**Команда разработки DDDA**

**Состав:**

- Демьян Власюк — разработчик интерфейсной части приложения (front)
- Данила Картузов — разработчик серверной части приложения (back)
- Дмитрий Осиповский — проектный директор, тим-лид команды, разработчик полного цикла (full-stack)
- Алексей Перло — технический писатель, тестировщик

**Подход к разработке программного обеспечения:** комбинация Agile и Scrum

**Модель организации кода в Git:** Git-Flow

**Система управления проектом:** YouTrack

**Тип работы:** удалённый

**Способ онлайн взаимодействия:** Discord-собрания

**Название проекта: D-Track**

**Стек технологий front-части:**

- JavaScript — скриптовый язык разработки
- TypeScript — типизированный фреймворк для JS
- HTML + CSS — языки разметки
- NodeJS — платформа
- Webpack — система сборки
- ReactJS — библиотека для разработки SPA — приложения (single-page-application)
- React-Router — библиотека маршрутизации запросов
- ReactMUI — UI библиотека для создания пользовательских интерфейсов согласно принципам Material Design
- Redux-Toolkit — библиотека для управления состоянием и данными SPA

**Стек технологий back-части:**

- GoLang — язык разработки
- GoKit — фреймворк для написания микросервисной архитектуры
- Gin — фреймворк для написания REST-API сервера
- GodotEnv — библиотека для парсинга конфигурационных файлов переменной среды
- PostgreSQL — реляционная база данных для хранения информации
- Pgx — драйвер для PSQL
- JWT — авторизация

**Стек технологий тестовой-части:**

- Python — язык для написания тестов клиента
- Behave — фреймворк для написания самодокументируемых тестов
- Selenium — фреймворк для автоматизации действий в браузерах

**Инструменты:**

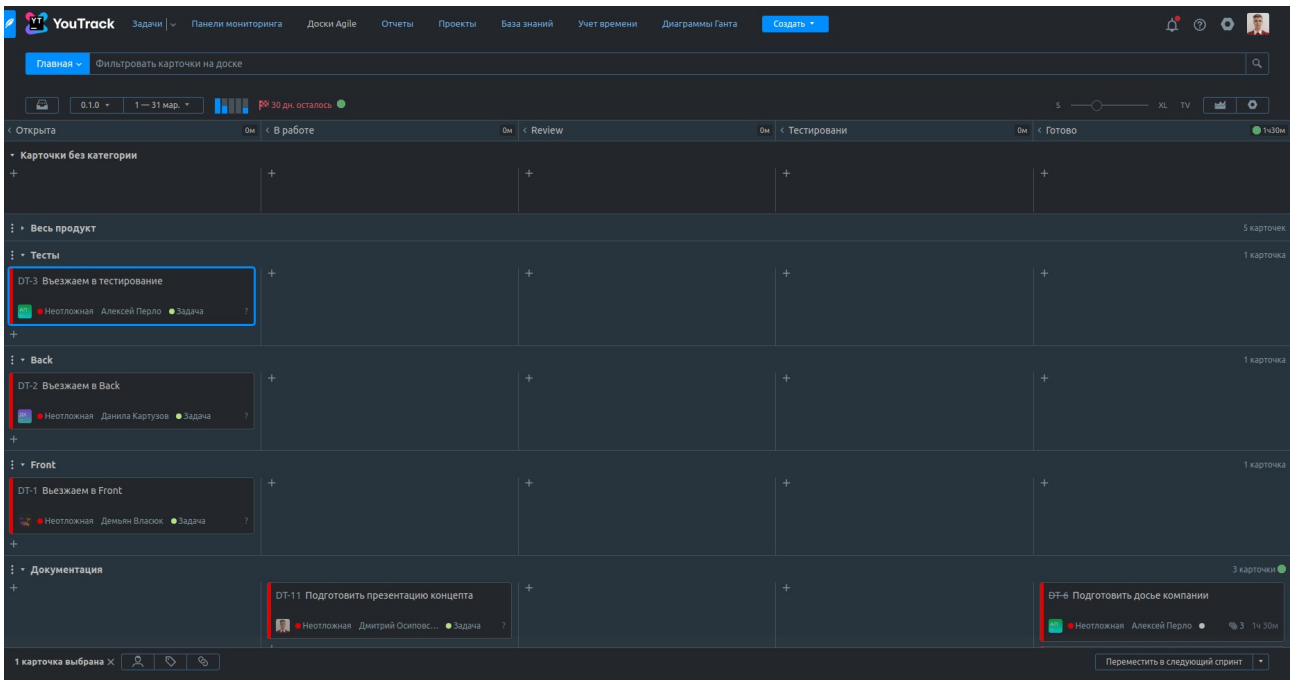
- Docker - программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений
- Docker-Compose — консольная утилита для аркестрирования контейнеров Docker
- Nginx — веб-сервер и почтовый прокси-сервер, работающий на Unix-подобных операционных системах. Используется для маршрутизации запросов между серверной и клиентской частями приложения
- GitLab — система контроля версий кода
- WebStorm — IDE для веб-разработки
- GoLand — IDE для разработки на языке GoLang
- Sublime-Merge — графический клиент для Git

