Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет информационных технологий и программирования

Дисциплина «Введение в цифровую культуру и программирование»

Лабораторная работа №4

Системы управления репозиториями.

Выполнил студент группы № M3109

Хлучин Георгий Владимирович.

Подпись:

Проверил:

Повышев Владислав Вячеславович

Санкт-Петербург

2021

Сравнение систем управления проектами.

Важнейшие инструменты для эффективной работы предприятия — это сервисы планирования и управления проектами. В зависимости от размеров, отрасли и целей это могут быть как простые системы контроля, так и крупные бизнес-решения. Сегодня мы разберем три сервиса для управления проектами внутри организации, а также сравним их функционал сравним функционал.

Таблица:

	Мобильное приложение	Встроенный мессенджер	Диаграмма Ганта	Выставление статусов
Trello	+	-	-	+
BaseCamp	-	-	+	+
Clickup	+	+	-	+

Предыстория:

1. Trello

   Trello - это новая компания одного продукта. С 2010 года компания-разработчик программного обеспечения Fog Creek Software (Нью Йорк, США) проводила еженедельные внутренние исследования потенциальных продуктов. Там, в январе 2011 года и был презентован прототип, предназначенный для решения высокоуровневых задач планирования, тогда еще Trellis.

   После бета тестирования и презентации, набрав за год более 500 000 пользователей, Fog Creek создает приложений и под Android. Тогда же у Trello появляется маскот - говорящий пес-хаски, моделью для которого послужил питомец сооснователя Fog Creek. В июле 2014, насчитывая 4,75 миллионов пользователей, Trello отделяется от Fog Creek и становится Trello, Inc. С 2017 года Trello принадлежит компании Atlassian.

На данный момент приложение Trello — одна из самых популярных систем управления проектами в режиме онлайн, которая пользуется особенным спросом среди небольших компаний и стартапов. Организация работы строится по методу “канбан” (от японского - точно в срок), базируясь на четком распределении задач между сотрудниками.

По сути, это аналог рабочих досок со стикерами на ней, только в веб-приложении. Именно благодаря своей простоте и фактически неограниченному бесплатному функционалу, Trello удалось снискать всеобщую любовь.

2. BaseCamp
   Компания 37Signals была основана в 1999 году, сначала занимаясь заказным веб-дизайном. Позже, она сконцентрировалась на разработке веб-приложений, распространяемых как услуга, а в 2014 году была переименована в Basecamp - по названию их самого популярного продукта.

   Известна 37Signals и за создание фреймворка Ruby on Rails. Компания базируется в Чикаго, при этом ее сотрудники обитают по всему миру - опыт их удаленной работы даже описан в книге, занимающей сейчас топы в бизнес-изданиях на Амазоне.

   Разрабатывая приложение Basecamp, компания инвестировала в его сильные стороны - простоту и скорость. Команда выделила их, исходя из неизменности потребностей клиентов в этой сфере. Цитируя компанию, “10 years from now people aren’t going to say “I wish Basecamp was harder to use” or “I wish Basecamp was slower and less reliable”. (Спустя 10 лет, вряд ли кто-то скажет - “эх, хотел бы я, чтобы Basecamp было

тяжелее использовать” или “вот бы Basecamp был менее надежным и более медленным!”).

Basecamp - простая и эффективная система управления проектами. Приложение содержит задачи, календарь, дискуссии, профайлы, вики-документы, файлы, лог проекта. Подойдет менеджерам, которым необходимо вести несколько согласований по различным вопросам в рамках проекта, а также руководителю, который предпочитает контролировать работу сразу нескольких сотрудников.

Basecamp впишется в рабочий процесс, если в нем задействованы постоянные пересылки медиафайлов, например - при согласовании дизайна или контента с визуальным наполнением. Здесь довольно большой функционал коммуникации, опросов, контроля в режиме реального времени.

Кстати, компания открыто заявляет, что не гонится за количеством пользователей программой, предпочитая предоставлять услуги тем, кому софт действительно подходит. На сайте компании есть целая витрина крутых проектов, созданных с помощью этого ПО, так что, можно попытать удачу в такой бесплатной взаимовыгодной рекламе.

