**Техническое Задание: Система Управления Корпоративными Финансами (CFMS)**

**1. Общее Описание**

Система предназначена для мониторинга и контроля корпоративных финансовых операций, включая управление бюджетами, расходами, отчетностью и пользователями. Система будет обеспечивать централизованное хранилище финансовых данных и инструменты для их анализа и управления.

**2. Функциональные Требования**

**2.1 Управление Бюджетом**

* **Создание бюджета**
  + **Методы:**
    - POST /budgets
  + **Описание:** Создает новый бюджет с заданными параметрами и сохраняет его в базе данных.
  + **Параметры запроса:**
    - Budget: объект, содержащий:
      * id: уникальный идентификатор (автоматически генерируется)
      * name: название бюджета
      * amount: сумма бюджета
      * categories: список категорий
      * startDate: дата начала
      * endDate: дата окончания
* **Обновление бюджета**
  + **Методы:**
    - PUT /budgets/{budgetId}
  + **Описание:** Обновляет данные существующего бюджета в базе данных.
  + **Параметры запроса:**
    - Budget: объект с обновленными данными бюджета.
* **Просмотр бюджета**
  + **Методы:**
    - GET /budgets/{budgetId}
  + **Описание:** Извлекает информацию о бюджете по его идентификатору.
  + **Параметры запроса:**
    - budgetId: уникальный идентификатор бюджета.
* **Удаление бюджета**
  + **Методы:**
    - DELETE /budgets/{budgetId}
  + **Описание:** Удаляет бюджет из базы данных по его идентификатору.
  + **Параметры запроса:**
    - budgetId: уникальный идентификатор бюджета.
* **Отслеживание выполнения бюджета**
  + **Методы:**
    - GET /budgets/{budgetId}/performance
  + **Описание:** Подсчитывает и анализирует фактические расходы по отношению к запланированному бюджету.
  + **Параметры запроса:**
    - budgetId: уникальный идентификатор бюджета.

**2.2 Управление Расходами**

* **Регистрация расхода**
  + **Методы:**
    - POST /expenses
  + **Описание:** Создает новый расход и сохраняет его в базе данных.
  + **Параметры запроса:**
    - Expense: объект, содержащий:
      * id: уникальный идентификатор (автоматически генерируется)
      * date: дата расхода
      * amount: сумма расхода
      * category: категория расхода
      * description: описание расхода
      * budgetId: идентификатор бюджета, к которому относится расход
* **Обновление расхода**
  + **Методы:**
    - PUT /expenses/{expenseId}
  + **Описание:** Обновляет данные существующего расхода в базе данных.
  + **Параметры запроса:**
    - Expense: объект с обновленными данными расхода.
* **Просмотр расхода**
  + **Методы:**
    - GET /expenses/{expenseId}
  + **Описание:** Извлекает информацию о расходе по его идентификатору.
  + **Параметры запроса:**
    - expenseId: уникальный идентификатор расхода.
* **Удаление расхода**
  + **Методы:**
    - DELETE /expenses/{expenseId}
  + **Описание:** Удаляет расход из базы данных по его идентификатору.
  + **Параметры запроса:**
    - expenseId: уникальный идентификатор расхода.
* **Классификация расхода**
  + **Методы:**
    - PUT /expenses/{expenseId}/category
  + **Описание:** Обновляет категорию расхода в базе данных.
  + **Параметры запроса:**
    - expenseId: уникальный идентификатор расхода.
    - category: новая категория расхода.

**2.3 Финансовая Отчетность**

* **Создание финансового отчета**
  + **Методы:**

 **generateReport(ReportType reportType, LocalDate startDate, LocalDate endDate)**

* Генерация отчета за указанный период времени.

 **getReportById(Long reportId)**

* Получение отчета по его уникальному идентификатору.

 **getAllReports()**

* Получение списка всех отчетов.

 **getReportsByType(ReportType reportType)**

* Получение списка отчетов определенного типа (ежедневные, ежемесячные и т.д.).

 **deleteReport(Long reportId)**

* Удаление отчета по его идентификатору.

 **updateReport(Long reportId, Report updatedReport)**

* Обновление существующего отчета.

 **exportReport(Long reportId, String format)**

* Экспорт отчета в указанный формат (PDF, Excel, и т.д.).

