**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет информационных технологий**

**ОТЧЕТ**

**О ВЫПОЛНЕНИИ ПРОЕКТНОЙ РАБОТЫ**

«Реализация приложения базы данных в архитектуре клиент-сервер»

студента 3 курса, группы 22205

***Тропина Никиты Васильевича***

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

С.А.Рылов

Новосибирск 2025

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc200116261)

[ЗАДАНИЕ 3](#_Toc200116262)

[ОПИСАНИЕ ПРОЕКТА 4](#_Toc200116263)

[Разработка структуры базы данных (серверная часть) 4](#_Toc200116264)

[СУЩНОСТИ 5](#_Toc200116265)

[СУЩНОСТИ И АТРИБУТЫ 6](#_Toc200116266)

[СУЩНОСТИ И СВЯЗИ 9](#_Toc200116267)

[ОГРАНИЧЕНИЯ ЦЕЛОСТНОСТИ 14](#_Toc200116268)

[СОЗДАНИЕ ТАБЛИЦ 14](#_Toc200116269)

[Написание SQL запросов к спроектированной базе данных. 21](#_Toc200116270)

[РЕАЛИЗАЦИЯ ТРИГГЕРОВ 28](#_Toc200116271)

[Разработка приложения клиента 33](#_Toc200116272)

[ЗАКЛЮЧЕНИЕ 36](#_Toc200116273)

# ЗАДАНИЕ

Для выбранного проекта студент разрабатывает структуру базы данных и реализует приложение в архитектуре клиент-сервер, выполняющее операции внесения данных в базу данных, редактирование данных и запросы, указанные в проекте. Клиентская часть реализуется на языке программирования высокого уровня.

В описании проекта дана обобщенная пользовательская спецификация приложения. Спецификация не предполагает оптимального определения структур данных, но задает полный перечень хранимой в базе данных информации и выполняемых программой функций.

Таблицы должны заполняться адекватными значениями, примерно по 7 объектов в каждой.

Разработка проекта предполагает выполнение следующих этапов:

1. **Разработка структуры базы данных (серверная часть)**
   1. Проектирование инфологической модели задачи. Определение сущностей, атрибутов сущностей, идентифицирующих атрибутов, связей между сущностями. При проектировании должны учитываться требования гибкости структур для выполнения перечисленных функций и не избыточного хранения данных.
   2. Проектирование схемы базы данных: описание схем таблиц, типов (доменов) атрибутов, определение ограничений целостности. Написание SQL скриптов по созданию таблиц БД.
   3. Создание и заполнение разработанной БД на стороне сервера.
2. **Написание SQL запросов к спроектированной базе данных согласно заданию.**
3. **Реализация триггеров и хранимых процедур (PL / SQL).**
4. **Разработка приложения клиента (формы ввода, редактирования и поиска данных по запросам)**

## ОПИСАНИЕ ПРОЕКТА

**12.Информационная система авиастроительного предприятия**

Структурно предприятие разбито на цеха, которые в свою очередь подразделяются на участки. Выпускаемые изделия предприятия - самолеты (гражданские, транспортные, военные), планеры, вертолеты, дельтапланы, ракеты (артиллерийские, авиационные, военно-морские), прочие изделия. Каждая категория изделий имеет специфические, присущие только ей атрибуты. Например, для самолетов это число двигателей, для ракеты - мощность заряда и т. д. По каждой категории изделий может собираться несколько видов изделий. Каждой категории инженерно-технического персонала (инженеры, технологи, техники) и рабочих (сборщики, токари, слесари, сварщики и пр.) также свойственны характерные только для этой группы атрибуты. Рабочие объединяется в бригады, которыми руководят бригадиры. Бригадиры выбираются из числа рабочих, мастера, начальники участков и цехов назначаются из числа инженерно-технического персонала.

Каждое изделие собирается в своем цехе (в цехе может собираться несколько видов изделий) и в процессе изготовления проходит определенный цикл работ, перемещаясь с одного участка на другой. Все работы по сборке конкретного изделия на определенном участке выполняет одна бригада рабочих, при этом на участке может работать несколько бригад. Возглавляет работу на участке начальник участка, в подчинении которого находится несколько мастеров. Различные изделия могут проходить одни и те же циклы работ на одних и тех же участках цеха.

Собранное изделие проходит серию испытаний в испытательных лабораториях (полигонах). Испытательные лаборатории могут обслуживать несколько цехов, в свою очередь цеха пользуются, возможно, несколькими испытательными лабораториями. Испытания проводятся испытателями на оборудовании испытательной лаборатории, при этом при испытании конкретного изделия в лаборатории могут быть задействованы различные виды оборудования.

Ведется учет движения кадров и учет выпускаемой продукции.

# Разработка структуры базы данных (серверная часть)

## СУЩНОСТИ

|  |  |  |
| --- | --- | --- |
| **№** | **Сущность** | **Описание** |
| 1 | Категории\_ИТ\_персонала | Набор категорий инженерно-технического персонала |
| 2 | Категории\_рабочих | Набор категория рабочих |
| 3 | ИТ\_персонал | Список людей инженерно\_технического персонала |
| 4 | Рабочие | Список рабочих |
| 5 | Люди | Общая информация обо всех сотрудниках |
| 6 | Бригады | Список бригад |
| 7 | Категории\_самолетов | Набор категорий самолетов |
| 8 | Самолеты | Модели выпускающихся самолетов |
| 9 | Категории\_ракет | Набор категорий ракет |
| 10 | Ракеты | Модели выпускаемых ракет |
| 11 | Вертолеты | Модели выпускаемых вертолетов |
| 12 | Планеры | Модели выпускаемых планеров |
| 13 | Дельтапланы | Модели выпускаемых дельтапланов |
| 14 | Виды изделий | Вид выпускаемых изделий |
| 15 | Модели\_изделий | Список всех выпускаемых моделей изделий |
| 16 | Выпускаемые\_изделия | Учет выпускаемых изделий |
| 17 | Цеха | Список цехов предприятия |
| 18 | Участки\_цехов | Список участков цехов |
| 19 | Мастера\_участка | Мастера, закрепленные за каждым участком |
| 20 | Виды\_работ\_на\_участке | Набор работ, выполняемых на участке |
| 21 | Выполненные работы | Учет выполненных работ |
| 22 | План\_сборки | Список необходимых работ для сборки модели изделий |
| 23 | Набор\_испытаний | Список необходимых испытаний для тестирования модели изделий |
| 24 | Испытания | Учеты выполненных испытаний |
| 25 | Испытатели | Список испытателей |
| 26 | Виды\_испытательного\_оборудования | Набор видов испытательного оборудования |
| 27 | Испытательное\_оборудование | Список имеющегося испытательного оборудования |
| 28 | Лаборатории | Лаборатории предприятия |
| 29 | Цех\_лаборатория | Связь цеха и лаборатории |