3. Clickup
   Компания Clickup была основана в 2016 году - это стартап из Сан Франциско, США, который считает, что норма - это скучно. Исходя из этого, небольшая команда (не больше 50 человек) спроектировала свое оригинальное приложения по проектному управлению. В Clickup считают, что существующие платформы управления проектами слишком сложны или недостаточно функциональны. И они создали приложение,

балансирующие посередине - имеющее необходимые функции, но и интуитивно понятное.

Приложение ClickUp предоставляет пользователям три разных варианта просмотра задач: в виде доски, таблицы и бокса. Здесь можно создавать собственные статусы для каждого уникального проекта, которые можно редактировать, переупорядочивать и окрашивать доски в соответствии с потребностями организации.

Менеджеры могут назначать задачи нескольким людям одновременно, а также беспрепятственно управлять несколькими задачами сами, с помощью многозадачного табличного представления.

Панель ClickUp обеспечивает Agile-представление, разработанное под SCRUM, обеспечивая простую, но в то же время высокоинформативную репрезентацию данных.

Технические параметры:

1. Trello
   Оплата: бесплатно/платно (месячная или годовая оплата). Стоит отметить - freemium-модель у Trello позволяет полноценно работать, с немного урезанным функционалом.

   На чем доступно: Web, IOS App, Android App.

   Интеграция с другими сервисами: MailChimp, Slack, Dropbox, Google Drive, Google Hangouts, InVision, Evernote, GitHub, Bitbucket Cloud, Jira, Salesforce, Zendesk, other.

Варианты доступа к проекту: личный проект, групповой проект, открытый проект, проект с совместным доступом.

Лайв чаты: отсутствуют.

Подробные отчеты о выполнении задач: отсутствуют.

Шаблоны проектов: отсутствуют.

2. BaseCamp
   Оплата: ежемесячная.

   На чем доступно: Web, IOS App, Android App, Desktop App.

   Интеграция с другими сервисами: Dropbox, Google Drive, Box, One Drive, other.

   Варианты доступа к проекту: групповой проект, проект с совместным доступом.

   Лайв чаты: есть (+ автоматические опросы).

   Подробные отчеты о выполнении задач: отсутствуют

   Шаблоны проектов: есть.

3. Clickup
   Оплата: бесплатно (урезанный функционал)/ежемесячно/ежегодно.

   На чем доступно: Web, IOS App, Android App.

   Интеграция с другими сервисами: Slack, GitHub, Google Drive, Dropbox, GitLab, Bitbucket, other.

Варианты доступа к проекту: групповой проект, личній проект.

Лайв чаты: отсутствуют.

Подробные отчеты: есть.

Шаблоны проектов: есть.

Логотипы:

Описание ClickUp(выбран для выполнения лабораторной работы)

ClickUp — сервис, включающий себя все необходимые функции для таск-менеджмента. Сервис интуитивно понятен для пользователя, облегчённый дизайн и обучающие подсказки, всплывающие при первом использовании ClickUp, помогут в течении часа разобраться в основных функциях. ClickUp совмещает в себе десятки различных функции: составления списка целей и задач, чат, создание документов, уведомлений, составление расписания и многие другие.

Цели и задачи в ClickUp распределяются по иерархии: «Рабочее пространство» — основа иерархии, представляет собой деятельность организации в целом; «рабочее пространство» подразделяется на отдельные пространства («Spaces») со своей структурой, каждое «пространство» имеет свои настройки и пользователей, например, может объединять в себе работников из одного отдела; «Листы» — папка, состоящая из отдельных «задач», «чек-листов», подзадач, хранящихся в «пространстве». Для просмотра задач и других элементов управления проектами на

сервисе существует несколько вариантов просмотра: дашборд, лист с задачами, канбан-доска, календарь и диаграмма Ганта. Задачи в ClickUp можно полностью кастомизировать под нужды пользователей: изменять цвет, добавлять ссылки, документы, изображения и видео, графики, назначать роли для сотрудников, вести чат, устанавливать сроки и т. д.

