

# 1. Термины и определения

## 1.1. Общие термины

**Система** – распределённая информационная система управления задачами, реализованная в рамках данного проекта.

**Пользователь** – физическое лицо, взаимодействующее с Системой (сотрудник, менеджер, HR, администратор).

**Филиал** – территориальное подразделение компании, имеющее собственную команду и проекты.

**Задача** – единица работы, назначенная сотруднику в рамках проекта с указанием срока, приоритета и статуса.

**Проект** – совокупность задач, объединённых общей целью и управляемых менеджером.

## 1.2. Бизнес термины

**Глобальная БД** – централизованное хранилище данных, содержащее справочную информацию, общую для всех филиалов (сотрудники, роли, навыки, шаблоны задач).

**Филиальная БД** – локальное хранилище операционных данных конкретного филиала (проекты, задачи, комментарии, логи времени).

**Фрагментация** – метод хранения данных, при котором информация распределяется по нескольким узлам на основе бизнес-признака (в данном случае — `branch_id`).

**CRUD** – базовые операции с данными: Create (создание), Read (чтение), Update (обновление), Delete (удаление).

## 1.3. Технические термины

**БД** – база данных, система хранения структурированной информации.

**PostgreSQL** – реляционная СУБД, используемая в проекте для хранения данных.

**Docker** – платформа контейнеризации, применяемая для развёртывания PostgreSQL-инстансов.

**ВМ** – виртуальная машина, вычислительный ресурс.

**Схема (schema)** – логический контейнер в БД для группировки объектов (например, `global`, `branch`).

**FK (Foreign Key)** – внешний ключ; в распределённой архитектуре реализуется на уровне приложения, а не СУБД.

**API** – программный интерфейс для взаимодействия между компонентами Системы.

## 1.4. Другие термины

**HR** – сотрудник отдела кадров, ответственный за управление профилями сотрудников.

**Менеджер** – пользователь с ролью «manager», управляющий проектами и задачами в филиале.

**Сотрудник** – исполнитель задач, имеющий роль «developer», «analyst» и т.п.

**Шаблон задачи** – предопределённая структура задачи (название, описание, оценка трудозатрат, приоритет).

**Уведомление** – системное сообщение, отправляемое сотруднику при назначении задачи или изменении её статуса.

## **2. Общие положения**

### **2.1. Назначение документа**

Настоящий документ определяет полный набор требований к распределённой системе управления задачами (далее — **Система**), предназначенной для координации работы сотрудников в условиях многофилиальной структуры компании.

Подписание настоящего Технического задания Заказчиком и Исполнителем означает согласие сторон с нижеследующими условиями:

2.1.1. При реализации проекта Исполнитель обязуется выполнить работы в объёме, указанном в настоящем документе.

2.1.2. Все неоднозначности, выявленные в тексте Технического задания после его подписания, подлежат согласованию между Заказчиком и Исполнителем в письменной форме.

### **2.2. Цели создания Системы**

#### **2.2.1. С точки зрения руководства компании:**

2.2.1.1. Обеспечить единое информационное пространство для управления проектами и задачами в распределённой географической структуре.

2.2.1.2. Повысить прозрачность загрузки сотрудников и эффективность распределения рабочих задач.

2.2.1.3. Снизить операционные издержки за счёт автоматизации учёта рабочего времени и уведомлений.

2.2.1.4. Обеспечить масштабируемость ИТ-инфраструктуры при открытии новых филиалов.

#### **2.2.2. С точки зрения менеджера филиала:**

2.2.2.1. Получить инструмент для оперативного назначения задач и контроля их выполнения.

2.2.2.2. Автоматизировать процесс учёта трудозатрат сотрудников по задачам и проектам.

2.2.2.3. Упростить взаимодействие с командой через встроенные комментарии и уведомления.

#### **2.2.3. С точки зрения сотрудника:**

2.2.3.1. Получить централизованный доступ к своим задачам, срокам и проектам.

2.2.3.2. Возможность оперативно отчитываться о затраченном времени.

2.2.3.3. Получать своевременные уведомления о новых задачах и изменениях в статусах.

## **2.3. Основные функциональные возможности Системы**

### **2.3.1. Управление справочниками (глобальный уровень)**

- 2.3.1.1. Регистрация и управление сотрудниками (ФИО, email, роль, филиал, дата приёма).
- 2.3.1.2. Управление ролями пользователей (developer, manager, analyst).
- 2.3.1.3. Ведение справочника навыков и привязка их к сотрудникам с указанием уровня компетенции.
- 2.3.1.4. Создание и редактирование шаблонов задач (название, описание, приоритет, оценка часов).
- 2.3.1.5. Управление структурой подразделений внутри филиалов.