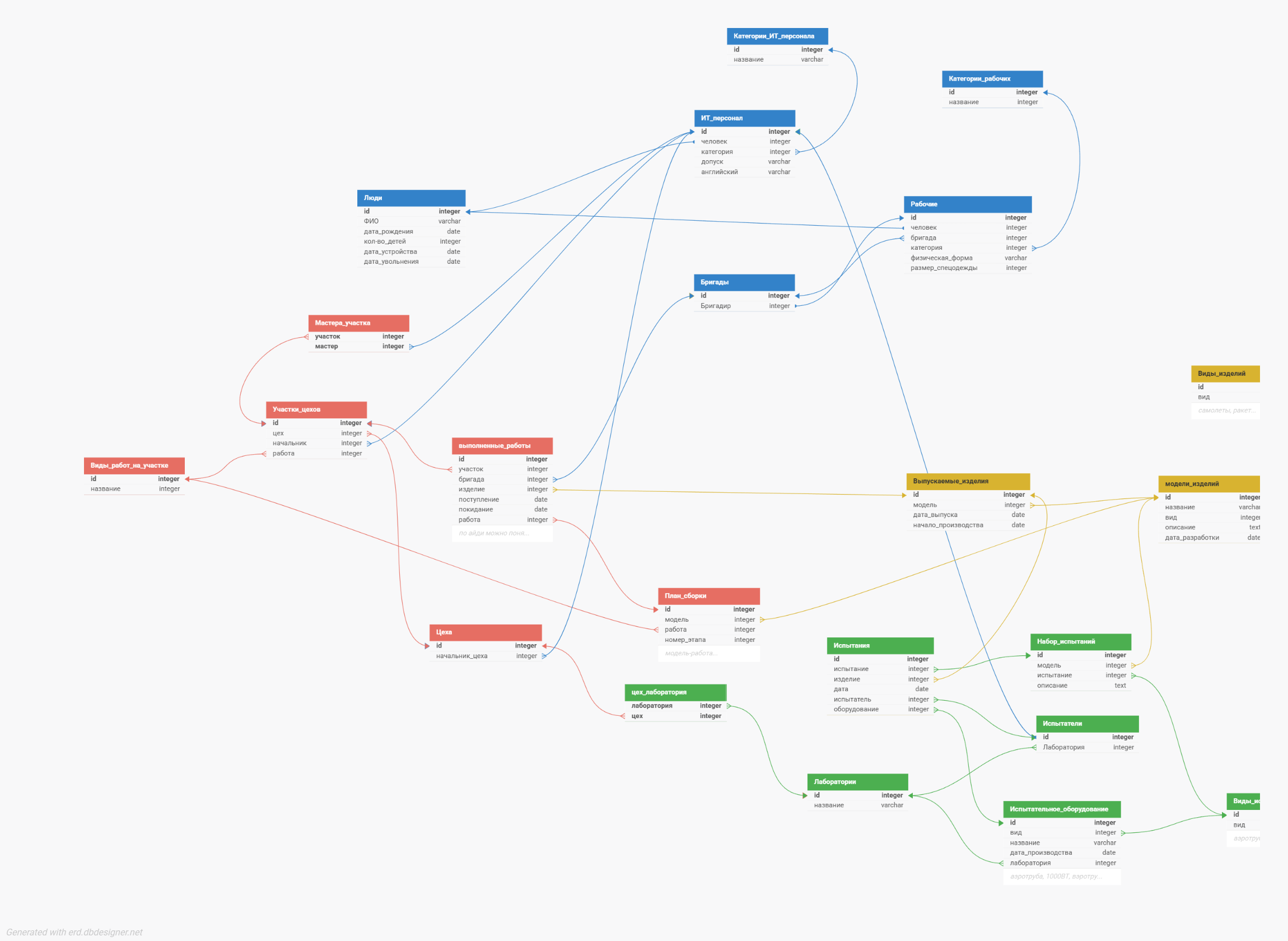
## СУЩНОСТИ И АТРИБУТЫ

|  |  |  |
| --- | --- | --- |
| **Сущность** | **Атрибуты** | **Описание** |
| Виды\_изделий | **Id**  Название | ИД вида  Название вида изделия |
| Модели\_изделий | **Id**  Вид  Название  Дата\_разработки  Описание | ИД модели  Категория изделия  Название модели  Дата разработки  Дополнительная информация |
| Выпускаемые\_изделия | **Id**  Модель  Начало\_производства  Дата\_выпуска | ИД выпущенного изделия  Модель изделия  Дата начала производства  Дата выпуска |
| Категории\_самолетов | **Id**  Название | ИД категории  Название категории |
| Самолеты | **Id**  Категория  Размах\_крыла  Кол-во\_мест  Макс\_скорость | ИД изделия  ИД категории самолета  Размах в м.  Кол-во пассажирских мест  Максимальная скорость в км/ч |
| Категории\_ракет | **Id**  Название | ИД категории  Название категории |
| Ракеты | **Id**  Категория  Мощность  Дальность  Вес | ИД изделия  ИД категории ракеты  Мощность в ТНТ  Дальность полета в КМ  Вес снаряженной ракеты в КГ |
| Вертолеты | **Id**  Грузоподъемность  Высота\_подъёма  Макс\_скорость | ИД изделия  Допустимый вес  Максимальная высота взлета  Максимальная скорость полета |
| Планеры | **Id**  Вес  Размах\_крыла  Макс\_скорость  Аэро\_качество | ИД изделия  Вес планера  Размах крыла  Максимальная скорость  Аэродинамическое качество |
| Дельтапланы | **Id**  Вес  Макс\_скорость Угол\_веера | ИД изделия  Вес дельтаплана  Максимальная высота  Угол веера крыла |
| Категории\_ИТ\_персонала | **Id**  Название | ИД категории  Название категории |
| Категории\_рабочих | **Id**  Название | ИД категории  Название категории |
| Люди | **Id**  ФИО  Дата\_рождения  Кол-во\_детей  Дата\_устройства  Дата\_увольнения | ИД человека  ФИО человека  Дата рождения человека  Кол-во детей человека  Дата устройства на должность  Дата увольнения |
| ИТ\_персонал | **Id**  Человек  Категория  Допуска  Английский | ИД персонала  ИД человека  ИД категории ИТ персонала  Допуск секретности  Знания английского |
| Рабочие | **Id**  Человек  Бригада  Категория  Физическая\_форма  Размер\_спецодежды | ИД рабочего  ИД человека  ИД бригады рабочего  ИД категории рабочего  Физическая подготовка  Размер рабочей одежды |
| Бригады | **Id**  Бригадир | ИД бригады  ИД рабочего-бригадира |
| Цеха | **Id**  Начальник\_цеха | ИД цеха  ИД ИТ\_персонала |
| Виды\_работ\_на\_участке | **Id**  Название | ИД работы  Название работы |
| Мастера\_участка | **Участок**  **Мастер** | ИД участка цеха  ИД ИТ персонала |
| Участки\_цехов | **Id**  Цех  Начальник\_участка  Вид\_работы | ИД участка  ИД цеха  ИД ИТ персонала  ИД работы на участке |
| План\_сборки | **Id**  Модель  Работа  Номер\_этапа | ИД этапа сборки  ИД модели изделия для сборки  ИД работы  Порядковый номер работы |
| Выполненные\_работы | **Id**  Участок  Бригада  Изделие  Поступление  Покидание  Работа | ИД выполнения работы  ИД участка цеха  ИД работающей бригады  ИД строящегося изделия  Дата поступления изделия  Дата окончания работы  ИД этапа сборки изделия |
| Набор\_испытаний | **Id**  Модель  Испытание  Описание | ИД испытания  ИД модели изделия  ИД испытательного Оборудования  Дополнительная информация |
| Виды\_оборудования | **Id**  Название | ИД вида  Название вида |
| Испытательное\_оборудование | **Id**  Вид  Название  Дата\_производства  Лаборатория | ИД оборудования  ИД вида  Название оборудования  Год выпуска оборудования  ИД лаборатории нахождения |
| Лаборатории | **Id**  Название | ИД лаборатории  Название лаборатории |
| Цех\_лаборатория | Лаборатория  Цех | ИД лаборатории  ИД цеха |
| Испытатели | **Id**  Лаборатория | ИД ИТ персонала  ИД лаборатории |
| Испытания | **Id**  Испытание  Изделие  Дата  Испытатель  Оборудование | ИД проведенного испытания  ИД испытания  ИД изделия  Дата проведения испытания  ИД испытателя  ИД оборудования |