Основные функции ClickUp:

Чат в реальном времени — возможность оставлять комментарии к задачам, где пользователи могут вести непрерывную беседу между собой;

Пользовательское разнообразие элементов для изменения — изменение цвета элементов и сервиса, тёмная тема, изменяемые поля и т. д.;

Создание документов — встроенный редактор документов, которые можно использовать в приложении;

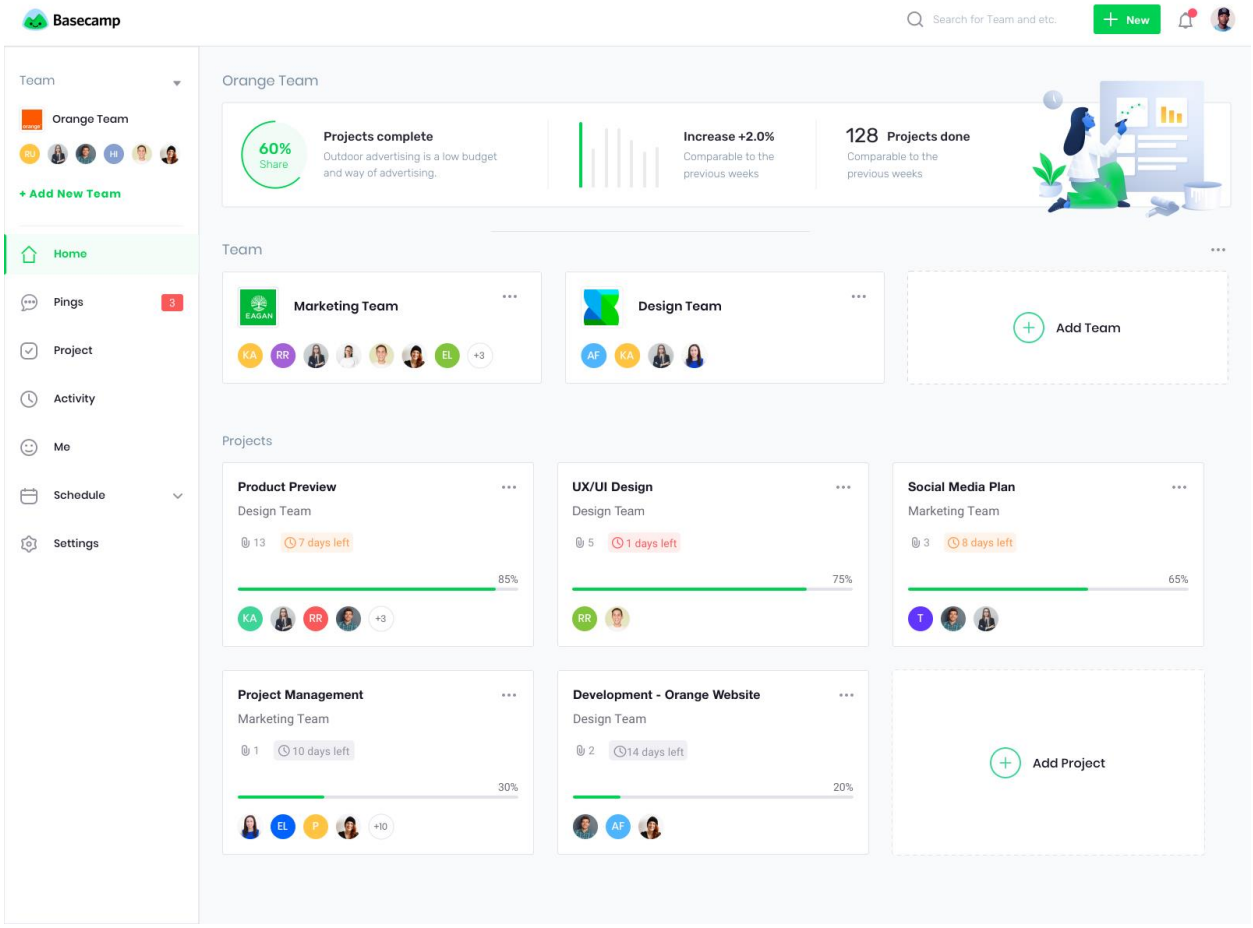
Установка и учёт времени для задач;

