Материалы занятия

Курс: Разработка Web-приложений на Python, с применением Фреймворка Django
Дисциплина: Основы программирования на Python

Тема занятия №12: Работа с комплексными файлами - excel, json, word. Git и Jira

1. Git. Система контроля версий. Github

Что такое Git
Чтобы команда могла совместно работать над одним файлом, нужен сервис, где будет храниться папка с файлами — репозиторий.
Сервер видит, какие изменения вносит каждый член команды, и контролирует, нет ли конфликтующих между собой частей кода. Такие VCS называют централизованными. Это удобнее, чем постоянно делать резервные версии. Но если сервер будет неисправен, все наработки пропадут.

Поэтому удобнее воспользоваться такими сервисами, как Git, — с распределёнными VCS. Файлы синхронизируются между ПК и центральным репозиторием. Каждый программист получает весь репозиторий, а не конкретные файлы. Можно как отправить изменения на сервер, так и скачать те, что внесли другие программисты.

Ревьюер — обычно коллега-программист — одобряет запрос на добавление кода, и тогда этот код становится частью репозитория. Если есть замечания, ревьюер пишет комментарии, и разработчик их учитывает.

В случае с распределёнными VCS актуальные файлы можно взять у любого из членов команды. И даже если что-то вышло из строя у одного из разработчиков, это не страшно.

В чём разница между Git и GitHub
С помощью Git программисты и разработчики ориентируются в коде и отслеживают изменения. Git помогает вернуть файлы в исходное состояние и видеть изменения, внесённые в определённый период. Разработчик выполняет разные команды (например, commit, push), а все изменения синхронизируются с центральным репозиторием.
Git — это система контроля версий, а GitHub — онлайн-сервис, по сути социальная сеть. Одна из основных целей GitHub — быть единым местом для проектов с исходным кодом. Предполагается, что пользователь делится чем-то полезным, а другие люди смогут участвовать в разработке.

Ещё один вариант — использовать GitHub как хранилище проектов для портфолио: легко дать на них ссылку.

Где научиться работать в Git
Делимся подборкой ресурсов, где можно учиться работать в Git.
Бесплатная книга Pro Git на официальном сайте Git.
Бесплатный онлайн-курс на английском и на русском языке.
Интерактивный-визуальный онлайн-тренажёр по Git.
Шпаргалка по Git, автор которой разместил сайт на самом GitHub. Это ещё одна его возможность — GitHub Pages.

Ещё один вариант — игра для изучения GIT Oh My Git!. Игра визуализирует внутреннюю структуру Git в режиме реального времени. Ученик подключает реальный репозиторий, изучает команды и выполняет их. Видит результат на игровом поле.

Для новичков игра предлагает режим в виде карточек, которые позволяют лучше и быстрее запоминать команды. Для более продвинутых те же действия можно делать во встроенном терминале. Исходники на GitHub.

Как создать проект на GitHub

Чтобы выложить на GitHub проект, нужна регистрация на сайте сервиса. Первоначально необходимо указать адрес электронной почты, который станет логином для учётной записи. Сюда будут приходить все уведомления, и с помощью электронной почты можно будет восстановить доступ к GitHub в случае необходимости.

Создание аккаунта в GitHub

Разберём подробно, как происходит процесс создания и загрузки нового проекта на GitHub. Пройдём весь путь от регистрации на сайте сервиса до создания репозитория и работы с ветвями проекта.

Создание репозитория

Для того чтобы загрузить проект на GitHub, нужно начать с репозитория. Репозиторий используют для организации одного проекта. В нём содержатся файлы с кодом, необходимые для разработки проекта. Есть приватные и публичные репозитории. К приватным можно подключить ограниченное число человек, публичные легко находятся в поиске другими пользователями.

Репозитории могут включать файл README с информацией о проекте. Для загрузки на GitHub своего репозитория добавлять файл README не нужно. В этом случае нужен именно пустой репозиторий, куда будет залит существующий код. С пустым репозиторием на GitHub отобразит подсказку, как создать или клонировать репозиторий.

Чтобы загрузить или создать репозиторий, воспользуйтесь выпадающим меню в правом верхнем углу на любой из страниц GitHub и выберите New repository.

В поле Repository name введите уникальное название. В учётной записи не может быть двух одинаковых. Можно добавить краткое описание. А далее выбрать, каким будет репозиторий: общедоступным или приватным.

Работа с ветками

По умолчанию при загрузке нового проекта на GitHub в хранилище создаётся ветвь с именем main — это основная ветка. Дополнительные ветви используют для одновременного создания различных версий проекта. Это удобно, чтобы добавлять новые возможности, не изменяя основной исходный код. Можно использовать ветви для экспериментов и внесения правок до внесения изменений в основной код. Например, когда идёт работа над новыми фичами.

Когда создаётся ответвление от основной ветви проекта в GitHub, создаётся копия основной ветви в том виде, в котором она была в этот момент. Если кто-то другой вносил изменения в основную ветку, пока владелец ветки работал над своей, он сможет использовать обновления. Этого не происходит автоматически. Человек должен сам обновить свою ветку, если захочет.