## СУЩНОСТИ И СВЯЗИ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Связанные сущности** | **Тип связи** | **Описание** | **Свойства** |
| R1 | Категории\_ИТ\_персонала-ИТ\_персонал | 1:N | Каждый сотрудник должен иметь специализацию | Нельзя удалить категории, пока есть сотрудник с ней |
| R2 | Категории\_рабочих-Рабочие | 1:N | Каждый рабочий имеет специализацию | Нельзя удалить категории, пока есть рабочий с ней |
| R3 | Люди-ИТ\_персонал | 1:1 | Связь профессиональной информации и общей о человеке | Сотрудник не может существовать без общей информации |
| R4 | Люди-Рабочие | 1:1 | Связь профессиональной информации и общей о человеке | Рабочий не может существовать без общей информации |
| R5 | Рабочие-Бригады | N:1 | Несколько рабочих может состоять в одной бригаде | Рабочий может существовать без бригады |
| R6 | Бригады-Рабочие | 1:1 | У бригады обязан быть один бригадир | Бригада существует вокруг бригадира |
| R7 | Участки\_цехов-ИТ\_персонал | N:N | Мастера участка назначаются из ИТ персонала и связываются с участком цеха | При удалении участка все мастера распадаются |
| R8 | Виды\_работ\_на\_участке-Участки\_цеха | 1:N | На каждом участке производится определенный вид работ | Нельзя удалить вид работы, если она реализуется хотя бы на 1 участке |
| R9 | Участки\_цехов-Выполненные\_работы | 1:N | На одном участке может производиться несколько работ | В учете работ всегда есть связь с местом выполнения работ |
| R10 | Бригады -Выполненные\_работы | 1:N | Бригада может работать над несколькими работами | В учете работ всегда есть связь с бригадой |
| R11 | Выпускаемые\_изделия-Выполненные\_работы | 1:N | Каждое изделие проходит несколько этапов работ | В учете работ всегда есть связь с изделием |
| R12 | План\_сборки-Выполненные\_работы | 1:N | План сборки реализуется для множества изделий этого вида | В учете работ всегда есть связь с планом работ |
| R13 | ИТ\_персонал-Цеха | 1:N | У цеха есть один начальник | Цех не существует без начальник |
| R14 | Цеха-Участки\_цехов | 1:N | Цех содержит несколько участков | При удалении цеха удаляются его участки |
| R15 | Цех-Лаборатория | N:N | Каждый цех может сотрудничать с несколькими лабораториями | Если цех или лаборатория исчезают, то и связи вместе с ними |
| R16 | Виды\_работ\_на\_участке-План\_сборки | 1:N | Одна и та же работа применяется при производстве разных изделий | Нельзя удалить работу, пока она находится в плане |
| R17 | Модели\_изделий-План\_сборки | 1:N | Каждая модель изделия может иметь несколько этапов сборки | План сборки существует в рамках существования модели изделия |
| R18 | Модели\_изделий-Выпускаемые\_изделия | 1:N | Можно собрать несколько изделий одной модели | Изделие всегда имеет модель |
| R19 | Виды\_изделий-Модели\_изделий | 1:N | Может существовать несколько изделий одного вида | Изделие и вид не существуют раздельно |
| R20 | Категории\_самолетов-Самолеты | 1:N | Существует несколько самолетов одной категории | Самолет не существует без категории |
| R21 | Категории\_ракет-Ракеты | 1:N | Существует несколько ракет одной категории | Ракета не существует без категории |
| R22 | Модели\_изделий-Самолеты | 1:1 | Модель самолета соответствует одной модели изделий | Самолет может существовать без айди изделия, но тогда его нельзя будет собрать |
| R23 | Модели\_изделий-Ракеты | 1:1 | Модель ракеты соответствует одной модели изделий | Ракета может существовать без айди изделия, но тогда его нельзя будет собрать |
| R24 | Модели\_изделий-Вертолеты | 1:1 | Модель вертолета соответствует одной модели изделий | Вертолет может существовать без айди изделия, но тогда его нельзя будет собрать |
| R25 | Модели\_изделий-Планеры | 1:1 | Модель планера соответствует одной модели изделий | Планер может существовать без айди изделия, но тогда его нельзя будет собрать |
| R26 | Модели\_изделий-Дельтапланы | 1:1 | Модель дельтаплаа соответствует одной модели изделий | Дельтаплан может существовать без айди изделия, но тогда его нельзя будет собрать |
| R27 | Модели\_изделий-Набор\_испытаний | 1:N | Модель может испытываться несколькими испытаниями | Испытания существуют только при существовании модели |
| R28 | Виды\_испытательного\_оборудования-Испытательное\_оборудование | 1:N | Существует несколько моделей оборудования одного вида | Оборудование существует при существовании вида |
| R29 | Виды\_испытательного\_оборудования-Набор\_испытаний | 1:N | Испытания повторяются для разных моделей | Испытание существует при существовании вида испытания |
| R30 | Лаборатории-Испытательное\_оборудование | 1:N | В лаборатории находится разное оборудование | Лаборатория существует пока в ней находится оборудование |
| R31 | Испытания-Испытательное\_оборудование | N:1 | Но одном оборудовании производится несколько испытаний | Испытания содержат информацию об оборудовании |
| R32 | Испытания-Испытатели | N:1 | Испытатели принимают участие в разных испытаниях | Испытания содержат информацию об испытателях |
| R33 | Испытания-Набор\_испытаний | N:1 | Одно испытание из плана может относиться к разным изделиям | Испытания содержат информацию о проведенном испытании |
| R34 | Испытания-Выпускаемые\_изделия | N:1 | Для конкретного изделия проводится список испытаний | Испытания содержат информацию об изделии испытания |
| R35 | Испытатели-ИТ\_персонал | 1:1 | Каждый испытатель привязан к одной лаборатории | Испытатель существует пока работает ИТ\_персонал |
| R36 | Лаборатории-Испытатели | 1:N | В одной лаборатории может работать несколько испытателей | При удалении лаборатории удаляются её испытатели |

Ссылка на схему DBDesigner - <https://dbdesigner.page.link/ai8oEkUxcSSLsey97>



Изображение выглядит как текст, диаграмма, снимок экрана, карта

Контент, сгенерированный ИИ, может содержать ошибки.

## ОГРАНИЧЕНИЯ ЦЕЛОСТНОСТИ

1. Поле “id” таблицы “Люди” должно быть связано лишь с одной записью в таблице “Рабочие” или в таблице “ИТ\_персонал”.
2. Бригадир должен состоять в бригаде, в которой он является бригадиром.
3. Поле “id” таблицы “Модели\_изделий” должно быть связано лишь с одной записью в одной таблице среди всех видов изделий: ”Самолеты”, “Вертолеты”, “Ракеты”, “Планеры”, “Дельтапланы”.
4. Испытание изделия должно быть из списка испытаний модели этого изделия и на соответствующем оборудовании.
5. Тестирование проводится испытателем, который был трудоустроен во время проведения испытания.

## СОЗДАНИЕ ТАБЛИЦ

1. **Люди**

CREATE TABLE IF NOT EXISTS "Люди" (

    "id" SERIAL PRIMARY KEY,

    "ФИО" VARCHAR(255) NOT NULL,

    "дата\_рождения" DATE NOT NULL,

    "кол-во\_детей" INTEGER NOT NULL CHECK ("кол-во\_детей" >= 0),

    "дата\_устройства" DATE NOT NULL,

    "дата\_увольнения" DATE,

    UNIQUE ("ФИО", "дата\_рождения"),

    CONSTRAINT "дата\_увольнения\_после\_устройства"

        CHECK ("дата\_увольнения" IS NULL OR "дата\_увольнения" > "дата\_устройства"),

    CONSTRAINT "дата\_рождения\_до\_устройства"

        CHECK ("дата\_рождения" < "дата\_устройства")

);

1. **Категории\_ИТ\_персонала**

CREATE TABLE IF NOT EXISTS "Категории\_ИТ\_персонала"

(

    "id"       serial PRIMARY KEY,

    "название" varchar(255) NOT NULL UNIQUE

);

1. **ИТ\_персонал**

CREATE TABLE IF NOT EXISTS "ИТ\_персонал" (

    "id" SERIAL PRIMARY KEY,

    "человек" INTEGER NOT NULL UNIQUE REFERENCES "Люди"("id") ON DELETE CASCADE ON UPDATE RESTRICT,

    "категория" INTEGER NOT NULL REFERENCES "Категории\_ИТ\_персонала"("id") ON DELETE CASCADE ON UPDATE RESTRICT,

    "уровень\_допуска" CHAR NOT NULL CHECK ("уровень\_допуска" IN ('A', 'B', 'C')),

    "знания\_английского" varchar(255) NOT NULL CHECK ("знания\_английского" IN ('A1', 'A2', 'B1', 'B2', 'C1', 'C2'))

);

1. **Категории\_рабочих**

CREATE TABLE IF NOT EXISTS "Категории\_рабочих" (

    "id" serial PRIMARY KEY,

    "название" INTEGER NOT NULL UNIQUE

);

1. **Рабочие**

CREATE TABLE IF NOT EXISTS "Рабочие" (

    "id" serial PRIMARY KEY,

    "человек" INTEGER NOT NULL UNIQUE REFERENCES "Люди"("id") ON DELETE CASCADE ON UPDATE RESTRICT,

    "бригада" INTEGER,

    "категория" INTEGER NOT NULL REFERENCES "Категории\_рабочих"("id") ON DELETE CASCADE ON UPDATE RESTRICT,

    "физическая\_форма" varchar(255) NOT NULL CHECK ("физическая\_форма" IN ('отличная', 'хорошая', 'удовлетворительная', 'плохая')),

    "размер\_спецодежды" INTEGER NOT NULL CHECK ("размер\_спецодежды" BETWEEN 40 AND 60)

);

1. **Бригады**

CREATE TABLE IF NOT EXISTS "Бригады" (

    "id" serial PRIMARY KEY,

    "Бригадир" INTEGER NOT NULL REFERENCES "Рабочие"("id") ON DELETE CASCADE ON UPDATE RESTRICT

