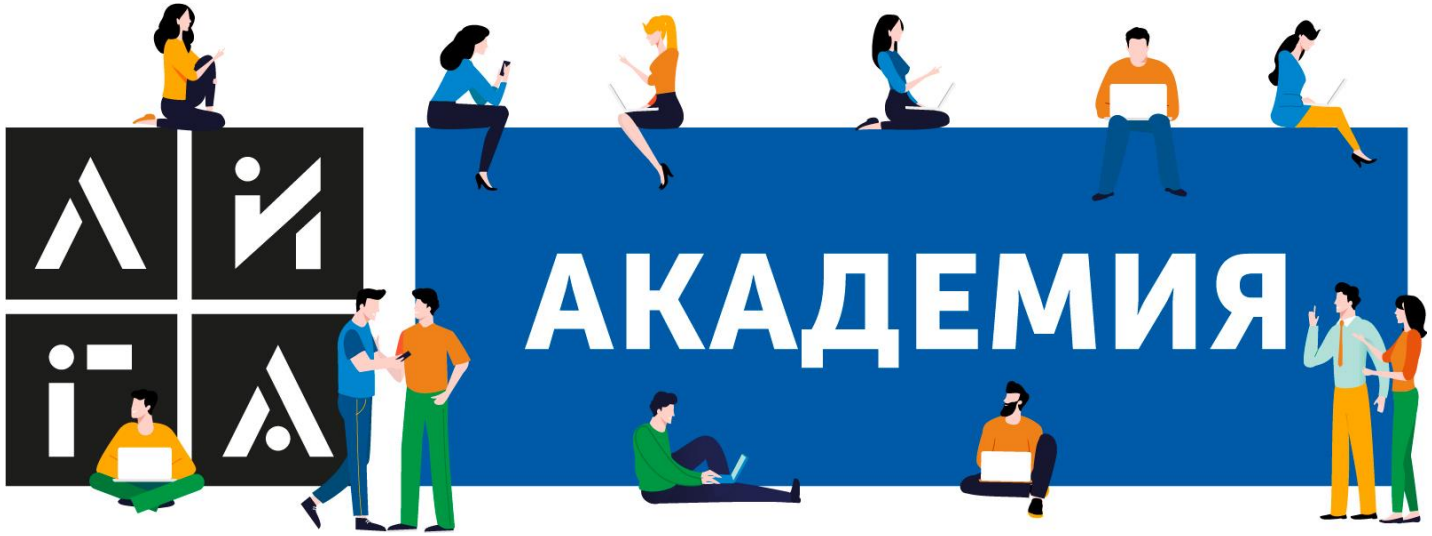

АКАДЕМИЯ

Основные принципы разработки

План занятия

1. Поговорим про системы сборки
2. Определим структуру общую будущего проекта на уровне сервисов
3. Организуем структуру проекта для каждого сервиса внутри
4. Расскажем про практики, применяемые у нас на проектах
5. Поговорим про git

Сборка и управление зависимостями

- Базируется на XML
- Ориентирован на описание структуры проекта
- Зрелое и стабильное средство, лучшие комьюнити и документация
- Огромный стандартный репозиторий

- Базируется на Groovy или Kotlin
- Ориентирован на описание цепочки задач
- Стремительно догоняет Maven по популярности
- Использует репозитории Maven

- Бегите, глупцы! (с) Гэндальф

Ant используется только в legacy проектах

АКАДЕМИЯ

Maven

Основные компоненты:

1) POM
2) Хранилище артефактов
3) mvn утилита
4) Плагины

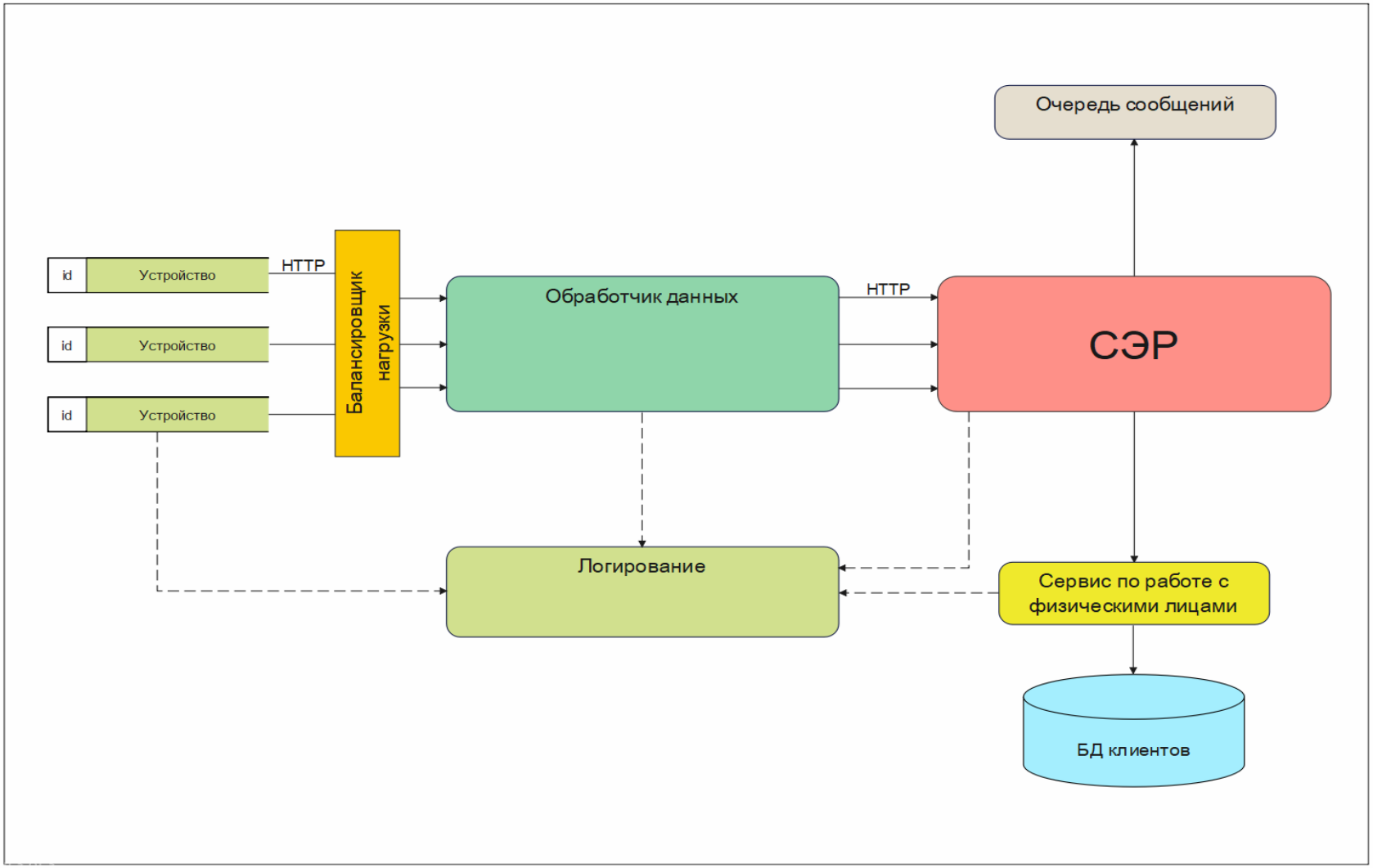
Что мы получаем:

- Сборка проекта + унификация
- Подключение внешних библиотек
- Автозапуск тестов
- Подключение внешних плагинов
- Организация структуры проекта
- Управление версиями

Циклы:

1. Базовый цикл
   - validate
   - compile
   - test
   - package
   - verify
   - install
   - deploy
2. Цикл отчистки
   - clean
3. Цикл создания сайта
   - site

Схема будущего проекта



Project

|

|__ dependencies-bom

|__ dto

|

|__ migration

|

|__ core

|

|__ core-api

__deploy

|

|_ файлы для деплоя

__build-tools

|

|_ файлы для сборки

Демо проекта

Виды систем контроля версий

Централизованные (CVCS)

- обмен кодом – только через центральный репозиторий
- устаревшая технология

Примеры: SVN, Perforce, MS TFS, ClearCase

Распределённые (DVCS)

- у каждого локально свой репозиторий
- синхронизация проходит через «центральный» репозиторий
- передавать изменения можно между любой парой репозиториев