**Доски Agile в YouTrack и рабочий процесс**

Команда работает над задачами создаваемыми на досках. Каждая задача может относится к внедрению новой функциональности, доработки, или улучшению существующей базы кода, или устранению ошибок. Задача прикрепляется к какому — либо участнику команды (может к нескольким). Задачи также делятся по компоненту продукта: back/front/deploy/весь продукт. Каждая задача проходит несколько стадий: открыта/в работе/review/тестирования/готово. Исполнитель задачи может установить время за которое он эту задачу выполнил, это помогает отслеживать эффективность работы каждого участника и корректировать сроки его работы в дальнейшим.

Задачи также имеют важный параметр: «приоритет». Приоритет отвечает за то, как быстро и в какой очерёдности должны выполняться задачи. Существует несколько типов приоритета среди них: незначительный/обычный/серьёзный/критический/неотложный. Приоритеты представлены в градации от самого медленного к самому «быстрому» типу реагирования.

Все задачи реализуются в рамках одного српинта. Спринт — это смысловая логическая часть, которая приносит какие — то улучшения в продукт. Спринт можно рассматривать, как очередную версию программы. Спринты, как правила являются короткими (до 4 недель), для быстрого реагирования и подстройки под нужды проекта.




## Модель организации кода в Git

В качестве модели организация кода в Git была выбрана модель построенная по принципам: Git-Flow. Данная модель полностью удовлетворяет потоковому подходу исполнения задач.

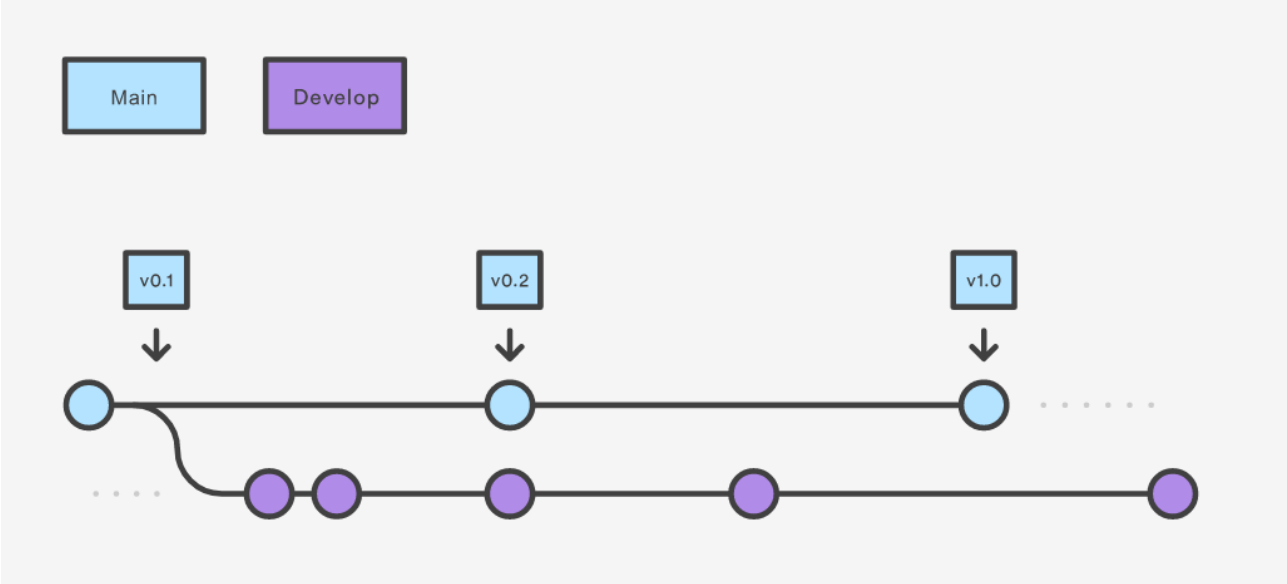
У нас имеется две основные ветки разработки: main и develop. В main — ветке содержится исправный код, который может быть получен и развёрнут в любое время. Каждая ревизия (конечная версия кода) сопровождается версионной меткой формата: 1.2.3, где 1 — мажорная версия, номер версии который отвечает за существенные изменения в продукте. Как правило увеличение этой цифры означает отсутствие обратной совместимости с прошлой версией. 2 — минорная версия, незначительные улучшение функциональности, которые не влияют на обратную совместимость продукта. 3 — версия hotfix — а, обновления, которое исправляет серьёзные ошибки, не меняя при этом никакой функциональности.