);

ALTER TABLE "Рабочие" ADD CONSTRAINT "Рабочие\_fk2" FOREIGN KEY ("бригада") REFERENCES "Бригады"("id") ON DELETE CASCADE ON UPDATE RESTRICT;

1. **Виды\_изделий**

CREATE TABLE IF NOT EXISTS "Виды\_изделий" (

    "id" SERIAL PRIMARY KEY,

    "название" VARCHAR(255) NOT NULL UNIQUE

);

1. **Модели\_изделий**

CREATE TABLE IF NOT EXISTS "Модели\_изделий" (

    "id" SERIAL PRIMARY KEY,

    "вид" INTEGER NOT NULL REFERENCES "Виды\_изделий"("id") ON DELETE CASCADE ON UPDATE RESTRICT,

    "название" VARCHAR(255) NOT NULL,

    "описание" VARCHAR(255) NOT NULL,

    "дата\_разработки" DATE NOT NULL

);

1. **Категории\_самолетов**

CREATE TABLE IF NOT EXISTS "Категории\_самолетов" (

    "id" SERIAL PRIMARY KEY,

    "название" VARCHAR(255) NOT NULL UNIQUE

);

1. **Самолеты**

CREATE TABLE IF NOT EXISTS "Самолеты" (

    "id" INTEGER NOT NULL UNIQUE REFERENCES "Модели\_изделий"("id") ON DELETE CASCADE ON UPDATE RESTRICT,

    "категория" INTEGER NOT NULL REFERENCES "Категории\_самолетов"("id") ON DELETE CASCADE ON UPDATE RESTRICT,

    "размах\_крыла" NUMERIC(6, 2) NOT NULL CHECK ("размах\_крыла" > 0),

    "кол-во\_мест" INTEGER NOT NULL CHECK ("кол-во\_мест" > 0),

    "макс\_скорость" INTEGER NOT NULL CHECK ("макс\_скорость" BETWEEN 0 AND 300000000)

);

1. **Категории\_ракет**

CREATE TABLE IF NOT EXISTS "Категории\_ракет" (

    "id" SERIAL PRIMARY KEY,

    "название" VARCHAR(255) NOT NULL UNIQUE

);

1. **Ракеты**

CREATE TABLE IF NOT EXISTS "Ракеты" (

    "id" INTEGER NOT NULL UNIQUE REFERENCES "Модели\_изделий"("id") ON DELETE CASCADE ON UPDATE RESTRICT,

    "категория" INTEGER NOT NULL REFERENCES "Категории\_ракет"("id") ON DELETE CASCADE ON UPDATE RESTRICT,

    "мощность" NUMERIC(10, 2) NOT NULL CHECK ("мощность" > 0),

    "дальность" INTEGER NOT NULL CHECK ("дальность" > 0),

    "вес" NUMERIC(10, 2) NOT NULL CHECK ("вес" > 0)

);

1. **Вертолеты**

CREATE TABLE IF NOT EXISTS "Вертолеты" (

    "id" INTEGER NOT NULL UNIQUE REFERENCES "Модели\_изделий"("id") ON DELETE CASCADE ON UPDATE RESTRICT,

    "грузоподъемность" NUMERIC(10, 2) NOT NULL CHECK ("грузоподъемность" > 0),

    "высота\_подъема" INTEGER NOT NULL CHECK ("высота\_подъема" BETWEEN 0 AND 100000),

    "макс\_скорость" INTEGER NOT NULL CHECK ("макс\_скорость" BETWEEN 0 AND 300000000)

);

1. **Планеры**

CREATE TABLE IF NOT EXISTS "Планеры" (

    "id" INTEGER NOT NULL UNIQUE REFERENCES "Модели\_изделий"("id") ON DELETE CASCADE ON UPDATE RESTRICT,

    "вес" INTEGER  NOT NULL CHECK ("вес" > 0),

    "размах\_крыла" INTEGER NOT NULL  CHECK ("размах\_крыла" > 0),

    "макс\_скорость" INTEGER  NOT NULL CHECK ("макс\_скорость" BETWEEN 0 AND 300000000),

    "аэро\_качество" NUMERIC(5,2) NOT NULL CHECK ("аэро\_качество" > 0 )

);

1. **Дельтапланы**

CREATE TABLE IF NOT EXISTS "Дельтапланы" (

    "id" INTEGER NOT NULL UNIQUE REFERENCES "Модели\_изделий"("id") ON DELETE CASCADE ON UPDATE RESTRICT,

    "вес" INTEGER  NOT NULL CHECK ("вес" > 0),

    "макс\_скорость" INTEGER  NOT NULL CHECK ("макс\_скорость" BETWEEN 0 AND 300000000),

    "угол\_веера" NUMERIC(5,2)  NOT NULL CHECK ("угол\_веера" BETWEEN 0 AND 180)

);

1. **Выпускаемые\_изделия**

CREATE TABLE IF NOT EXISTS "Выпускаемые\_изделия" (

    "id" SERIAL PRIMARY KEY,

    "модель" INTEGER NOT NULL REFERENCES "Модели\_изделий"("id") ON DELETE CASCADE ON UPDATE RESTRICT,

    "начало\_производства" DATE NOT NULL CHECK ("начало\_производства" <= CURRENT\_DATE),

    "дата\_выпуска" DATE CHECK ("дата\_выпуска" >= "начало\_производства" OR "дата\_выпуска" IS NULL)

);

1. **Цеха**

CREATE TABLE IF NOT EXISTS "Цеха" (

    "id" SERIAL PRIMARY KEY,

    "начальник\_цеха" INTEGER NOT NULL REFERENCES "ИТ\_персонал"("id") ON DELETE RESTRICT ON UPDATE CASCADE

);

1. **Виды\_работ\_на\_участке**

CREATE TABLE IF NOT EXISTS "Виды\_работ\_на\_участке" (

    "id" SERIAL PRIMARY KEY,

    "название" VARCHAR(255) NOT NULL UNIQUE

);

1. **Участки\_цехов**

CREATE TABLE IF NOT EXISTS "Участки\_цехов" (

    "id" SERIAL PRIMARY KEY,

    "цех" INTEGER NOT NULL REFERENCES "Цеха"("id") ON DELETE CASCADE ON UPDATE RESTRICT,

    "начальник" INTEGER NOT NULL REFERENCES "ИТ\_персонал"("id") ON DELETE RESTRICT ON UPDATE CASCADE,

    "вид\_работы" INTEGER NOT NULL REFERENCES "Виды\_работ\_на\_участке"("id") ON DELETE RESTRICT ON UPDATE RESTRICT

);

1. **Мастера\_участка**

CREATE TABLE IF NOT EXISTS "Мастера\_участка" (

    "участок" INTEGER NOT NULL REFERENCES "Участки\_цехов"("id") ON DELETE CASCADE ON UPDATE CASCADE,

    "мастер" INTEGER NOT NULL REFERENCES "ИТ\_персонал"("id") ON DELETE CASCADE ON UPDATE CASCADE,

    PRIMARY KEY ("участок", "мастер")

);

1. **План\_сборки**

CREATE TABLE IF NOT EXISTS "План\_сборки" (

    "id" SERIAL PRIMARY KEY,

    "модель" INTEGER NOT NULL REFERENCES "Модели\_изделий"("id") ON DELETE CASCADE ON UPDATE CASCADE,

    "работа" INTEGER NOT NULL REFERENCES "Виды\_работ\_на\_участке"("id") ON DELETE RESTRICT ON UPDATE RESTRICT

);

1. **Выполненные\_работы**

CREATE TABLE IF NOT EXISTS "Выполненные\_работы" (

    "id" SERIAL PRIMARY KEY,

    "участок" INTEGER NOT NULL REFERENCES "Участки\_цехов"("id") ON DELETE RESTRICT ON UPDATE RESTRICT,

    "бригада" INTEGER NOT NULL REFERENCES "Бригады"("id") ON DELETE RESTRICT ON UPDATE RESTRICT,

    "изделие" INTEGER NOT NULL REFERENCES "Выпускаемые\_изделия"("id") ON DELETE RESTRICT ON UPDATE RESTRICT,

    "этап" INTEGER NOT NULL REFERENCES "План\_сборки"("id") ON DELETE RESTRICT ON UPDATE RESTRICT,