Проектные шаблоны — десятки готовых шаблонов для сфер бизнеса (маркетинг, дизайн и т. д.);

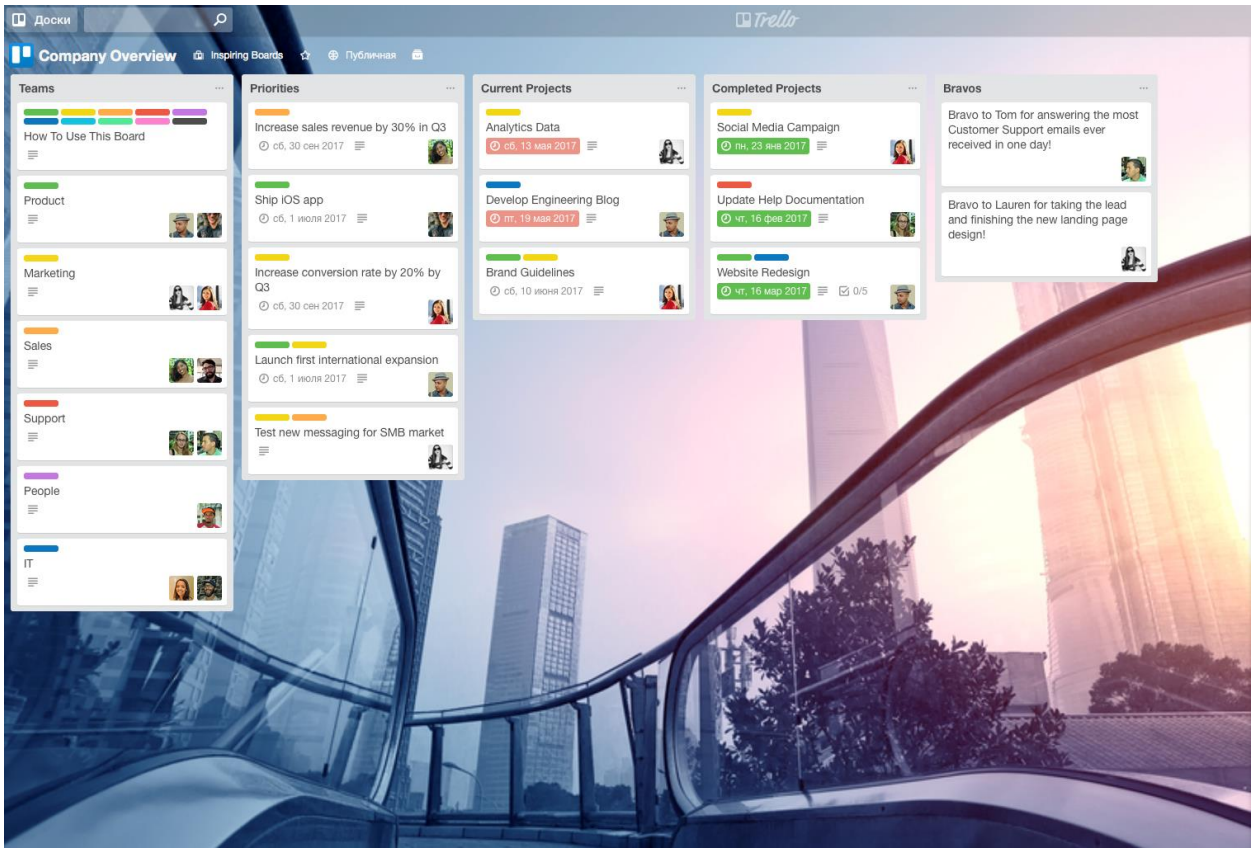
Установка приоритетов к задачам.