 **filterReports(LocalDate startDate, LocalDate endDate, ReportType reportType)**

* Фильтрация отчетов по указанному периоду и типу отчета.

 **archiveReport(Long reportId)**

* Архивирование отчета.

 **generateSummaryReport(ReportType reportType, LocalDate startDate, LocalDate endDate)**

* Генерация сводного отчета за указанный период времени.

 **scheduleReportGeneration(ReportType reportType, LocalDate startDate, LocalDate endDate, LocalDateTime scheduleTime)**

* Планирование автоматической генерации отчета на определенное время.

 **getLatestReports(int limit)**

* Получение последних n отчетов.

**2.4 Управление Пользователями и Доступом**

* **Регистрация пользователя**
  + **Методы:**
    - POST /users
  + **Описание:** Создает нового пользователя и сохраняет его в базе данных.
  + **Параметры запроса:**
    - User: объект, содержащий:
      * id: уникальный идентификатор (автоматически генерируется)
      * username: имя пользователя
      * password: пароль (в зашифрованном виде)
      * roles: список ролей пользователя
* **Обновление пользователя**
  + **Методы:**
    - PUT /users/{userId}
  + **Описание:** Обновляет данные существующего пользователя в базе данных.
  + **Параметры запроса:**
    - User: объект с обновленными данными пользователя.
* **Просмотр пользователя**
  + **Методы:**
    - GET /users/{userId}
  + **Описание:** Извлекает информацию о пользователе по его идентификатору.
  + **Параметры запроса:**
    - userId: уникальный идентификатор пользователя.
* **Удаление пользователя**
  + **Методы:**
    - DELETE /users/{userId}
  + **Описание:** Удаляет пользователя из базы данных по его идентификатору.
  + **Параметры запроса:**
    - userId: уникальный идентификатор пользователя.
* **Аудит действий пользователя**
  + **Методы:**
    - GET /users/{userId}/activity
  + **Описание:** Извлекает журнал действий пользователя для анализа.
  + **Параметры запроса:**
    - userId: уникальный идентификатор пользователя.

**3. Архитектура**

* **Backend:**
  + **Spring Boot**: для создания RESTful API.
  + **Spring Security**: для управления доступом и аутентификацией пользователей.
  + **Hibernate**: для работы с базой данных (PostgreSQL).
  + **JPA**: для создания и управления сущностями в базе данных.
* **База данных:**
  + **PostgreSQL**: для хранения данных о бюджетах, расходах, отчетах и пользователях.

**4. Документация**

* **API Документация:**
  + Подробное описание всех эндпоинтов API, включая примеры запросов и ответов. Используйте Swagger или OpenAPI для автоматической генерации документации.
* **Техническая документация:**
  + Руководство по установке и настройке системы.
  + Инструкции по использованию основных функций.

В контексте системы управления корпоративными финансами (CFMS), бюджет представляет собой набор ресурсов, выделенных на определенный период времени для конкретных категорий расходов. Бюджет используется для планирования и контроля финансовых потоков, чтобы убедиться, что затраты не превышают установленные лимиты и соответствуют стратегическим целям компании.

**Описание Базовых Моделей**

**1. Бюджет (Budget)**

**Описание:** Модель "Бюджет" представляет собой планирование финансовых ресурсов на определенный период. Бюджеты могут быть связаны с различными категориями расходов и могут использоваться для контроля за выполнением финансовых планов.

**Поля:**

* id: уникальный идентификатор бюджета (автоматически генерируется).
* name: название бюджета (например, "Маркетинговый бюджет на Q1").
* amount: общая сумма бюджета, выделенная на указанный период.
* categories: список категорий, к которым относится данный бюджет (например, "Маркетинг", "Операции").
* startDate: дата начала действия бюджета.
* endDate: дата окончания действия бюджета.

**Функции:**

* Создание нового бюджета.
* Обновление информации о бюджете.
* Просмотр деталей бюджета.
* Удаление бюджета.
* Отслеживание выполнения бюджета и сравнение фактических расходов с запланированными.

**2. Расход (Expense)**

**Описание:** Модель "Расход" представляет собой финансовое обязательство компании, зарегистрированное как расход. Расходы могут быть привязаны к определенным бюджетам и категориям для более детального контроля.