### **2.3.2. Операционная деятельность (филиальный уровень)**

- 2.3.2.1. Создание и управление проектами (название, менеджер, сроки, статус).
- 2.3.2.2. Назначение задач сотрудникам с привязкой к проекту, шаблону и сроку выполнения.
- 2.3.2.3. Добавление комментариев и файлов к задачам.
- 2.3.2.4. Учёт рабочего времени по задачам с указанием даты и описания.
- 2.3.2.5. Отправка уведомлений сотрудникам при назначении задач и изменении статусов.

### **2.3.3. Архитектурные особенности**

- 2.3.3.1. Распределённое хранение данных: глобальная БД (сотрудники, справочники) + филиальные БД (проекты, задачи).
- 2.3.3.2. Горизонтальная фрагментация данных по признаку `branch_id`.
- 2.3.3.3. Поддержка CRUD-операций через централизованное приложение с маршрутизацией запросов.
- 2.3.3.4. Валидация целостности данных на уровне приложения (проверка принадлежности сотрудника к филиалу и т.п.).

## **2.4. Использование Технического Задания**

- 2.4.1. Настоящий документ является основой для проектирования, разработки, тестирования и сдачи Системы.
- 2.4.2. Все материалы, созданные в рамках проекта (исходный код, схемы БД, скрипты развёртывания), должны соответствовать требованиям, изложенным в данном Техническом задании.
- 2.4.3. Отношения между Заказчиком и Исполнителем в части конфиденциальности регулируются отдельным соглашением о неразглашении (NDA).

## **3. Функциональные требования**

### **3.1. Диаграммы Вариантов Использования**

*Диаграммы Вариантов Использования (Use Case Diagrams) будут размещены в данном разделе на этапе проектирования пользовательского интерфейса. В текущей версии*

проекта взаимодействие с системой осуществляется через консольный интерфейс, поэтому диаграммы временно опущены.

## **3.2. Описание Вариантов Использования**

### **3.2.1. ВИ «Создать задачу» (Сценарий 1)**

#### **3.2.1.1. Описание ВИ**

Менеджер филиала создаёт новую задачу для сотрудника своего филиала через CLI-интерфейс.

#### **3.2.1.2. Предусловия**

- Менеджер и исполнитель зарегистрированы в глобальной БД.
- Оба сотрудника принадлежат одному филиалу (`branch_id`).
- В филиальной БД существует проект, управляемый менеджером.

#### **3.2.1.3. Основной поток действий**

1. Пользователь запускает CLI-скрипт и выбирает опцию «Создать задачу».
2. Система запрашивает ID менеджера, ID исполнителя, ID шаблона задачи, срок выполнения и название.
3. Система проверяет существование сотрудников в `global.employees` и их принадлежность к одному филиалу.
4. Система определяет филиал по `branch_id` исполнителя.
5. Система подключается к соответствующей филиальной БД (192.168.64.5 или 192.168.64.6).
6. Система создаёт запись в `branch.tasks`.
7. Система создаёт уведомление в `branch.notifications` для исполнителя.
8. Система выводит сообщение об успешном создании задачи.

#### **3.2.1.4. Альтернативные потоки**

- Если исполнитель не найден → вывод ошибки.
- Если менеджер и исполнитель из разных филиалов → операция отклоняется.
- Если проект не найден → операция отклоняется.

#### **3.2.1.5. Бизнес-правила**

- Задача может быть назначена только активному сотруднику.
- Уведомление создаётся автоматически при создании задачи.

### **3.2.2. ВИ «Отчитаться по времени» (Сценарий 2)**

#### **3.2.2.1. Описание ВИ**

Сотрудник добавляет запись о затраченном времени на выполнение задачи.

### 3.2.2.2. Предусловия

- Сотрудник зарегистрирован в `global.employees`.
- Задача существует в филиальной БД и назначена этому сотруднику.

### 3.2.2.3. Основной поток действий

1. Пользователь выбирает опцию «Отчёт по времени».
2. Вводит свой ID, ID задачи, количество часов, дату и описание.
3. Система проверяет принадлежность задачи сотруднику.
4. Система подключается к филиальной БД по `branch_id` сотрудника.
5. Система добавляет запись в `branch.time_logs`.
6. Подтверждает успешное добавление.

### 3.2.2.4. Альтернативные потоки

- Если задача не назначена сотруднику → ошибка.
- Если филиал недоступен → ошибка подключения.

### 3.2.2.5. Бизнес-правила

- Одна запись = одна дата + один сотрудник + одна задача.
- Часы могут быть дробными (например, 2.5).