Примеры: git, Mercurial, Bazaar

АКАДЕМИЯ

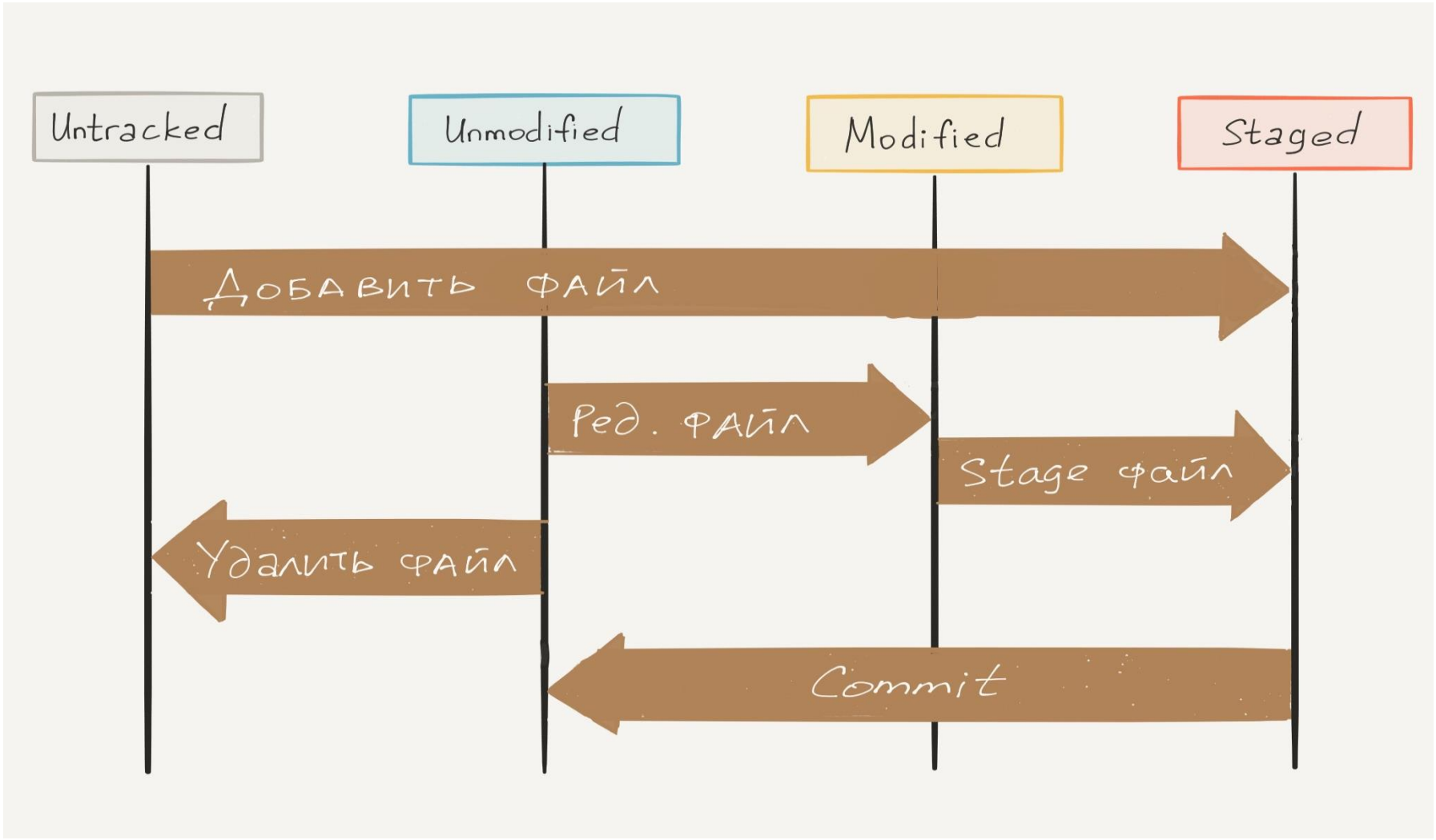
Большая часть операций выполняются локально

Если вы привыкли к ЦСКВ, где большинство операций страдают от задержек из-за работы с сетью, то этот аспект Git заставит вас думать, что боги скорости наделили Git несказанной мощью. Так как вся история проекта хранится прямо на вашем локальном диске, большинство операций кажутся чуть ли не мгновенными

Целостность данных

В Git для всего вычисляется хеш-сумма, и только потом происходит сохранение. В дальнейшем обращение к сохранённым объектам происходит по этой хеш-сумме. Это значит, что невозможно изменить содержимое файла или директории так, чтобы Git не узнал об этом. Данная функциональность встроена в Git на низком уровне и является неотъемлемой частью его философии. Вы не потеряете информацию во время её передачи и не получите повреждённый файл без ведома Git.