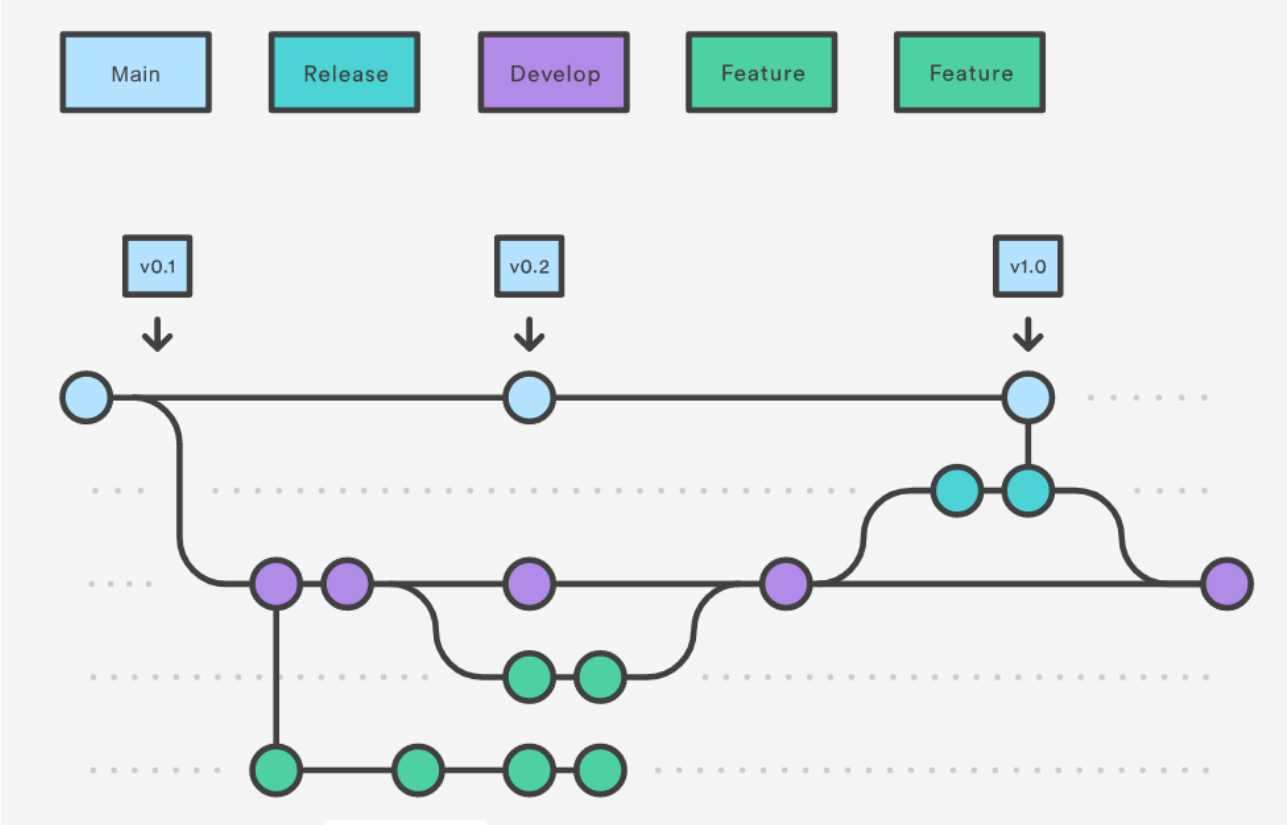
В ветке develop содержится код над которым работают в данный момент времени. Все новые функции, улучшения и модернизации вносятся именно сюда. Для каждой новой функции (для каждой новой задачи) создаётся от develop новая ветка с номером задачи из трекера в названии и в неё сохраняется весь новый функционал. Такие ветки называют: feature, т. е. Ветки новых функций. Как только функция дописана ветка сливается в исходную develop через merge request.