**Поля:**

* id: уникальный идентификатор расхода (автоматически генерируется).
* date: дата, когда был зафиксирован расход.
* amount: сумма расхода.
* category: категория, к которой относится расход (например, "Путевые расходы", "Реклама").
* description: описание расхода (например, "Закупка рекламных материалов").
* budgetId: идентификатор бюджета, к которому относится этот расход.

**Функции:**

* Регистрация нового расхода.
* Обновление данных расхода.
* Просмотр деталей расхода.
* Удаление расхода.
* Изменение категории расхода.

**3. Финансовый Отчет (Report)**

**Описание:** Модель "Финансовый отчет" используется для создания и хранения отчетов о финансовом состоянии компании на основе данных о бюджетах и расходах. Отчеты могут быть использованы для анализа эффективности бюджета и расходов.

**Поля:**

* id: уникальный идентификатор отчета (автоматически генерируется).
* startDate: дата начала отчетного периода.
* endDate: дата окончания отчетного периода.
* budgetId: идентификатор бюджета, для которого создается отчет (необязательно).
* content: содержимое отчета (например, текстовое описание или таблицы с данными).

**Функции:**

* Создание нового финансового отчета на основе заданных критериев.
* Просмотр созданного отчета.
* Извлечение информации о финансовом состоянии за указанный период.

**Поля Отчета (Report)**

Отчет (Report) в системе управления корпоративными финансами (CFMS) должен содержать информацию, которая позволяет анализировать финансовое состояние компании, сравнивать планируемые бюджеты с фактическими расходами, а также генерировать различные виды отчетности для принятия управленческих решений.

**Основные поля Отчета:**

1. **id (Long)**: Уникальный идентификатор отчета.
2. **reportDate (LocalDateTime)**: Дата и время формирования отчета.
3. **reportType (ReportType)**: Тип отчета (например, "Monthly", "Quarterly", "Annual", "Custom").
4. **startDate (LocalDateTime)**: Дата начала отчетного периода.
5. **endDate (LocalDateTime)**: Дата окончания отчетного периода.
6. **budgetSummary (Map<BudgetCategory, Double>)**: Сводка по бюджету — сумма, запланированная для каждой категории бюджета в отчетный период.
7. **expenseSummary (Map<BudgetCategory, Double>)**: Сводка по фактическим расходам — сумма расходов по каждой категории в отчетный период.
8. **varianceSummary (Map<BudgetCategory, Double>)**: Отклонение (Variance) — разница между запланированным бюджетом и фактическими расходами по каждой категории.
9. **totalBudget (Double)**: Общая сумма бюджета на отчетный период.
10. **totalExpenses (Double)**: Общая сумма фактических расходов на отчетный период.
11. **totalVariance (Double)**: Общее отклонение — разница между общим бюджетом и общими расходами.
12. **createdBy (String)**: Имя пользователя или идентификатор, создавшего отчет.

**Что должен делать Отчет?**

1. **Формирование сводки по бюджету**:
   * Отчет должен суммировать запланированные суммы по каждой категории бюджета за отчетный период.
   * Суммирование по категориям позволяет видеть общие финансовые цели компании на отчетный период.
2. **Формирование сводки по фактическим расходам**:
   * Отчет должен агрегировать фактические расходы по каждой категории за отчетный период.
   * Сравнение фактических расходов с бюджетом помогает выявить области, где расходы превысили или не достигли запланированных значений.
3. **Расчет отклонений (Variance)**:
   * Для каждой категории бюджета отчет должен вычислять разницу между запланированными и фактическими расходами.
   * Это позволяет управлению компании понимать, насколько эффективно используются ресурсы.
4. **Генерация отчета по разным периодам**:
   * Отчет может генерироваться за различные временные периоды (месяц, квартал, год) в зависимости от нужд компании.
   * Пользователи могут выбирать даты начала и окончания отчетного периода.
5. **Сводка по общим показателям**:
   * Отчет должен отображать общую сумму бюджета, общую сумму расходов и общее отклонение для всего отчетного периода.
   * Это дает общее представление о финансовом состоянии компании.
6. **Сохранение и экспорт**:
   * Отчет должен быть сохранен в базе данных для дальнейшего использования.
   * Возможность экспорта отчета в различные форматы (например, PDF, Excel) для внешней отчетности или архивирования.