    "поступление" DATE NOT NULL CHECK ("поступление" <= CURRENT\_DATE),

    "покидание" DATE CHECK ("покидание" >= "поступление" OR "покидание" IS NULL),

    UNIQUE("участок", "бригада", "изделие", "этап")

);

1. **Виды\_испытательного\_оборудования**

CREATE TABLE IF NOT EXISTS "Виды\_испытательного\_оборудования" (

    "id" SERIAL PRIMARY KEY,

    "название" VARCHAR(255) NOT NULL UNIQUE

);

1. **Лаборатории**

CREATE TABLE IF NOT EXISTS "Лаборатории" (

    "id" SERIAL PRIMARY KEY,

    "название" VARCHAR(255) NOT NULL UNIQUE

);

1. **Лаборатория\_цех**

CREATE TABLE IF NOT EXISTS "Лаборатория\_цех" (

    "лаборатория" INTEGER REFERENCES "Лаборатории"("id") ON DELETE CASCADE ON UPDATE CASCADE,

    "цех" INTEGER REFERENCES "Цеха"("id") ON DELETE CASCADE ON UPDATE CASCADE,

    PRIMARY KEY ("лаборатория", "цех")

);

1. **Испытатели**

CREATE TABLE IF NOT EXISTS "Испытатели" (

    "id" INTEGER UNIQUE REFERENCES "ИТ\_персонал"("id")  ON DELETE RESTRICT ON UPDATE CASCADE,

    "лаборатория" INTEGER REFERENCES "Лаборатории"("id") ON DELETE CASCADE ON UPDATE CASCADE

);

1. **Набор\_испытаний**

CREATE TABLE IF NOT EXISTS "Набор\_испытаний" (

    "id" SERIAL PRIMARY KEY,

    "модель" INTEGER NOT NULL REFERENCES "Модели\_изделий"("id") ON DELETE CASCADE ON UPDATE CASCADE,

    "вид\_испытания" INTEGER NOT NULL REFERENCES "Виды\_испытательного\_оборудования"("id") ON DELETE RESTRICT ON UPDATE CASCADE,

    "описание" TEXT,

    UNIQUE ("модель", "вид\_испытания")

);

1. **Испытательное\_оборудование**

CREATE TABLE IF NOT EXISTS "Испытательное\_оборудование" (

    "id" SERIAL PRIMARY KEY,

    "вид" INTEGER NOT NULL REFERENCES "Виды\_испытательного\_оборудования"("id") ON DELETE RESTRICT ON UPDATE CASCADE,

    "название" VARCHAR(255) NOT NULL,

    "дата\_производства" DATE NOT NULL,

    "лаборатория" INTEGER NOT NULL REFERENCES "Лаборатории"("id") ON DELETE RESTRICT ON UPDATE CASCADE

);

1. **Испытания**

CREATE TABLE IF NOT EXISTS "Испытания" (

    "id" SERIAL PRIMARY KEY,

    "испытание" INTEGER NOT NULL REFERENCES "Набор\_испытаний"("id") ON DELETE CASCADE ON UPDATE CASCADE,

    "изделие" INTEGER NOT NULL REFERENCES "Выпускаемые\_изделия"("id") ON DELETE CASCADE ON UPDATE CASCADE,

    "дата" DATE NOT NULL CHECK ("дата" <= CURRENT\_DATE),

    "испытатель" INTEGER NOT NULL REFERENCES "Испытатели"("id") ON DELETE RESTRICT ON UPDATE CASCADE,

    "оборудование" INTEGER NOT NULL REFERENCES "Испытательное\_оборудование"("id") ON DELETE RESTRICT ON UPDATE CASCADE

);

# Написание SQL запросов к спроектированной базе данных.

1. Получить перечень видов изделий отдельной категории и в целом, собираемых указанным цехом, предприятием.

CREATE OR REPLACE FUNCTION перечень\_изделий(

    p\_цех\_id INTEGER DEFAULT NULL,

    p\_категория\_изделия TEXT DEFAULT NULL

)

RETURNS TABLE (

    id\_модели INTEGER,

    название\_модели TEXT,

    категория TEXT,

    цех\_id INTEGER

) AS $$

BEGIN

    RETURN QUERY

    WITH Категории\_изделий AS (

        SELECT m."id", m."название"::text AS "название", 'Самолеты'::text AS "категория"

        FROM "Модели\_изделий" m JOIN "Самолеты" s ON m."id" = s."id"

        UNION ALL

        SELECT m."id", m."название"::text, 'Ракеты'::text

        FROM "Модели\_изделий" m JOIN "Ракеты" r ON m."id" = r."id"

        UNION ALL

        SELECT m."id", m."название"::text, 'Вертолеты'::text

        FROM "Модели\_изделий" m JOIN "Вертолеты" v ON m."id" = v."id"

        UNION ALL

        SELECT m."id", m."название"::text, 'Планеры'::text

        FROM "Модели\_изделий" m JOIN "Планеры" p ON m."id" = p."id"

        UNION ALL

        SELECT m."id", m."название"::text, 'Дельтапланы'::text

        FROM "Модели\_изделий" m JOIN "Дельтапланы" d ON m."id" = d."id"

    )

    SELECT DISTINCT

        k."id" AS id\_модели,

        k."название" AS название\_модели,

        k."категория",

        c."id" AS цех\_id

    FROM "Выполненные\_работы" wr

    JOIN "Участки\_цехов" u ON wr."участок" = u."id"

    JOIN "Цеха" c ON u."цех" = c."id"

    JOIN "Выпускаемые\_изделия" vi ON wr."изделие" = vi."id"

    JOIN Категории\_изделий k ON vi."модель" = k."id"

    WHERE

        (p\_цех\_id IS NULL OR c."id" = p\_цех\_id)

        AND (p\_категория\_изделия IS NULL OR k."категория" = p\_категория\_изделия)

    ORDER BY c.id;

END;

$$ LANGUAGE plpgsql;

1. Получить число и перечень изделий отдельной категории и в целом, собранных указанным цехом, участком, предприятием в целом за определенный отрезок времени.

CREATE OR REPLACE FUNCTION получить\_изделия2(

    цех\_id INTEGER,

    участок\_id INTEGER,

    категория\_изделия TEXT,

    дата\_с DATE,

    дата\_по DATE

)

RETURNS TABLE (

    id\_модели INTEGER,

    название\_модели TEXT,

    категория TEXT,

    количество BIGINT

) AS $$

BEGIN

    RETURN QUERY

    WITH Категории\_изделий AS (

        SELECT m."id", m."название"::TEXT, 'Самолеты'::TEXT AS "категория" FROM "Модели\_изделий" m JOIN "Самолеты" s ON m."id" = s."id"

        UNION ALL

        SELECT m."id", m."название"::TEXT, 'Ракеты'::TEXT FROM "Модели\_изделий" m JOIN "Ракеты" r ON m."id" = r."id"

        UNION ALL

        SELECT m."id", m."название"::TEXT, 'Вертолеты'::TEXT FROM "Модели\_изделий" m JOIN "Вертолеты" v ON m."id" = v."id"

        UNION ALL

        SELECT m."id", m."название"::TEXT, 'Планеры'::TEXT FROM "Модели\_изделий" m JOIN "Планеры" p ON m."id" = p."id"

        UNION ALL

        SELECT m."id", m."название"::TEXT, 'Дельтапланы'::TEXT FROM "Модели\_изделий" m JOIN "Дельтапланы" d ON m."id" = d."id"

    )

    SELECT

        k."id" AS id\_модели,

        k."название"::TEXT AS название\_модели,

        k."категория"::TEXT AS категория,

        COUNT(\*) AS количество

    FROM "Выполненные\_работы" wr

    JOIN "Участки\_цехов" u ON wr."участок" = u."id"

    JOIN "Цеха" c ON u."цех" = c."id"

    JOIN "Выпускаемые\_изделия" vi ON wr."изделие" = vi."id"

    JOIN Категории\_изделий k ON vi."модель" = k."id"

    WHERE

        (цех\_id IS NULL OR c."id" = цех\_id) AND

        (участок\_id IS NULL OR u."id" = участок\_id) AND

        (категория\_изделия IS NULL OR k."категория" = категория\_изделия) AND

        wr."поступление" BETWEEN дата\_с AND дата\_по

    GROUP BY k."id", k."название", k."категория";

END;

$$ LANGUAGE plpgsql;

1. Получить данные о кадровом составе цеха, предприятия в целом и по указанным категориям инженерно-технического персонала и рабочих.