Скриншоты рабочего пространства:

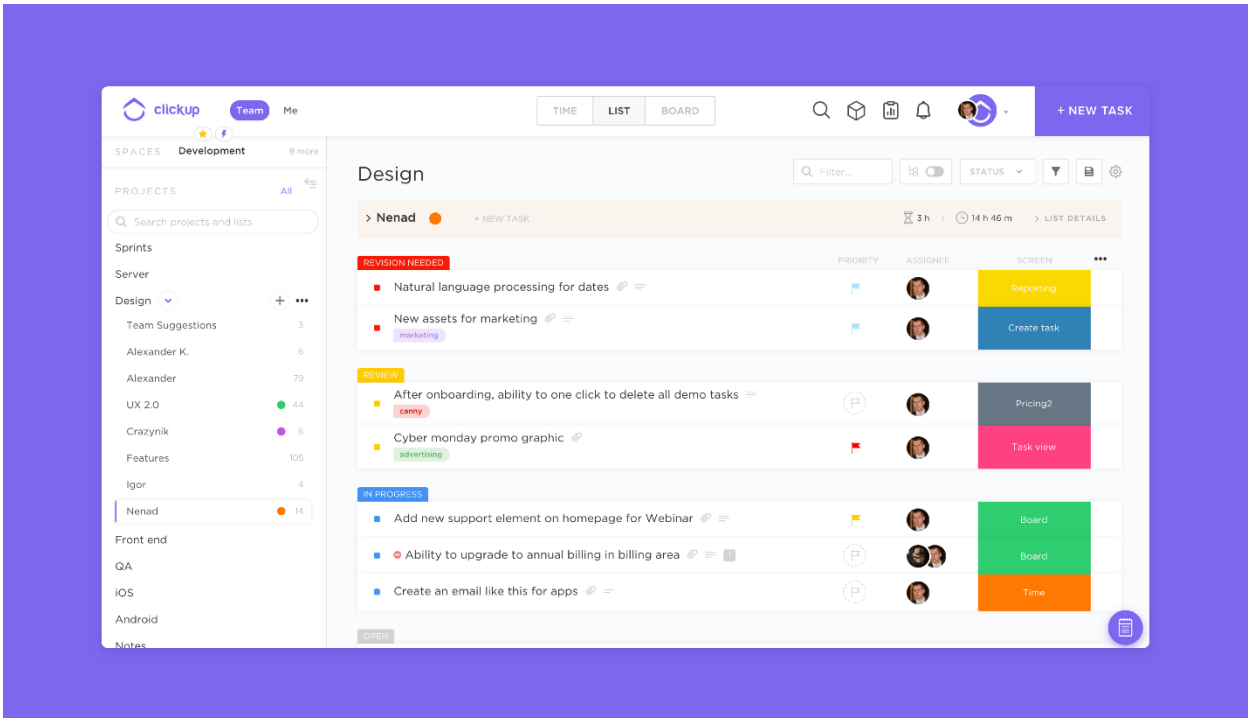
1. BaseCamp


Basecamp
Search for Team and etc.
New
Team
Orange Team
+ Add New Team
Home
Pings
3
Project
Activity
Me
Schedule
Settings
Orange Team
60%
Share
Projects complete
Outdoor advertising is a low budget and way of advertising.
Increase +2.0%
Comparable to the previous weeks
128 Projects done
Comparable to the previous weeks
Team
Marketing Team
Design Team
Add Team
Projects
Product Preview
Design Team
13
7 days left
85%
UX/UI Design
Design Team
5
1 days left
75%
Social Media Plan
Marketing Team
3
8 days left
65%
Project Management
Marketing Team
1
10 days left
30%
Development - Orange Website
Design Team
2
14 days left
20%
Add Project

2.Trello


Доски
Trello
Company Overview
Inspiring Boards
Публичная
Teams
How To Use This Board
Product
Marketing
Sales
Support
People
IT
Priorities
Increase sales revenue by 30% in Q3
сб, 30 сен 2017
Ship iOS app
сб, 1 июля 2017
Increase conversion rate by 20% by Q3
сб, 30 сен 2017
Launch first international expansion
сб, 1 июля 2017
Test new messaging for SMB market
Current Projects
Analytics Data
сб, 13 мая 2017
Develop Engineering Blog
пт, 19 мая 2017
Brand Guidelines
сб, 10 июня 2017
Completed Projects
Social Media Campaign
пн, 23 янв 2017
Update Help Documentation
чт, 16 фев 2017
Website Redesign
чт, 16 мар 2017
0/5
Bravos
Bravo to Tom for answering the most Customer Support emails ever received in one day!
Bravo to Lauren for taking the lead and finishing the new landing page design!