**На основе чего формируется Отчет?**

1. **Бюджетные данные**:
   * Отчет использует данные из таблицы бюджетов (Budget), чтобы извлечь запланированные суммы по каждой категории на отчетный период.
2. **Данные о расходах**:
   * Отчет извлекает фактические данные о расходах из таблицы расходов (Expense), чтобы получить информацию о фактических затратах за тот же период.
3. **Категории**:
   * Категории бюджета и расходов, которые используются в расчетах, извлекаются из соответствующего перечисления (enum), чтобы обеспечить согласованность и точность данных.

**4. Пользователь (User)**

**Описание:** Модель "Пользователь" представляет собой запись о пользователе системы, который имеет доступ к функционалу CFMS. Управление пользователями включает в себя создание, обновление, удаление и управление правами доступа.

**Поля:**

* id: уникальный идентификатор пользователя (автоматически генерируется).
* username: имя пользователя для входа в систему.
* password: пароль пользователя (в зашифрованном виде).
* roles: список ролей пользователя, определяющих его права доступа (например, "Администратор", "Менеджер").

**Функции:**

* Регистрация нового пользователя.
* Обновление информации о пользователе.
* Просмотр данных о пользователе.
* Удаление пользователя.
* Аудит действий пользователя.

Эти модели являются основными компонентами системы и обеспечивают управление ключевыми аспектами корпоративных финансов. Вы можете расширить их функциональность и добавить дополнительные поля в зависимости от потребностей вашего проекта.

Для системы управления корпоративными финансами (CFMS) можно предложить следующую архитектуру проекта. Эта архитектура включает в себя компоненты серверной части, базу данных и взаимодействие между ними.

**Архитектура Проекта CFMS**

**1. Обзор Архитектуры**

Проект будет построен на основе архитектуры многослойного приложения с использованием Java Spring. Основные слои включают:

* **Презентационный слой (Controller)**
* **Сервисный слой (Service)**
* **Слой доступа к данным (Repository)**
* **Модели данных (Entities)**
* **Конфигурации и утилиты**

База данных будет использовать PostgreSQL, а для управления зависимостями и сборки проекта будет использоваться Maven.

**2. Слои Архитектуры**

1. **Презентационный Слой (Controller)**
   * **Цель:** Обработка входящих HTTP-запросов и предоставление ответов клиенту.
   * **Компоненты:**
     + Контроллеры (@RestController): Управляют запросами и вызовами соответствующих сервисов.
     + Примеры:
       - BudgetController
       - ExpenseController
       - ReportController
       - UserController
   * **Примеры Методов:**
     + GET /budgets – Получить список всех бюджетов.
     + POST /budgets – Создать новый бюджет.
     + PUT /budgets/{id} – Обновить существующий бюджет.
     + DELETE /budgets/{id} – Удалить бюджет.
2. **Сервисный Слой (Service)**
   * **Цель:** Реализация бизнес-логики и взаимодействие между контроллерами и репозиториями.
   * **Компоненты:**
     + Сервисы (@Service): Обрабатывают бизнес-логику и вызывают методы репозиториев.
     + Примеры:
       - BudgetService
       - ExpenseService
       - ReportService
       - UserService
   * **Примеры Методов:**
     + createBudget(Budget budget) – Создать новый бюджет.
     + updateBudget(Long id, Budget budget) – Обновить бюджет по идентификатору.
     + deleteBudget(Long id) – Удалить бюджет по идентификатору.
     + generateReport(ReportCriteria criteria) – Создать финансовый отчет.
3. **Слой Доступа к Данным (Repository)**
   * **Цель:** Управление доступом к базе данных и выполнение операций CRUD.
   * **Компоненты:**
     + Репозитории (@Repository): Используют Spring Data JPA для доступа к базе данных.
     + Примеры:
       - BudgetRepository
       - ExpenseRepository
       - ReportRepository
       - UserRepository
   * **Примеры Методов:**
     + findById(Long id) – Найти запись по идентификатору.
     + save(Budget budget) – Сохранить новый или обновленный бюджет.
     + deleteById(Long id) – Удалить запись по идентификатору.
4. **Модели Данных (Entities)**
   * **Цель:** Определение структуры данных, хранимых в базе данных.
   * **Компоненты:**
     + Сущности (@Entity): Представляют таблицы в базе данных.
     + Примеры:
       - Budget
       - Expense
       - Report
       - User
   * **Примеры Полей:**
     + id: уникальный идентификатор.
     + name: название бюджета или расхода.
     + amount: сумма.
     + date: дата расхода.
     + category: категория.
5. **Конфигурации и Утилиты**
   * **Цель:** Настройка компонентов и вспомогательные функции.
   * **Компоненты:**
     + Конфигурационные файлы (application.properties или application.yml): Для настройки подключения к базе данных и других параметров.
     + Утилиты: Классы для обработки общих задач, таких как валидация данных, преобразование форматов и т.д.
   * **Примеры:**
     + DatabaseConfig: Конфигурация для подключения к PostgreSQL.
     + SecurityConfig: Настройки безопасности и аутентификации (Spring Security).