Процесс отслеживания файла git



Основные команды гита

git add . Добавляет все модифицированные файлы в статус отслеживаемых (stage)

git commit –a –m «» Коммитит все индексированные файлы

git push Отправляет все новые коммиты в удаленный репозиторий

git pull Синхронизирует состояние локального репозитория с удаленным

git checkout –b [ветка] Создает новую ветку и переключает на нее

git merge [ветка] Объединяет две ветки в одну (есть свои нюансы)

git rebase [ветка] Объединяет две ветки в одну (есть свои нюансы)

git stash Складывает все не закоммиченные изменения в внутренний буфер с возможностью извлечения

git fetch Обновляет удаленные локальные ветки

git reset [коммит] Сбрасывает состояние текущей ветки до указанного коммита

И др...

Но вообще можно и UI использовать😫

Полезная литература

Java Core

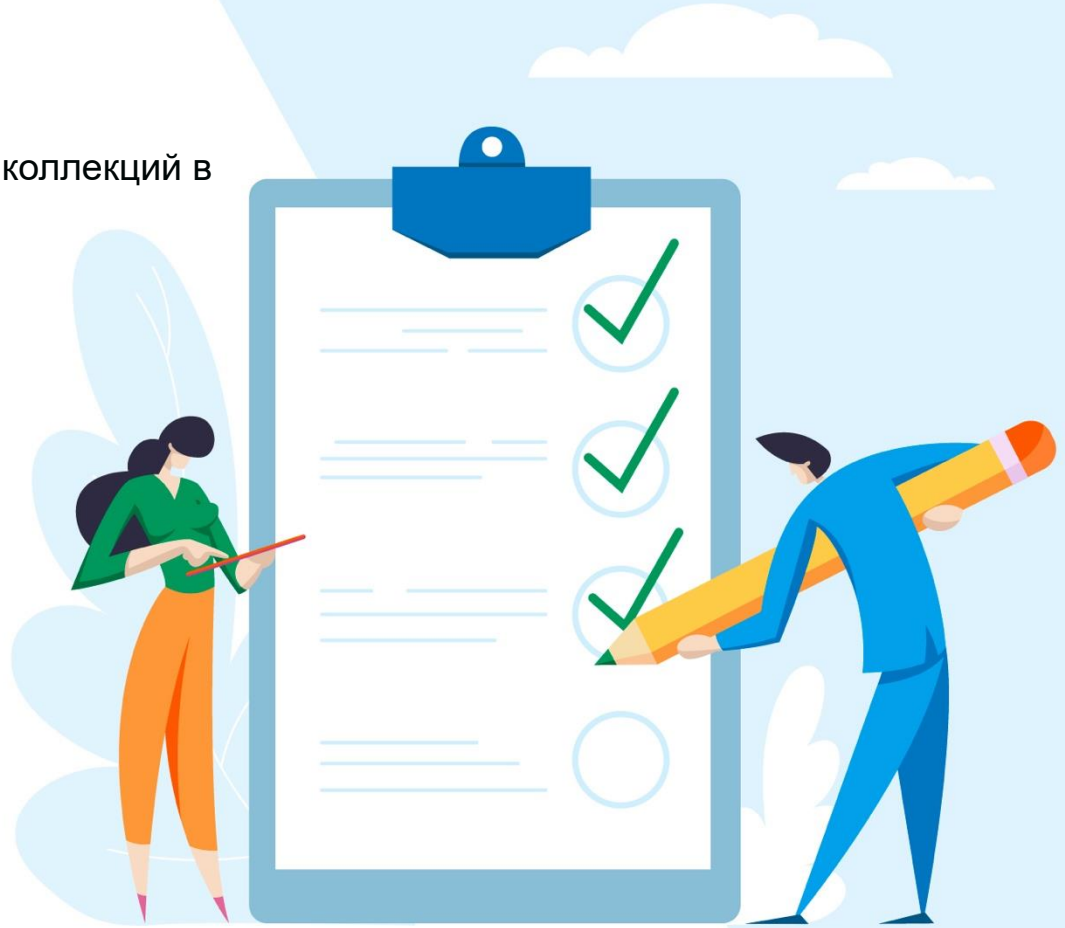
- proft.me - Достаточно полное описание всех коллекций в Java
- habr.com - как работает HashMap
- habr.com - Java Stream API
- habr.com - ещё Java Stream API

Maven

- dzone.com – vs Gradle (English)
- easyjava.ru – циклы maven
- java-online.ru - основные плагины Maven
- habr.com – основы maven

