-46d8-a3de-b0e4536774a1'::uuid,
'BM3478689', 'Октябрьский РОВД', '2015-10-14'::date,
'Улица Торжковская, дом 15', 1000000, 'Молодой специалист'))

INSERT
INTO person (id, name, surname, patronymic, sex, birthday, first_work_day,
  position, status, phone_number, email, job_time_start, job_time_end,
  office_id, nationality_id,
  passport_number, passport_issued_place, passport_issued_date,
  address, salary, comment)
SELECT *
FROM my_constants_person;
```

Изменение информации о сотруднике:

```
WITH my_constants (usr_id, usr_name, usr_surname, usr_patronymic, usr_sex,
  usr_birthday, usr_first_work_day, usr_position, usr_status,
  usr_phone_number, usr_email, usr_job_time_start, usr_job_time_end,
  usr_office_id, usr_nationality_id, usr_passport_number,
  usr_passport_issued_place, usr_passport_issued_date,
  usr_address, usr_salary, usr_comment)
  as (values ('c48fadb6-4bed-435b-9d1b-6947ccb77644'::uuid, 'Лена', 'Иванова',
'Алексеевна', 'FEMALE'::sex, '2001-10-14'::date, '2022-07-14'::date,
'JUNIOR_BACKEND_DEVELOPER'::usr_position, 'WORKING'::person_status,
'79117236438', 'lelens@gmail.com', '9:00', '18:00',
'bd711817-57a0-430d-8553-15dd4e3d7de0'::uuid,
'f7d00710-1f66-46d8-a3de-b0e4536774a1'::uuid,
'BM3478689', 'Октябрьский РОВД', '2015-10-14'::date,
'Улица Торжковская, дом 15', 1000000, 'Молодой специалист'))

UPDATE person
SET name = (SELECT usr_name FROM my_constants),
  surname = (SELECT usr_surname FROM my_constants),
  patronymic = (SELECT usr_patronymic FROM my_constants),
  sex = (SELECT usr_sex FROM my_constants),
  birthday = (SELECT usr_birthday FROM my_constants),
```

```

first_work_day      = (SELECT usr_first_work_day FROM my_constants),
position            = (SELECT usr_position FROM my_constants),
status              = (SELECT usr_status FROM my_constants),
phone_number        = (SELECT usr_phone_number FROM my_constants),
email               = (SELECT usr_email FROM my_constants),
job_time_start      = (SELECT usr_job_time_start FROM my_constants),
job_time_end        = (SELECT usr_job_time_end FROM my_constants),
office_id           = (SELECT usr_office_id FROM my_constants),
nationality_id      = (SELECT usr_nationality_id FROM my_constants),
passport_number     = (SELECT usr_passport_number FROM my_constants),
passport_issued_place = (SELECT usr_passport_issued_place FROM my_constants),
passport_issued_date = (SELECT usr_passport_issued_date FROM my_constants),
address             = (SELECT usr_address FROM my_constants),
salary              = (SELECT usr_salary FROM my_constants),
comment             = (SELECT usr_comment FROM my_constants)
WHERE id = (SELECT usr_id::uuid FROM my_constants);

```

Удаление сотрудника:

```

WITH my_constants (usr_id)
  as (values ('78a1f37d-9043-4783-a5f5-3a353d9bffa9'))

UPDATE person
SET status = 'NOT_WORKING'
WHERE id = (SELECT usr_id::uuid FROM my_constants);

```

Просмотр всех заказанных справок:

```

SELECT *
FROM person
  LEFT JOIN document d on person.id = d.person_id
  LEFT JOIN nationality n on person.nationality_id = n.id
  LEFT JOIN office o on person.office_id = o.id
WHERE d.status NOT IN ('DONE', 'CANCELED');

```

Изменение статуса заказа на "Отменено":

```

WITH my_constants (doc_id, new_status)
  as (values ('703d1140-96b8-4543-bf17-3d393c41f332', 'CANCELED'))

UPDATE document
SET status = (SELECT new_status::document_status FROM my_constants)
WHERE id = (SELECT doc_id::uuid FROM my_constants);

```

Сценарий использования "Массовый импорт / экспорт данных":

```

SELECT *
FROM person
  LEFT JOIN document d on person.id = d.person_id
  LEFT JOIN nationality n on person.nationality_id = n.id
  LEFT JOIN office o on person.office_id = o.id;

SELECT *
FROM person
  LEFT JOIN auth a on person.id = a.id
  LEFT JOIN nationality n on person.nationality_id = n.id
  LEFT JOIN office o on person.office_id = o.id;

```

```
SELECT *
FROM person
  LEFT JOIN nationality n on person.nationality_id = n.id
  LEFT JOIN office o on person.office_id = o.id
  LEFT JOIN person_to_project ptp on person.id = ptp.person_id
  LEFT JOIN project p on ptp.project_id = p.id;

SELECT *
FROM person
  LEFT JOIN watcher w on person.id = w.person_id
  LEFT JOIN nationality n on person.nationality_id = n.id
  LEFT JOIN office o on person.office_id = o.id;
```