3.Clickup



CI\CD.

Непрерывная интеграция (Continuous Integration, CI) и непрерывная поставка (Continuous Delivery, CD) представляют собой культуру, набор принципов и практик, которые позволяют разработчикам чаще и надежнее развертывать изменения программного обеспечения.

CI/CD — это одна из DevOps-практик. Она также относится и к agile-практикам: автоматизация развертывания позволяет разработчикам сосредоточиться на реализации бизнес-требований, на качестве кода и безопасности.

Определение CI/CD

Непрерывная интеграция — это методология разработки и набор практик, при которых в код вносятся небольшие изменения с частыми коммитами. И поскольку большинство современных приложений разрабатываются с использованием различных платформ и инструментов, то появляется необходимость в механизме интеграции и тестировании вносимых изменений.

С технической точки зрения, цель CI — обеспечить последовательный и автоматизированный способ сборки, упаковки и тестирования приложений. При налаженном процессе непрерывной интеграции разработчики с большей вероятностью будут делать частые коммиты, что, в свою очередь, будет способствовать улучшению коммуникации и повышению качества программного обеспечения.

Непрерывная поставка начинается там, где заканчивается непрерывная интеграция. Она автоматизирует развертывание приложений в различные окружения: большинство разработчиков работают как с продакшн-окружением, так и со средами разработки и тестирования.

Инструменты CI/CD помогают настраивать специфические параметры окружения, которые конфигурируются при развертывании. А также CI/CD-автоматизация выполняет необходимые запросы к веб-серверам, базам данных и другим сервисам, которые могут нуждаться в перезапуске или выполнении каких-то дополнительных действий при развертывании приложения.

Непрерывная интеграция и непрерывная поставка нуждаются в непрерывном тестировании, поскольку конечная цель — разработка качественных приложений. Непрерывное тестирование часто реализуется в виде набора различных автоматизированных тестов (регрессионных, производительности и других), которые выполняются в CI/CD-конвейере.

Зрелая практика CI/CD позволяет реализовать непрерывное развертывание: при успешном прохождении кода через CI/CD-конвейер, сборки автоматически развертываются в продакшн-окружении. Команды, практикующие непрерывную поставку, могут позволить себе ежедневное или даже ежечасное развертывание. Хотя здесь стоит отметить, что непрерывная поставка подходит не для всех бизнес-приложений.

Непрерывная интеграция улучшает коммуникации и качество

Непрерывная интеграция — это методология разработки, основанная на регламентированных процессах и автоматизации. При внедренной непрерывной интеграции разработчики часто коммитят свой код в репозиторий исходного кода. И большинство команд придерживается правила коммитить как минимум один раз в день. При небольших изменениях проще выявлять дефекты и различные проблемы, чем при больших изменениях, над которыми работали в течение длительного периода времени. Кроме того, работа с короткими циклами коммитов уменьшает вероятность изменения одной и той же части кода несколькими разработчиками, что может привести к конфликтам при слиянии.

Команды, внедряющие непрерывную интеграцию, часто начинают с настройки системы контроля версий и определения порядка работы. Несмотря на то что коммиты делаются часто, реализация фич и исправление багов могут выполняться довольно долго. Для контроля того, какие фичи и код готовы существует несколько подходов.

