Сводка

за 1-2 неделю

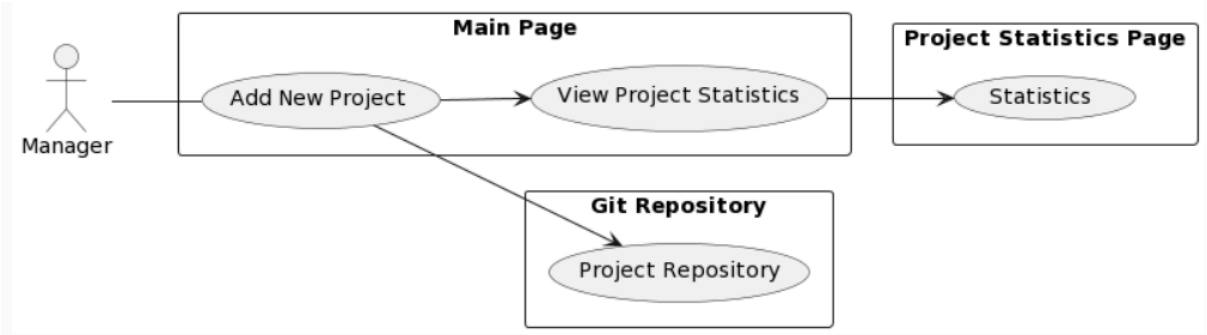
User Stories:

1. Как клиент я хочу получить репорт в ~~csv,~~ json файлы (Must).
2. Возможность поднять сервис анализа локально (Must - self-hosted services).
3. Инструкции для поднятия сервиса (Must - documentation/README).
4. Поддержка анализа Git репозиторий (Must).
6. Работа с уникальными строками кода (Must).
7. Учет частоты комитов и их содержания (Must).
8. Визуализация результатов анализа (Should - web or UI/UX).
9. Поддержка разных форматов репозиториев public\private (Should).
10. Фильтрация и агрегация данных по разработчикам (Could - web or UI/UX).

Остальные требования описаны тут.

Simple use case diagram:

Диаграмма доступна тут



Анализ Решений

Близкие решения:

- Code Quality Report Analyzer **(Good)**. It has a frontend and two servers in Python and Java. It provides options to analyze the code base.
- GitLab Analytics **(Good)**. It gives analytics on commits, wikis, issues, comments, merge-requests.

Альтернативные решения:

- GitLab Prometheus solution **(Good)** has tons of metrics on issues, time and bla-bla.
- Extract GitLab Merge Metrics **(Good)** on Python. It extracts branch name, merge commit hash, merge timestamp, number of added lines, number of removed lines, and number of touched files for each merged merge request.
- Radon (Сомнительно) on Python. It calculates several code metrics (cyclomatic complexity, halstead metrics, maintainability index, and basic ones [loc, sloc]).

- CK (Hmmm) on Java. It calculates many metrics.
- DesigniteJava (Hmmm) on Java. It calculates many metrics.

Остальные решения описаны тут.

Метрики

Ниже представлены лист метрик, которые мы поделили по цветам:

- зеленый показывает, что метрика полезна,
- желтый - удовлетворительна,
- оранжевый - не совсем удовлетворительна,
- красный - сомнительна.

Подробное описание метрик с минусами и плюсами можно ознакомиться тут.

Commits Count	Считает кол-во коммитов, выполненных определенным пользователем
Commit Size	Измеряет размер или строки кода, измененные при каждом коммите.
Commit Frequency	Проверяет, как часто разработчик вносит изменения в репозиторий.
Commit Messages	Оценивает качество и информативность сообщений о коммитах.
Branch Management	Проверяет, насколько хорошо разработчик справляется с ветвлением, слиянием и разрешением конфликтов.
Lines of Code (LOC)	Измеряет количество строк кода, добавленных, измененных или удаленных разработчиком.
Code Churn	Отслеживает частоту изменений кода (добавления, модификации, удаления) разработчиком.
Code Review Feedback	Оценивает качество обратной связи, предоставленной разработчиком во время проверки кода.
Code Duplication	Определяет наличие дублированного кода в репозитории.
Test Coverage	Измеряет процент кода, покрытого автоматическими тестами.
Code Complexity	Оценивает сложность кода с помощью таких показателей, как цикломатическая сложность.
Halstead Metrics	Метрика сложности Халстеда используется для измерения сложности программы без ее запуска.

Bug Fixing Speed	Измеряет время, затраченное разработчиком на устранение обнаруженных ошибок.
Bug Allocation Speed	Количество Issues
Collaboration Metrics	Отслеживает шаблоны совместной работы, такие как обзоры запросов на включение, обсуждения и взаимодействие с другими членами команды.
Feature Implementation	Измеряет успешность внедрения новых функций или пользовательских историй.