Git

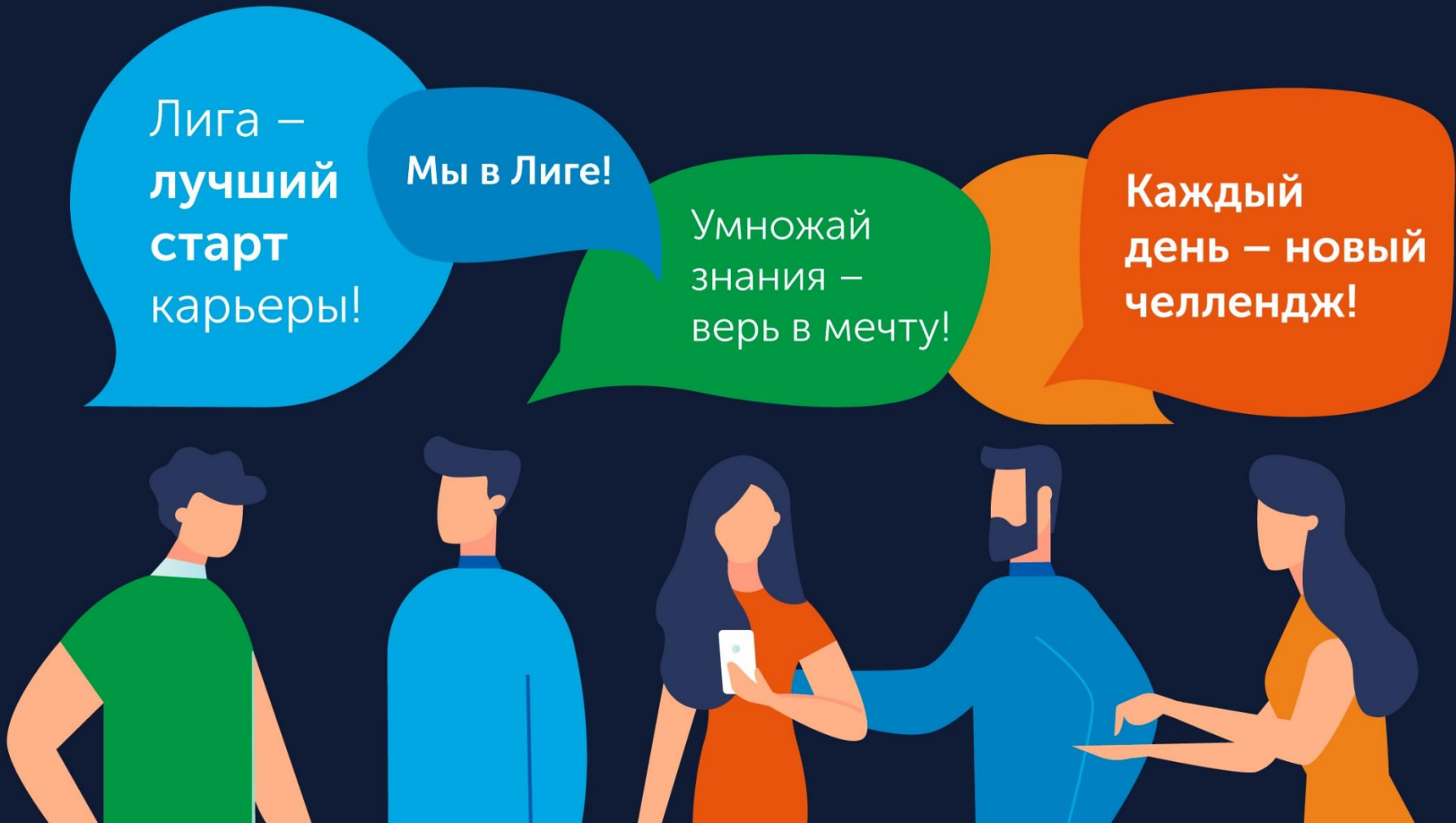
- habr.com – git rebase
- habr.com – введение в git
- coldfox.ru – подробное руководство по git



Задачи:

1. Написать в аналогичной структуре сервисы СЭР (medical-monitoring) и СФЛ (сервис физических лиц).
   Особенности: СФЛ будет работать с базой данных, там нужен будет дополнительный модуль “migration” в котором должен быть pom и resources.
2. Продумать и подготовить основные модели данных (dto) для ФЛ, уведомлений пользователей, отклонений по физ. показателям, приема информации от внешних устройств. Предварительно распределить по нужным модулям.
3. Всю работу выложить на github в ветке develop-3 для каждого проекта. Ссылка на репо должна быть в excel файле. (Далее цифра будет увеличиваться)
4. *Подключить статический анализатор кода PMD.

Всем спасибо!



Владислав Сыров

Разработчик

АКАДЕМИЯ