

1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Распределённая система управления задачами для многофилиальной компании

1. Общее описание

Компания «GlobalTech» имеет несколько филиалов по всему миру. Для эффективного управления проектами и задачами сотрудникам требуется единая, но распределённая система, которая позволяет:

- Централизованно управлять *сотрудниками, ролями, навыками и структурой компании*.
- Локально (в каждом филиале) управлять *проектами, задачами, временными затратами и комментариями*.
- Обеспечить целостность данных при горизонтальной фрагментации по филиалам.
- Поддерживать CRUD-операции в реальном времени с возможностью масштабирования.

2. Цели системы

- Упростить распределение задач между сотрудниками разных филиалов.
- Автоматизировать учёт рабочего времени и отслеживание прогресса по задачам.
- Обеспечить доступ к глобальным справочникам (роли, навыки, шаблоны задач) из любого филиала.
- Гарантировать согласованность данных при работе в распределённой среде.

3. Функциональные требования

3.1. Управление сотрудниками (глобальная часть)

- Регистрация и редактирование сотрудников (ФИО, email, роль, филиал, дата найма).
- Назначение ролей: manager, developer, analyst, qa, designer.
- Привязка навыков к сотрудникам с уровнем компетенции (`junior`, `middle`, `senior`, `Team Lead`).
- Управление подразделениями внутри филиалов (например, "Отдел разработки", "Отдел тестирования").
- Возможность назначать сотрудника руководителем подразделения.

3.2. Управление филиалами (глобальная часть)

- Создание/редактирование филиалов с указанием:
 - Названия
 - Локации
 - Часового пояса
- Ведение локальных праздников для каждого филиала.

3.3. Управление задачами (филиальная часть)

- Создание задач с привязкой к проекту, исполнителю, автору (репортеру), шаблону.
- Установка статуса задачи: ``todo``, ``in_progress``, ``review``, ``done``.
- Установка приоритета: ``low``, ``medium``, ``high``, ``critical``.
- Установка срока выполнения (``due_date``).
- Прикрепление файлов к задачам.
- Добавление комментариев к задачам от любых сотрудников.
- Учёт временных затрат: сколько часов сотрудник потратил на задачу в конкретный день.

3.4. Управление проектами (филиальная часть)

- Создание проектов с менеджером, описанием, датами начала/окончания.
- Статус проекта: ``active``, ``paused``, ``completed``.

3.5. Уведомления (филиальная часть)

- Отправка уведомлений сотрудникам при:
 - Назначении задачи
 - Комментариях к задаче
 - Изменении статуса задачи
- Возможность пометки уведомления как прочитанного.

3.6. Шаблоны задач (глобальная часть)

- Создание типовых задач (шаблонов) с предустановленным описанием, оценкой трудозатрат, приоритетом.
- Использование шаблонов при создании новых задач.

4. Архитектурные требования

4.1. Распределённая архитектура

- **Глобальная БД** (``global``) — хранит общие данные:
 - ``employees``, ``roles``, ``branches``, ``skills``, ``employee_skills``, ``departments``, ``employee_departments``, ``holidays``, ``task_templates``
- **Филиальные БД** (``branch_1``, ``branch_2``, ...) — хранят локальные данные:
 - ``tasks``, ``projects``, ``comments``, ``attachments``, ``time_logs``, ``notifications``

4.2. Фрагментация

- Горизонтальная фрагментация по ``branch_id``.
- Все филиальные таблицы содержат только данные, относящиеся к их ``branch_id``.
- Ключи ``assignee_id``, ``reporter_id``, ``manager_id``, ``uploaded_by``, ``author_id``, ``employee_id`` ссылаются на ``global.employees.id``.

4.3. Интеграция и согласованность

- При создании задачи в филиале система проверяет существование `'assignee_id'` через глобальный сервис.
- При удалении сотрудника — не удаляется, а переводится в состояние `'is_active = false'`.
- Все операции чтения/записи должны быть атомарными и согласованными.
- Использование централизованного генератора ID (UUID или sequence).

5. Нефункциональные требования

- Масштабируемость: возможность добавления новых филиалов без перестройки всей системы.
- Производительность: время отклика на CRUD-операции — менее 500 мс.

6. Ограничения

- Задачи и проекты не могут быть перемещены между филиалами.
- Навыки и роли — глобальные, не зависят от филиала.
- Уведомления отправляются только сотрудникам текущего филиала.

7. Примеры использования

Сценарий 1: Менеджер создаёт задачу

> Менеджер филиала `'Moscow'` создаёт задачу «Разработать API для оплаты», назначает её сотруднику `'Ivan Petrov'` (ID 101, филиал `'Moscow'`), выбирает шаблон «Backend task», устанавливает срок до 2025-11-30. Система автоматически создаёт запись в `'branch.tasks'` и отправляет уведомление сотруднику.

Сценарий 2: Сотрудник отчитывается по времени

> Сотрудник `'Anna Sidorova'` (ID 105) заходит в задачу «Разработать API для оплаты» и добавляет запись в `'branch.time_logs'`: 3 часа, дата 2025-10-22, описание «Реализация контроллера». Система обновляет общую статистику по задаче.

Сценарий 3: HR добавляет нового сотрудника

> HR-менеджер добавляет нового сотрудника `'Alexei Ivanov'` в филиал `'Berlin'`, назначает роль `'developer'`, привязывает навык `'Python'` на уровне `'intermediate'`. Сотрудник сразу доступен для назначения задач в филиале `'Berlin'`.

The diagram illustrates the database schema for a task management system, featuring 14 tables and their relationships:

- global.employees**: Attributes include `id` (PK), `full_name`, `email`, `role_id` (FK), `branch_id` (FK), `hired_at`, and `is_active`.
- global.roles**: Attributes include `id` (PK), `name`, and `description`.
- global.branches**: Attributes include `id` (PK), `name`, `location`, and `timezone`.
- global.task_templates**: Attributes include `id` (PK), `title`, `description`, `estimated_hours`, and `priority_level`.
- global.skills**: Attributes include `id` (PK) and `name`.
- global.employee_skills**: Attributes include `employee_id` (FK), `skill_id` (FK), and `proficiency_level`.
- branch.comments**: Attributes include `task_id` (FK), `author_id` (FK), `content`, and `created_at`.
- branch.attachments**: Attributes include `task_id` (FK), `file_name`, `file_path`, `uploaded_by` (FK), and `uploaded_at`.
- branch.projects**: Attributes include `id` (PK), `name`, `description`, `manager_id` (FK), `start_date`, `end_date`, and `status`.
- branch.notifications**: Attributes include `employee_id` (FK), `message`, `is_read`, `created_at`, and `related_task_id` (FK).
- global.departments**: Attributes include `id` (PK), `name`, and `branch_id` (FK).
- global.employee_departments**: Attributes include `employee_id` (FK), `department_id` (FK), and `is_head`.
- branch.time_logs**: Attributes include `task_id` (FK), `employee_id` (FK), `hours_spent`, `date_worked`, and `description`.
- global.holidays**: Attributes include `id` (PK), `branch_id` (FK), `name`, `date_from`, and `date_to`.

Relationships are defined by lines connecting attributes across tables, indicating foreign key constraints and cardinalities.