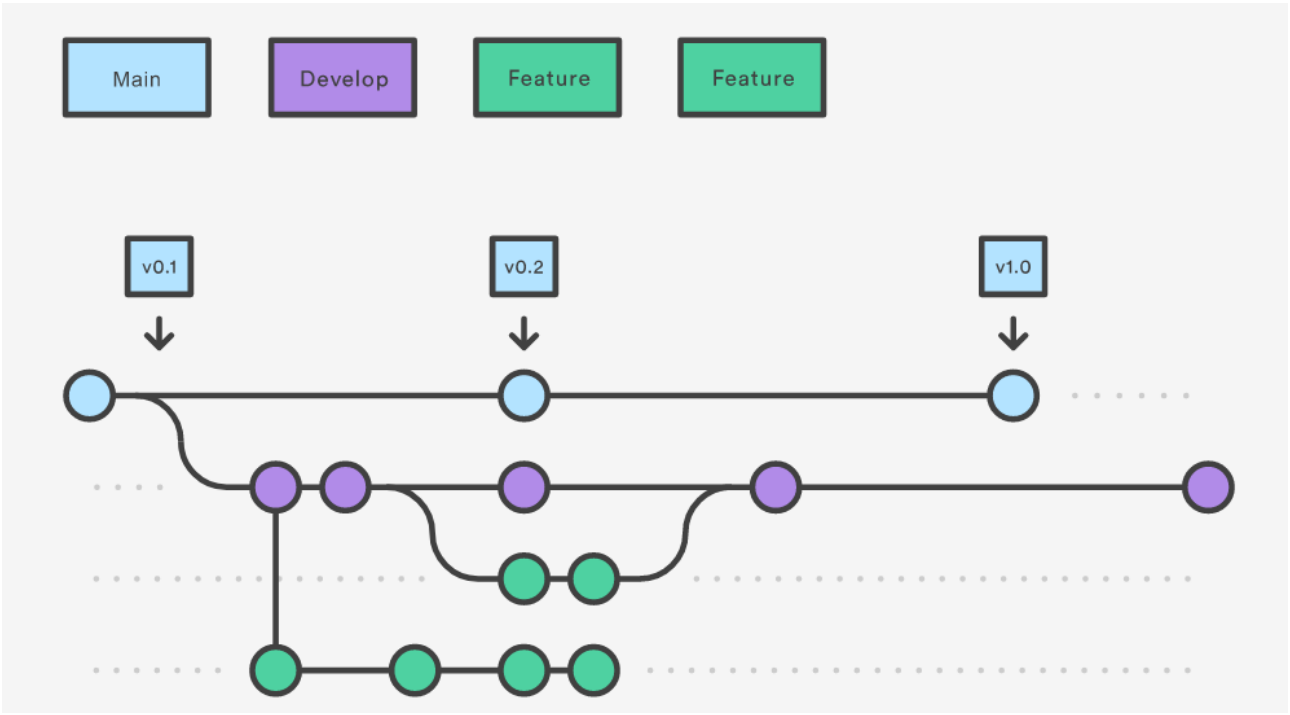
Merge request помогает контролировать процесс написания кода и отлавливать баги и ошибки на этапе code review, чтобы они в последствии не попали в главные ветки продукта.

