Тестовое задание "Регистр счетов в банке"

Last Edited Time	@July 2, 2021 1:42 PM

Описание приложения

Создать систему учета банковских счетов пользователей.

Функциональность

- **Создание пользователя** с указанием тегов.

  Для создания пользователя нужно заполнить поля: ФИО, идентификационный номер, теги.

  Ограничения: нельзя создать двух пользователей с одинаковым идентификационным номером.

  Сделать максимально просто. Пример: отдельный эндпоинт, rake-task или часть "меню" cli.

- **Открытие счета для пользователя** по идентификационному номеру пользователя.

  Для открытия счета нужно заполнить поля: валюта (по стандарту ISO 4217), идентификационный номер пользователя. После открытия счета система указывает уникальный номер нового счета.

  Ограничения: пользователь не может создать два счета с одной и той же валютой, сумма должна быть неотрицательной.

- **Пополнение счета** по идентификационному номеру пользователя и валюте на определенную сумму. Если у получателя нет счета в данной валюте, необходимо создать и провести операцию.

- **Перевод между счетами** по идентификационному номеру пользователя-отправителя, валюте и идентификационному номеру получателя. Если у получателя нет счета в данной валюте, необходимо создать и провести операцию.
- **Отчет "О сумме пополнений за период времени по-валютно"** с возможностью фильтрации по пользователям.
- **Отчет "Средняя, максимальная и минимальная сумма переводов по тегам пользователей за период времени"** с возможностью фильтрации по тегам.
- **Отчет "Сумма всех счетов на текущий момент времени повалютно"** с фильтрацией по тегам пользователей.

Требования

- Реализовать проект на языке ruby.
- Для реализации можно использовать:
  - любой веб-фреймворк в режиме json api (пример: rails, sinatra, hanami-api, roda);
  - или консольное приложение без веб-интерфеса.
- Для работы с проектом, зависимостями и gems использовать bundler.
- Написать тесты с использованием rspec на функционал "Перевод между счетами" и отчет "о сумме пополнений за период времени по-валютно".
- Ограничение по времени: 2 недели с момента получения тестового задания.
- Можно добавлять ограничения на функциональность по своему усмотрению, но они должны быть логичны и причины описаны в Readme-файле.
- Разрешается добавлять дополнительные сущности, индексы, связи, классы, модули и ограничения и т.п. по мере необходимости. В Readme-файле указать причины добавления.

- Предоставить выполненное задание в виде ссылки на открытый репозиторий на github. Все замечания, идеи для проверяющего просьба поместить в Readme-файл.

Дополнительно будет учитываться

- дополнительное покрытие тестами
- документация о том, как разворачивать приложение и работать с приложением
- дополнительный функционал, что все отчеты могут быть выведены на экран или в csv файл на диске
- docker-контейнер или деплой проекта