Для создания ветви перейдите на вкладку «Code» в созданном репозитории и щёлкните на выпадающем списке веток и тегов в верхней части списка с надписью main. В текстовом поле введите название ветви, например edits, после чего нажмите «Create branch» — edits отразится в основной ветви.

Изменение файлов и коммиты
В Git и впоследствии на GitHub сохранённые изменения в файлах репозитория называются коммитами. Каждый коммит имеет связанное с ним сообщение коммита — описание, объясняющее, почему сделано изменение. Сообщения коммита фиксируют историю изменений для других участников. Важно написать, что вы конкретно изменили.
Если пользователь сделал коммит на своём компьютере, то сразу же в центральном репозитории коммит не появится. Чтобы он там появился, необходимо выполнить команду push — надо отправить свою текущую версию ветки.

Для того чтобы сделать коммит, в созданной ветке edits отредактируйте файл README.md. В поле «Edit file» напишите сообщение об изменениях и сохраните их.

● Таким образом, работа происходит так: Разработчик создаёт новую ветку репозитория, работает в ней, после чего делает коммит и пушит.
● Создаёт пулл реквест (pull request).

● Ревьюер перепроверяет работу и её одобряет, после чего объединяет с основной веткой.

Отчёты об ошибках
Issues позволяют отслеживать работу на GitHub, где происходит разработка. Можно выбрать наиболее удобный метод для рабочего процесса.
Например, вы можете создать отчёт об ошибках из репозитория, элемента в списке задач, заметки в проекте, комментария в проблеме, конкретной строке кода.

С тем, чтобы отслеживать отчёты об ошибках, помогут списки задач. А чтобы распределить связанные вопросы по категориям, удобно использовать метки.

Управление проектами
Репозиторий — это хранилище проектов в GitHub. Пользователь организует работу над этим кодом и видит доску с завершёнными задачами. Разберёмся, как выложить проект на GitHub.
Чтобы перейти во вкладку создания проекта, нажмите на аватар профиля в правом верхнем углу и выберите соответствующий пункт. Откроется меню, где отображаются все созданные проекты.

2. Jira. Командная разработка

Jira — это инструмент управления проектами, который помогает оптимизировать работу команды. Принцип работы сервиса похож на диспетчер задач в компьютере: с его помощью отслеживают запущенные процессы (проекты) и контролируют число ресурсов (сотрудников). В Jira проджект-менеджер грамотно распределяет сотрудников для выполнения задач и планирует работу. Например, если в работе уже четыре проекта, в которых задействованы все разработчики, значит, новый проект запускать не стоит, нужно дождаться завершения хотя бы одного.

У Jira есть бесплатная версия для команды до 10 человек, поэтому она популярна в небольших стартапах. Если команда больше, есть тариф Standard стоимостью $7 за одного пользователя. Он дает доступ к 250 ГБ хранилища и техподдержку в рабочие часы. В тарифе Premium стоимость одного пользователя $14, при этом он предоставляет неограниченный доступ к хранилищу и круглосуточную техническую поддержку. Оба тарифа дают возможность подключения до 20 000 пользователей и оплачиваются ежемесячно или

ежегодно. Самая дорогая подписка — Enterprise — оплачивается только ежегодно и нужна, если есть особые пожелания к безопасности и масштабированию.

Agile-разработка с Jira

В Jira работают Samsung, Coca Cola, Visa, Dropbox и Audi. NASA использует Jira для создания ПО, которое управляет беспилотными исследовательскими аппаратами в космосе, например марсоходом Curiosity. Еще все эти компании придерживаются методологии Agile.

Agile — это гибкая система разработки, в которой сложные задачи разбиваются на итерации — небольшие этапы. После каждого из них команда постепенно выдает готовые части продукта, их тестируют и оценивают. Одну итерацию называют спринтом (англ. sprint — бег на короткую дистанцию). В конце спринта команда подводит итоги и ставит себе задачи на следующий.

Главные принципы Agile:

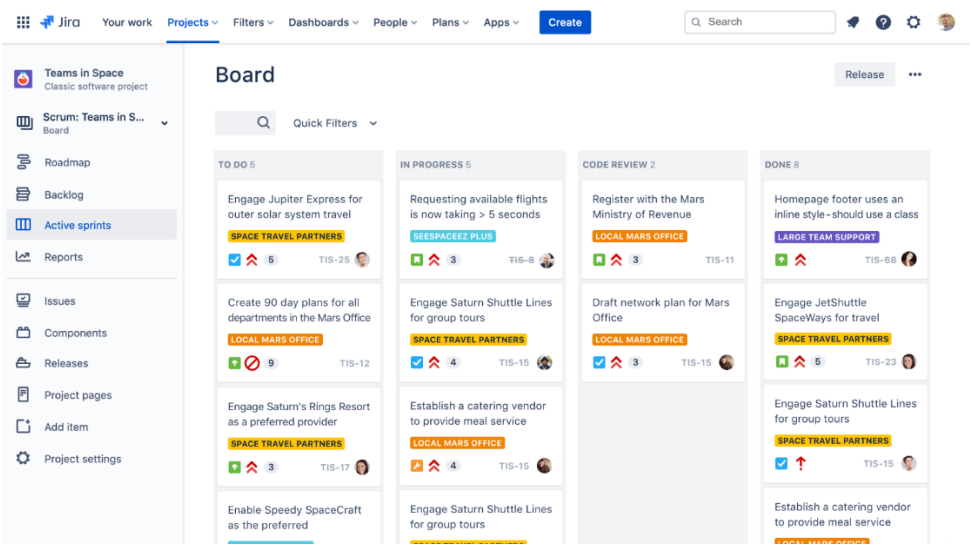
Люди и их взаимодействие важнее процессов и инструментов.
Работающий продукт важнее исчерпывающей документации.
Сотрудничество с заказчиком важнее согласования условий контракта.
Готовность к изменениям важнее следования первоначальному плану.
Есть два подхода к работе над проектом, основанные на Agile.