## 3.2.3. ВИ «Добавить сотрудника» (Сценарий 3)

### 3.2.3.1. Описание ВИ

HR-менеджер регистрирует нового сотрудника в глобальной БД.

### 3.2.3.2. Предусловия

- Нет (операция доступна всегда через CLI).

### 3.2.3.3. Основной поток действий

1. Пользователь выбирает опцию «Добавить сотрудника».
2. Вводит ФИО, email, филиал, роль, навык и уровень.
3. Система подключается к глобальной БД (192.168.64.4).
4. Создаёт запись в `global.employees`.
5. Добавляет связь в `global.employee_skills`.
6. Привязывает сотрудника к отделу в `global.employee_departments`.
7. Подтверждает успешное добавление.

### 3.2.3.4. Альтернативные потоки

- Если email уже существует → ошибка (уникальность).
- Если филиал не поддерживается (не 1 или 2) → ошибка.

### 3.2.3.5. Бизнес-правила

- Email должен быть уникальным.
- Сотрудник сразу становится доступным для назначения задач в своём филиале.

### 3.2.4. ВИ «Просмотреть свои задачи»

#### 3.2.4.1. Описание ВИ

Сотрудник просматривает список всех задач, назначенных ему.


#### 3.2.4.2. Предусловия

- Сотрудник существует в `global.employees`.

#### 3.2.4.3. Основной поток действий

1. Пользователь вводит свой ID.
2. Система определяет `branch_id`.
3. Подключается к филиальной БД.
4. Выполняет JOIN между `branch.tasks` и `branch.projects`.
5. Выводит список задач с названием, статусом, сроком и проектом.

#### 3.2.4.4. Альтернативные потоки

- Если задач нет → выводится сообщение « У вас нет задач».

### 3.2.5. ВИ «Удалить задачу»

#### 3.2.5.1. Описание ВИ

Администратор удаляет задачу и связанные с ней данные.

#### 3.2.5.2. Предусловия

- Задача существует в одной из филиальных БД.

#### 3.2.5.3. Основной поток действий

1. Пользователь вводит ID задачи.
2. Система последовательно проверяет наличие задачи в `branch_1` и `branch_2`.
3. При нахождении:
  - Удаляет записи из `branch.comments`, `branch.time_logs`, `branch.attachments`, `branch.notifications`.
  - Удаляет саму задачу из `branch.tasks`.
- 4.

5. Подтверждает удаление.

### 3.3. Дополнительные функциональные требования

#### 3.3.1. Распределённое хранение данных

Система хранит данные в трёх независимых PostgreSQL-инстансах:

- Глобальная БД (192.168.64.4) — справочники.
- Филиал 1 (192.168.64.5) — операционные данные филиала Moscow.
- Филиал 2 (192.168.64.6) — операционные данные филиала Berlin.

#### 3.3.2. Горизонтальная фрагментация

Данные фрагментированы по признаку `branch_id`. Записи филиала 1 хранятся только в БД 192.168.64.5, филиала 2 — только в 192.168.64.6.

#### 3.3.3. Целостность данных

Связи между глобальной и филиальными БД (например, `assignee_id` → `global.employees.id`) обеспечиваются на уровне приложения, а не СУБД.

#### 3.3.4. Логирование операций

Все операции CRUD логируются в консоль (`stdout`). В продакшене может быть добавлен файловый или централизованный логгер.

#### 3.3.5. Безопасность подключения

Подключение к БД осуществляется по приватным IP-адресам. Пароли хранятся в коде временно (в продакшене — в переменных окружения или секретах).

# 5. Модель данных







7.1.3. Файлы `docker-compose.yaml` для каждой из трёх ВМ.

7.1.4. Скрипты наполнения данных (`seed_all_tables.py`) и демонстрации CRUD (`task_manager_cli.py`).

7.1.5. Краткую инструкцию по развёртыванию и запуску системы.

7.2. Приемо-сдаточные испытания включают:

7.2.1. Проверку работоспособности всех трёх PostgreSQL-инстансов.

7.2.2. Демонстрацию горизонтальной фрагментации: данные филиала 1 отсутствуют в БД филиала 2 и наоборот.

7.2.3. Выполнение всех пяти сценариев из раздела 3.2 (создание задачи, учёт времени, добавление сотрудника и т.д.).

7.2.4. Проверку целостности данных: невозможность назначить задачу сотруднику из другого филиала.

7.3. Результаты испытаний фиксируются в **протоколе сдачи**, который подписывается Заказчиком и Исполнителем.

7.4. После успешной сдачи проект считается **принятым** и готовым к использованию в качестве учебного/демонстрационного прототипа распределённой системы.