CREATE OR REPLACE FUNCTION кадровый\_состав(

    цех\_id INTEGER DEFAULT NULL,

    категория\_ит TEXT DEFAULT NULL,

    категория\_рабочих TEXT DEFAULT NULL

)

RETURNS TABLE (

    id\_человека INTEGER,

    фио TEXT,

    категория TEXT,

    тип TEXT,

    цех INTEGER

) AS $$

BEGIN

    RETURN QUERY

    -- ИТ-персонал

    SELECT

        p."id",

        p."ФИО"::TEXT,

        кат\_ит."название"::TEXT,

        'ИТ\_персонал'::TEXT,

        ц."id"

    FROM "ИТ\_персонал" i

    JOIN "Люди" p ON p."id" = i."id"

    JOIN "Категории\_ИТ\_персонала" кат\_ит ON кат\_ит."id" = i."категория"

    LEFT JOIN "Цеха" ц ON i."id" = ц."начальник\_цеха"

    WHERE

        (категория\_ит IS NULL OR кат\_ит."название" = категория\_ит)

        AND (цех\_id IS NULL OR ц."id" = цех\_id)

    UNION ALL

    -- Рабочие

    SELECT

        p."id",

        p."ФИО"::TEXT,

        кат\_раб."название"::TEXT,

        'Рабочие'::TEXT,

        NULL

    FROM "Рабочие" r

    JOIN "Люди" p ON p."id" = r."id"

    JOIN "Категории\_рабочих" кат\_раб ON кат\_раб."id" = r."категория"

    WHERE

        (категория\_рабочих IS NULL OR кат\_раб."название" = категория\_рабочих);

END;

$$ LANGUAGE plpgsql;

1. Получить число и перечень участков указанного цеха, предприятия в целом и их начальников.

CREATE OR REPLACE FUNCTION перечень\_участков(

    цех\_id INTEGER DEFAULT NULL

)

RETURNS TABLE (

    id\_участка INTEGER,

    начальник\_id INTEGER,

    начальник\_ФИО TEXT,

    цех INTEGER,

    количество\_участков BIGINT

) AS $$

BEGIN

    RETURN QUERY

    SELECT

        у."id",

        у."начальник",

        п."ФИО"::TEXT,

        у."цех",

        COUNT(\*) OVER (PARTITION BY у."цех") AS количество\_участков

    FROM "Участки\_цехов" у

    JOIN "ИТ\_персонал" ит ON ит."id" = у."начальник"

    JOIN "Люди" п ON п."id" = ит."id"

    WHERE (цех\_id IS NULL OR у."цех" = цех\_id);

END;

$$ LANGUAGE plpgsql;

1. Получить перечень работа, которые проходит указанное изделие.

CREATE OR REPLACE FUNCTION перечень\_работ\_изделия(

    изделие\_id INTEGER

)

RETURNS TABLE (

    этап INTEGER,

    вид\_работы TEXT

) AS $$

BEGIN

    RETURN QUERY

    SELECT

        п."этап",

        в."название"::TEXT

    FROM "Выпускаемые\_изделия" ви

    JOIN "План\_сборки" п ON ви."модель" = п."модель"

    JOIN "Виды\_работ\_на\_участке" в ON в."id" = п."работа"

    WHERE ви."id" = изделие\_id

    ORDER BY п."этап";

END;

$$ LANGUAGE plpgsql;

1. Получить состав бригад указанного участка, цеха.

CREATE OR REPLACE FUNCTION состав\_бригад(

    цех\_id INTEGER DEFAULT NULL,

    участок\_id INTEGER DEFAULT NULL

)

RETURNS TABLE (

    бригада\_id INTEGER,

    бригадир INTEGER,

    рабочий\_id INTEGER

) AS $$

BEGIN

    RETURN QUERY

    SELECT DISTINCT

        б."id" AS бригада\_id,

        б."бригадир" AS бригадир,

        р."id" AS рабочий\_id

    FROM "Бригады" б

    JOIN "Рабочие" р ON р."бригада" = б."id"

    JOIN "Выполненные\_работы" вр ON вр."бригада" = б."id"

    JOIN "Участки\_цехов" у ON у."id" = вр."участок"

    WHERE (цех\_id IS NULL OR у."цех" = цех\_id)

      AND (участок\_id IS NULL OR у."id" = участок\_id)

    ORDER BY б."id", р."id";

END;

$$ LANGUAGE plpgsql;

1. Получить список мастеров указанного участка, цеха.

CREATE OR REPLACE FUNCTION состав\_бригад2(

    цех\_id INTEGER DEFAULT NULL,

    участок\_id INTEGER DEFAULT NULL

)

RETURNS TABLE (

    участок INTEGER,

    мастер INTEGER

) AS $$

BEGIN

    RETURN QUERY

    SELECT DISTINCT

        у."id" AS участок,

        б."id" AS мастер

    FROM "ИТ\_персонал" б

    JOIN "Мастера\_участка" р ON р."мастер" = б."id"

    JOIN "Участки\_цехов" у ON у."id" = р."участок"

    WHERE (цех\_id IS NULL OR у."цех" = цех\_id)

      AND (участок\_id IS NULL OR у."id" = участок\_id)

    ORDER BY б."id", у."id";

END;

$$ LANGUAGE plpgsql;

1. Получить перечень изделий отдельной категории и в целом, собираемых в настоящий момент указанным участком, цехом, предприятием.

CREATE OR REPLACE FUNCTION текущие\_изделия(

    участок\_id INTEGER DEFAULT NULL,

    цех\_id INTEGER DEFAULT NULL,

    категория\_изделия TEXT DEFAULT NULL

)

RETURNS TABLE (

    id\_модели INTEGER,

    название\_модели TEXT,

    категория TEXT,

    изделие\_id INTEGER,

    участокid INTEGER,

    цехid INTEGER

) AS $$

BEGIN

    RETURN QUERY

    WITH Категории\_изделий AS (

        SELECT m."id", m."название"::TEXT, 'Самолеты'::TEXT AS "категория" FROM "Модели\_изделий" m JOIN "Самолеты" s ON m."id" = s."id"

        UNION ALL

        SELECT m."id", m."название"::TEXT, 'Ракеты'::TEXT FROM "Модели\_изделий" m JOIN "Ракеты" r ON m."id" = r."id"

        UNION ALL

        SELECT m."id", m."название"::TEXT, 'Вертолеты'::TEXT FROM "Модели\_изделий" m JOIN "Вертолеты" v ON m."id" = v."id"

        UNION ALL

        SELECT m."id", m."название"::TEXT, 'Планеры'::TEXT FROM "Модели\_изделий" m JOIN "Планеры" p ON m."id" = p."id"

        UNION ALL

        SELECT m."id", m."название"::TEXT, 'Дельтапланы'::TEXT FROM "Модели\_изделий" m JOIN "Дельтапланы" d ON m."id" = d."id"

    )

    SELECT DISTINCT

        k."id" AS id\_модели,

        k."название",

        k."категория",

        wr."изделие" AS изделие\_id,

        u."id" AS участокid,

        c."id" AS цехid

    FROM "Выполненные\_работы" wr

    JOIN "Участки\_цехов" u ON wr."участок" = u."id"

    JOIN "Цеха" c ON u."цех" = c."id"

    JOIN "Выпускаемые\_изделия" vi ON wr."изделие" = vi."id"

    JOIN Категории\_изделий k ON vi."модель" = k."id"

    WHERE wr."покидание" IS NULL

      AND (участок\_id IS NULL OR u."id" = участок\_id)

      AND (цех\_id IS NULL OR c."id" = цех\_id)

      AND (категория\_изделия IS NULL OR k."категория" = категория\_изделия);

END;

$$ LANGUAGE plpgsql;

# РЕАЛИЗАЦИЯ ТРИГГЕРОВ

1. **Уникальность человека среди рабочих и ИТ персонала**

CREATE OR REPLACE FUNCTION check\_person\_unique()

RETURNS TRIGGER AS $$

BEGIN

    IF TG\_TABLE\_NAME = 'Рабочие' THEN

        IF EXISTS (SELECT 1 FROM "ИТ\_персонал" WHERE "человек" = NEW."человек") THEN

            RAISE EXCEPTION 'Person with id % already exists in it table', NEW."человек";

        END IF;

    END IF;

    IF TG\_TABLE\_NAME = 'ИТ\_персонал' THEN

        IF EXISTS (SELECT 1 FROM "Рабочие" WHERE "человек" = NEW."человек") THEN

            RAISE EXCEPTION 'Person with id % already exists in workers table', NEW."человек";