1. Методика Kanban — это способ визуализации задач с помощью досок, на которых задачи располагаются в соответствии со статусом. Стандартная канбан-доска делится на три колонки:

to do — список задач;
in progress — задачи, которые взяли в работу и выполняют;
done — завершенные задачи.

К этим колонкам можно добавлять другие. Например, между задачами в работе и завершенными поставить колонку с этапом тестирования. Такие доски используют для разных процессов — в команде маркетинга или при любой проектной работе.

2. Методика Scrum — в ней собраны все принципы гибкой разработки: деление на спринты, взаимодействие в команде и с заказчиком, нацеленность на рабочий продукт. Для визуализации рабочего процесса в Scrum тоже используют доски, на которых отслеживается процесс разработки. Отличие от канбан-досок в том, что самую важную роль играют спринты и задача не может находиться в работе дольше, чем длится спринт. Доски бывают физическими — тогда команда перемещает задачи, переклеивая стикеры. Такую методику скрам-мастера рекомендуют для небольших команд, у которых все разработчики в одном офисе. В Jira виртуальные скрам-доски выглядят так:



Для чего используют Jira

Этот инструмент создавали для отслеживания статуса задач и ошибок, но со временем его функционал расширился. Сегодня в Jira можно управлять процессом разработки от идеи до запуска готового продукта. Кроме IT-команд, ее используют маркетологи, аналитики, тестировщики и другие специалисты.

Для чего может помочь Jira:

Управление требованиями. Требования — это вводные данные для работы над проектом. Их пишут в отдельном документе вместе с заказчиком, чтобы не возникало разногласий в процессе работы, а разработчики могли на них ориентироваться. Чтобы изменить или составить требования для команды, используют Jira в сочетании с Confluence — инструментом для совместной работы. В нем можно создавать, обсуждать и редактировать документы.

Управление продуктами. Команды составляют в Jira дорожные карты — пошаговые планы масштабных проектов. Такие карты помогают наладить взаимодействие между отделами. Например, при грамотном управлении маркетологи могут планировать промокампанию параллельно с разработкой, а не ждать готового продукта. В дорожных картах не прописываются подробные задачи и методы выполнения, в них расставляются цели, приоритеты и обозначаются зависимости работы одного отдела от работы другого.

Управление проектами. Jira настраивается под проекты, поэтому такой инструмент полезен проджект-менеджерам. Можно визуально отследить путь каждой задачи от создания до результата: генерация идей и гипотез, создание прототипа, дизайн, согласование дизайн-концепта, разработка, создание контента, тестирование.

Интерфейс

Внешне Jira похожа на любой другой таск-менеджер.

В верхней строке меню шесть вкладок:

Ваша работа (Your Work) — тут отображаются проекты конкретного члена команды, в которых он исполнитель.

Проекты (Projects) — вкладка с доступными досками команды, активными спринтами и отчетами о работе.

Фильтры (Filters) — таблица сортировки проектов по авторам, проектам, статусам и другим показателям.

Дашборды (Dashboards) — аналитические сводки по проектам.

Люди (People) — список профилей членов команды.

Приложения (Apps) — сторонние сервисы, которые интегрированы в Jira и расширяют ее функционал.

Боковое меню состоит из элементов управления проектами:

Доска (Board) — это вкладка, в которой хранятся доски проектов, доступных команде. Между ними переключаются с помощью кнопки «Вниз».

Дорожная карта (Roadmap) — трекер, с помощью которого выстраивается картина работы над проектом; в нем на линии времени отображаются этапы разработки до финального релиза.

Бэклог (Backlog) — список рабочих задач, расставленных по приоритетам на основе дорожной карты: важные задачи — в начале списка, а менее приоритетные — в конце. Каждая задача в бэклоге — отдельный спринт.

Активные спринты (Active sprints) — список спринтов (задач из бэклога), которые находятся у команды в работе.

Отчеты (Reports) — в этой вкладке более 20 параметров: соотношение нагрузки на команду в спринтах, ее эффективность, длительность спринтов, прогресс проекта и прочее.