Вот компактное описание всех возможных запросов (эндпоинтов) для контроллеров и соответствующих функций, которые вам нужно реализовать в системе управления корпоративными финансами (CFMS):

**1. Бюджеты (Budgets)**

* **Получить все бюджеты**
  + GET /budgets
  + **Функция:** Получение списка всех бюджетов.
* **Создать новый бюджет**
  + POST /budgets
  + **Функция:** Создание нового бюджета.
* **Обновить существующий бюджет**
  + PUT /budgets/{budgetId}
  + **Функция:** Обновление данных бюджета по идентификатору.
* **Удалить бюджет**
  + DELETE /budgets/{budgetId}
  + **Функция:** Удаление бюджета по идентификатору.
* **Получить информацию о бюджете**
  + GET /budgets/{budgetId}
  + **Функция:** Получение информации о бюджете по идентификатору.
* **Получить выполнение бюджета**
  + GET /budgets/{budgetId}/performance
  + **Функция:** Получение отчета о выполнении бюджета.

**2. Расходы (Expenses)**

* **Получить все расходы**
  + GET /expenses
  + **Функция:** Получение списка всех расходов.
* **Создать новый расход**
  + POST /expenses
  + **Функция:** Создание нового расхода.
* **Обновить существующий расход**
  + PUT /expenses/{expenseId}
  + **Функция:** Обновление данных расхода по идентификатору.
* **Удалить расход**
  + DELETE /expenses/{expenseId}
  + **Функция:** Удаление расхода по идентификатору.
* **Получить информацию о расходе**
  + GET /expenses/{expenseId}
  + **Функция:** Получение информации о расходе по идентификатору.
* **Обновить категорию расхода**
  + PUT /expenses/{expenseId}/category
  + **Функция:** Обновление категории расхода по идентификатору.

**3. Финансовые отчеты (Reports)**

* **Создать финансовый отчет**
  + POST /reports
  + **Функция:** Создание нового финансового отчета на основе заданных критериев.
* **Получить информацию о отчете**
  + GET /reports/{reportId}
  + **Функция:** Получение информации о финансовом отчете по идентификатору.

**4. Пользователи (Users)**

* **Получить всех пользователей**
  + GET /users
  + **Функция:** Получение списка всех пользователей.
* **Создать нового пользователя**
  + POST /users
  + **Функция:** Создание нового пользователя.
* **Обновить существующего пользователя**
  + PUT /users/{userId}
  + **Функция:** Обновление данных пользователя по идентификатору.
* **Удалить пользователя**
  + DELETE /users/{userId}
  + **Функция:** Удаление пользователя по идентификатору.
* **Получить информацию о пользователе**
  + GET /users/{userId}
  + **Функция:** Получение информации о пользователе по идентификатору.
* **Получить активность пользователя**
  + GET /users/{userId}/activity
  + **Функция:** Получение журнала активности пользователя.

**Итоговые Функции для Реализации**

1. **Бюджеты**
   * Создание, чтение, обновление и удаление бюджетов.
   * Отслеживание выполнения бюджета.
2. **Расходы**
   * Создание, чтение, обновление и удаление расходов.
   * Изменение категорий расходов.
3. **Финансовые отчеты**
   * Создание и получение отчетов о финансовом состоянии.
4. **Пользователи**
   * Управление пользователями: создание, обновление, удаление и просмотр.
   * Получение журнала активности пользователей.

Эти эндпоинты и функции обеспечивают основные операции для управления корпоративными финансами в вашей системе. Если нужны дополнительные детали или изменения, сообщите мне!