        END IF;

    END IF;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS check\_worker\_unique ON "Рабочие";

CREATE TRIGGER check\_worker\_unique

BEFORE INSERT OR UPDATE ON "Рабочие"

FOR EACH ROW EXECUTE FUNCTION check\_person\_unique();

DROP TRIGGER IF EXISTS check\_it\_unique ON "ИТ\_персонал";

CREATE TRIGGER check\_it\_unique

BEFORE INSERT OR UPDATE ON "ИТ\_персонал"

FOR EACH ROW EXECUTE FUNCTION check\_person\_unique();

1. **Бригадир принадлежит бригаде, в которой он бригадир**

CREATE OR REPLACE FUNCTION check\_brigadir\_consistency()

RETURNS trigger AS $$

DECLARE

    brigade\_id INTEGER;

BEGIN

    SELECT id INTO brigade\_id

    FROM "Бригады"

    WHERE "бригадир" = NEW.id;

    IF brigade\_id IS NOT NULL THEN

        IF NEW.бригада IS DISTINCT FROM brigade\_id THEN

            RAISE EXCEPTION 'Рабочий (id=%) является бригадиром бригады (id=%), но указан в другой бригаде (id=%).',

                NEW.id, brigade\_id, NEW.бригада;

        END IF;

    END IF;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS trg\_check\_brigadir\_consistency ON "Рабочие";

CREATE TRIGGER trg\_check\_brigadir\_consistency

BEFORE INSERT OR UPDATE ON "Рабочие"

FOR EACH ROW

EXECUTE FUNCTION check\_brigadir\_consistency();

1. **Уникальность модели изделия среди всех её видов**

CREATE OR REPLACE FUNCTION check\_unique\_model\_type()

RETURNS TRIGGER AS $$

DECLARE

    cnt INTEGER;

BEGIN

    SELECT COUNT(\*) INTO cnt FROM (

        SELECT id FROM "Самолеты" WHERE id = NEW.id AND TG\_TABLE\_NAME <> 'Самолеты'

        UNION ALL

        SELECT id FROM "Вертолеты" WHERE id = NEW.id AND TG\_TABLE\_NAME <> 'Вертолеты'

        UNION ALL

        SELECT id FROM "Ракеты" WHERE id = NEW.id AND TG\_TABLE\_NAME <> 'Ракеты'

        UNION ALL

        SELECT id FROM "Планеры" WHERE id = NEW.id AND TG\_TABLE\_NAME <> 'Планеры'

        UNION ALL

        SELECT id FROM "Дельтапланы" WHERE id = NEW.id AND TG\_TABLE\_NAME <> 'Дельтапланы'

    ) AS all\_refs;

    IF cnt > 0 THEN

        RAISE EXCEPTION 'Модель с id % уже зарегистрирована в другой таблице изделий.', NEW.id;

    END IF;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS trg\_planes\_unique\_model ON "Самолеты";

CREATE TRIGGER trg\_planes\_unique\_model

BEFORE INSERT OR UPDATE ON "Самолеты"

FOR EACH ROW

EXECUTE FUNCTION check\_unique\_model\_type();

DROP TRIGGER IF EXISTS trg\_helicopters\_unique\_model ON "Вертолеты";

CREATE TRIGGER trg\_helicopters\_unique\_model

BEFORE INSERT OR UPDATE ON "Вертолеты"

FOR EACH ROW

EXECUTE FUNCTION check\_unique\_model\_type();

DROP TRIGGER IF EXISTS trg\_rockets\_unique\_model ON "Ракеты";

CREATE TRIGGER trg\_rockets\_unique\_model

BEFORE INSERT OR UPDATE ON "Ракеты"

FOR EACH ROW

EXECUTE FUNCTION check\_unique\_model\_type();

DROP TRIGGER IF EXISTS trg\_gliders\_unique\_model ON "Планеры";

CREATE TRIGGER trg\_gliders\_unique\_model

BEFORE INSERT OR UPDATE ON "Планеры"

FOR EACH ROW

EXECUTE FUNCTION check\_unique\_model\_type();

DROP TRIGGER IF EXISTS trg\_hang\_gliders\_unique\_model ON "Дельтапланы";

CREATE TRIGGER trg\_hang\_gliders\_unique\_model

BEFORE INSERT OR UPDATE ON "Дельтапланы"

FOR EACH ROW

EXECUTE FUNCTION check\_unique\_model\_type();

1. **Испытание изделия должно быть из списка испытаний этой модели и на соответствующем оборудовании**

CREATE OR REPLACE FUNCTION проверить\_вид\_испытания\_и\_оборудования()

RETURNS TRIGGER AS $$

DECLARE

    вид\_из\_набора INTEGER;

    вид\_из\_оборудования INTEGER;

BEGIN

    SELECT "вид\_испытания" INTO вид\_из\_набора

    FROM "Набор\_испытаний"

    WHERE id = NEW."испытание";

    SELECT "вид" INTO вид\_из\_оборудования

    FROM "Испытательное\_оборудование"

    WHERE id = NEW."оборудование";

    IF вид\_из\_набора IS DISTINCT FROM вид\_из\_оборудования THEN

        RAISE EXCEPTION 'Несовпадение вида испытания (% из набора) и вида оборудования (%) в id=%', вид\_из\_набора, вид\_из\_оборудования, NEW.id;

    END IF;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER триггер\_проверка\_вида\_испытания

BEFORE INSERT OR UPDATE ON "Испытания"

FOR EACH ROW

EXECUTE FUNCTION проверить\_вид\_испытания\_и\_оборудования();

1. **Тестирование проводится испытателем, который работал в это время**

CREATE OR REPLACE FUNCTION проверить\_дату\_работы\_испытателя()

RETURNS TRIGGER AS $$

DECLARE

    дата DATE := NEW."дата";

    дата\_устройства DATE;

    дата\_увольнения DATE;

BEGIN

    SELECT л."дата\_устройства", л."дата\_увольнения"

    INTO дата\_устройства, дата\_увольнения

    FROM "Люди" л

    JOIN "ИТ\_персонал" ит ON ит."id" = (

        SELECT "id"

        FROM "Испытатели"

        WHERE "id" = NEW."испытатель"

    )

    WHERE л."id" = ит."человек";

    IF дата < дата\_устройства THEN

        RAISE EXCEPTION 'Дата испытания (%), раньше даты устройства испытателя (%)', дата, дата\_устройства;

    END IF;

    IF дата\_увольнения IS NOT NULL AND дата >= дата\_увольнения THEN

        RAISE EXCEPTION 'Дата испытания (%), позже даты увольнения испытателя (%)', дата, дата\_увольнения;

    END IF;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER триггер\_проверка\_даты\_испытателя

BEFORE INSERT OR UPDATE ON "Испытания"