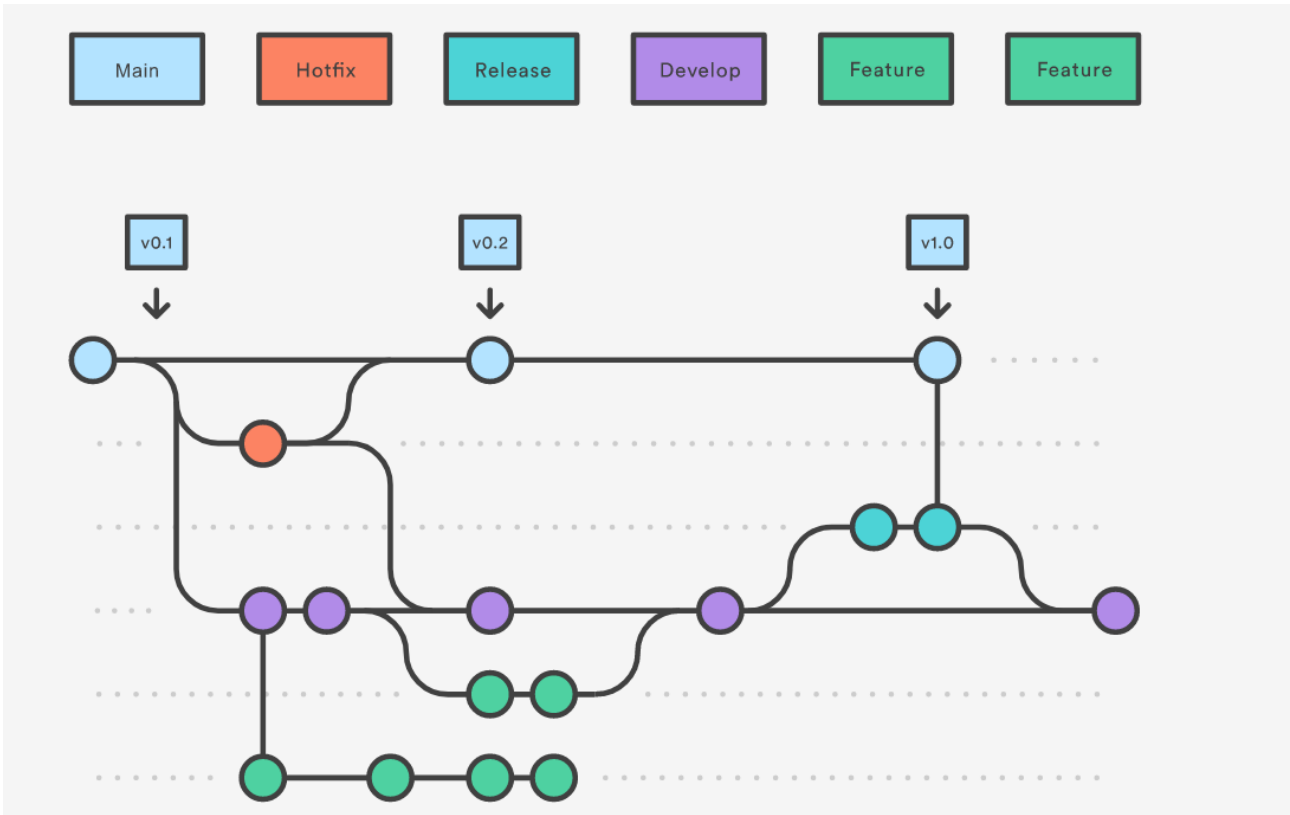
Для подготовки нового релиза от ветки develop отводится новая релизная ветка с номером готовившегося релиза в названии. Далее код в ней тестируется и отлаживается. После выявления большинства ошибок, она сливается в ветку main и помечается тегом новой версии и в ветку develop.

В случаи необходимости в hotfix-е, от ветки main отводится одноимённая ветка и в ней создаются заплатки для ошибок. После завершения эта ветка сливается в develop и main. Где в последней на слитый коммит выставляется тег новой версии.




Main
Develop
Feature
Feature
v0.1
v0.2
v1.0


Main
Hotfix
Release
Develop
Feature
Feature
v0.1
v0.2
v1.0