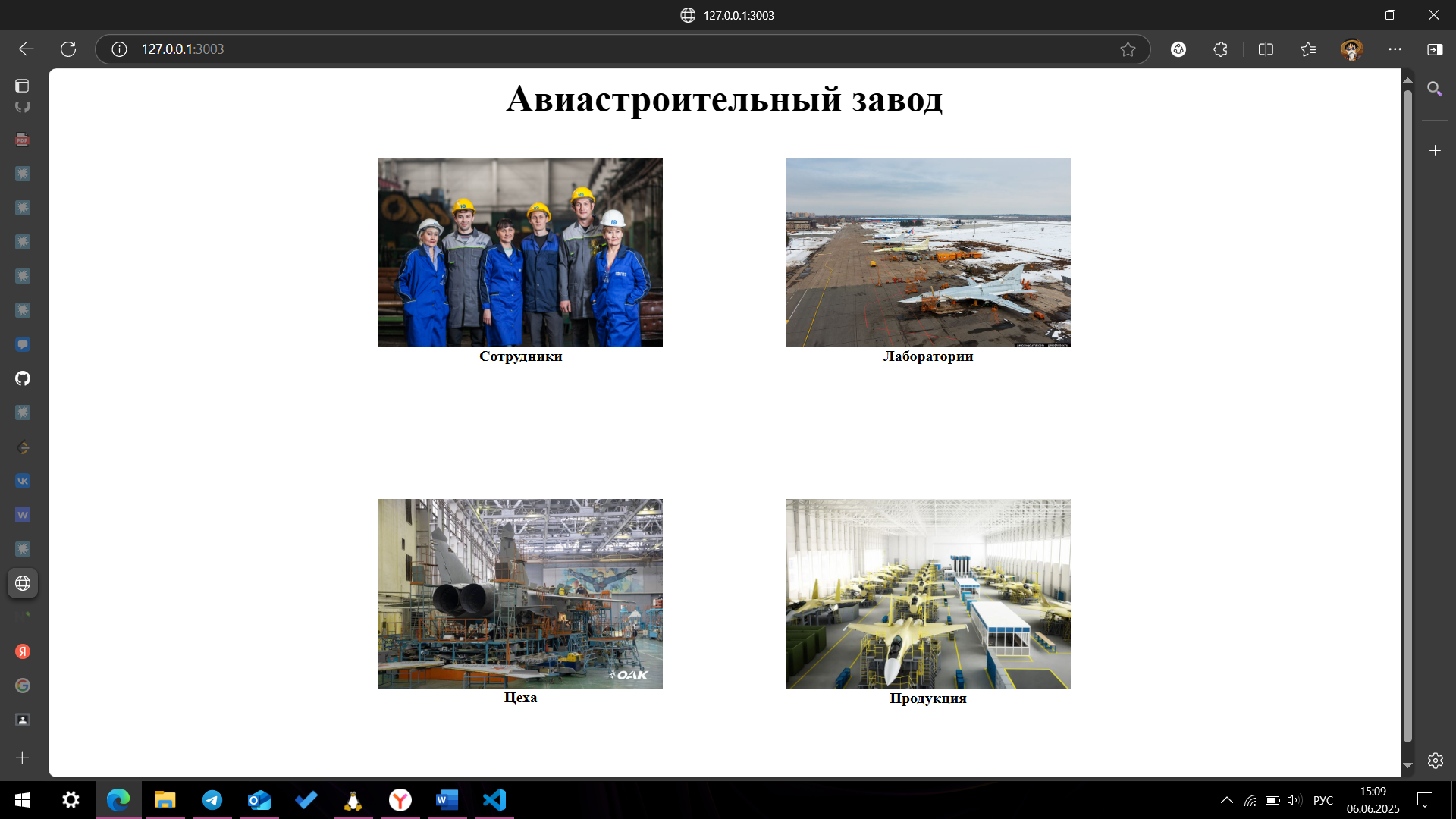
FOR EACH ROW

EXECUTE FUNCTION проверить\_дату\_работы\_испытателя();

# Разработка приложения клиента

Приложение разработано на языке программирования GO с использованием веб-фреймворка Fiber. Приложение доступно в любом браузере по адресу <http://127.0.0.1:3003/>.

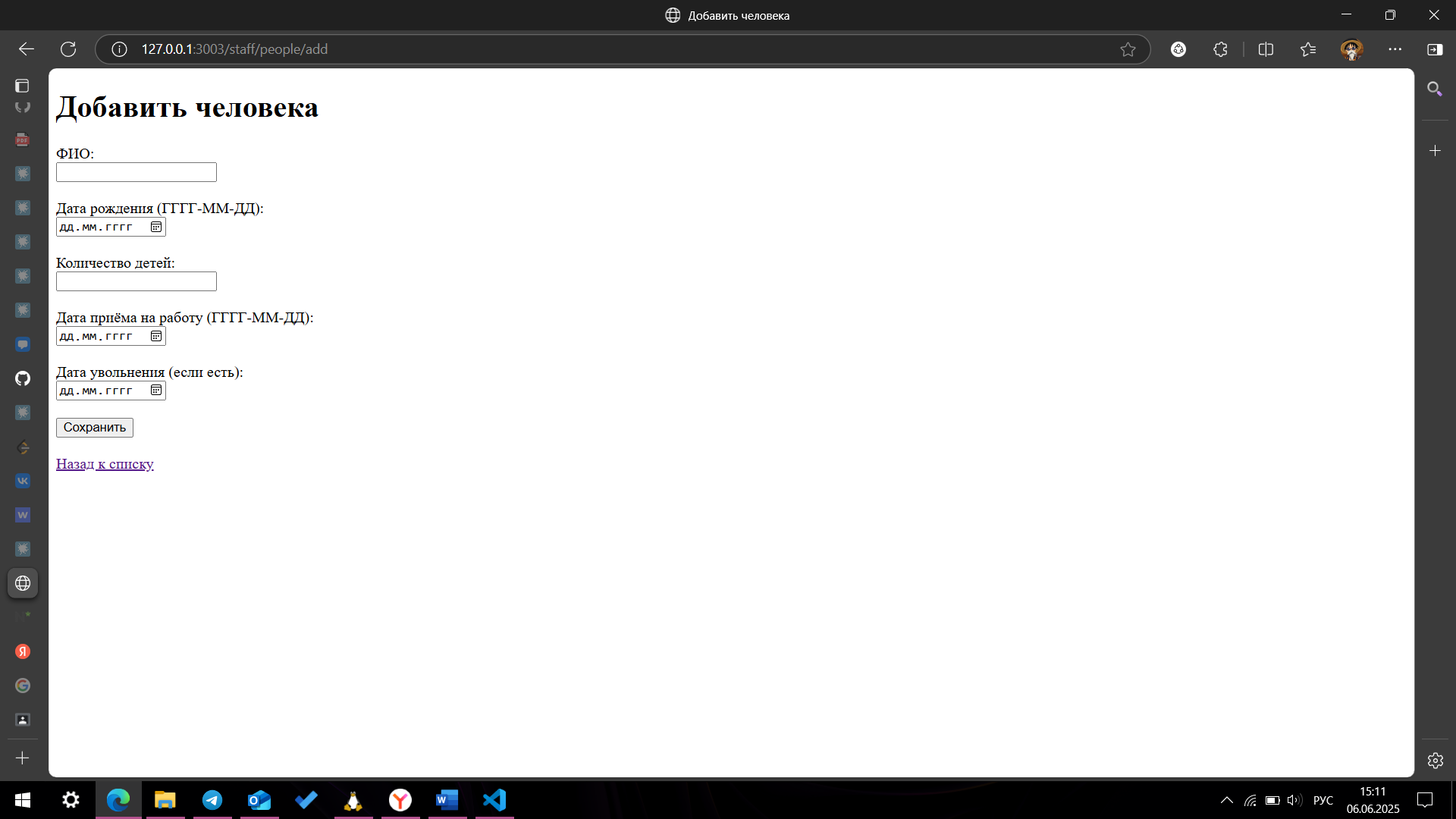
Каждая форма вывода – отрисованная html страница.



Скриншот 1. Главная страница приложения.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.Скриншот 2. Отображение списка людей.

Скриншот 3. Форма добавления человека.

Изображение выглядит как снимок экрана, программное обеспечение, компьютер, текст

Содержимое, созданное искусственным интеллектом, может быть неверным.Скриншот 4. Форма добавления рабочего с выпадающими списками связанных полей.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.Скриншот 5. Форма отображения цехов.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была разработана полноценная информационная система для авиастроительного предприятия, охватывающая все ключевые аспекты его деятельности.

Проект оказался масштабным и сложным, прежде всего из-за необходимости учесть множество сущностей и их связей, отражающих как иерархию предприятия, так и особенности сборки, испытаний и кадрового состава. Одной из главных трудностей стало проектирование гибкой и непротиворечивой схемы базы данных, способной хранить всю заданную информацию без избыточности. Это потребовало тщательного анализа условий задачи и построения логической модели, пригодной для реализации на уровне СУБД.

В процессе разработки были спроектированы и реализованы таблицы с учетом ограничений целостности, написаны SQL-запросы, реализованы триггеры, обеспечивающие автоматический контроль за допустимостью данных на уровне сервера.

Кроме того, была реализована клиентская часть приложения, обеспечивающая удобный ввод, редактирование и поиск данных по заранее сформулированным запросам. Разработка интерфейса показала, насколько важна роль клиентской части не только для взаимодействия пользователя с системой, но и как дополнительного механизма валидации данных и поддержания целостности, наряду с серверной логикой.

Интересной особенностью стало использование параметрических SQL-запросов, которые обеспечивают удобство повторного использования логики запросов с различными входными значениями.

Таким образом, проект продемонстрировал важность комплексного подхода к построению информационных систем: от тщательного проектирования структуры данных до реализации полноценного программного интерфейса и контроля целостности данных.

Полный код проекта доступен в репозитории моего профиля github - [NSU-Projects/DB/Client-server at main · Kanad4s/NSU-Projects](https://github.com/Kanad4s/NSU-Projects/tree/main/DB/Client-server)