Многие используют фича-флаги (feature flag) — механизм для включения и выключения функционала в рантайме. Функционал, который еще находится в стадии разработки, оборачивается фича-флагами и развертывается из master-ветки в продакшн, но отключается до тех пор, пока не будет полностью готов к использованию. По данным недавнего исследования 63 процента команд, использующих фича-флаги, говорят об улучшении тестируемости и повышении качества программного обеспечения. Для работы с фича-флагами есть специальные инструменты, такие как CloudBees Rollout, Optimizely Rollouts и LaunchDarkly, которые интегрируются в CI/CD и позволяют выполнять конфигурацию на уровне фич.

Другой способ работы с фичами — использование веток в системе контроля версий. В этом случае надо определить модель ветвления (например, такую как Gitflow) и описать как код попадает в ветки разработки, тестирования и продакшена. Для фич с длительным циклом разработки создаются отдельные фича-ветки. После завершения работы над фичей разработчики сливают изменения из фича-ветки в основную ветку разработки. Такой подход работает хорошо, но может быть неудобен, если одновременно разрабатывается много фич.

Этап сборки заключается в автоматизации упаковки необходимого программного обеспечения, базы данных и других компонент. Например, если вы разрабатываете Java-приложение, то CI упакует все статические файлы, такие как HTML, CSS и JavaScript, вместе с Java-приложением и скриптами базы данных.

CI не только упакует все компоненты программного обеспечения и базы данных, но также автоматически выполнит модульные тесты и другие виды тестирования. Такое тестирование позволяет разработчикам получить обратную связь о том, что сделанные изменения ничего не сломали.

Большинство CI/CD-инструментов позволяет запускать сборку вручную, по коммиту или по расписанию. Командам необходимо обсудить расписание сборки, которое подходит для них в зависимости от численности команды, ожидаемого количества ежедневных коммитов и других критериев. Важно, чтобы коммиты и сборка были быстрыми, иначе долгая сборка может стать препятствием для разработчиков, пытающихся быстро и часто коммитить.

Непрерывное тестирование — это больше, чем автоматизация тестирования

Фреймворки для автоматизированного тестирования помогают QA-инженерам разрабатывать, запускать и автоматизировать различные виды тестов, которые помогают разработчикам отслеживать успешность сборки. Тестирование включает в себя функциональные тесты, разрабатываемые в конце каждого спринта и объединяемые в регрессионные тесты для всего приложения. Регрессионные тесты информируют команду: не сломали ли их изменения что-то в другой части приложения.

Лучшая практика заключается в том, чтобы требовать от разработчиков запускать все или часть регрессионных тестов в своих локальных окружениях. Это гарантирует, что разработчики будут коммитить уже проверенный код.

Регрессионные тесты — это только начало. Тестирование производительности, тестирование API, статический анализ кода, тестирование безопасности — эти и другие виды тестирования тоже можно автоматизировать. Ключевым моментом является возможность запуска этих тестов из командной строки, через веб-хук (webhook) или через веб-сервис и возврат результата выполнения: успешный был тест или нет.

Непрерывное тестирование подразумевает не только автоматизацию, но и интеграцию автоматического тестирования в конвейер CI/CD. Модульные и функциональные тесты могут быть частью CI и выявлять проблемы до или во время запуска CI-конвейера. Тесты, требующие развертывания полного окружения, такие как тестирование производительности и безопасности, часто являются частью CD и выполняются после разворачивания сборок в целевых средах.

CD-конвейер автоматизирует поставку изменений в различные окружения

Непрерывная поставка — это автоматическое развертывание приложения в целевое окружение. Обычно разработчики работают с одним или несколькими окружениями разработки и тестирования, в которых приложение развертывается для тестирования и ревью. Для этого используются такие CI/CD-инструменты как Jenkins, CircleCI, AWS CodeBuild, Azure DevOps, Atlassian Bamboo, Travis CI.

Типичный CD-конвейер состоит из этапов сборки, тестирования и развертывания. Более сложные конвейеры включают в себя следующие этапы:

Получение кода из системы контроля версий и выполнение сборки.

Настройка инфраструктуры, автоматизированной через подход “инфраструктура как код”.

Копирование кода в целевую среду.

Настройка переменных окружения для целевой среды.

Развертывание компонентов приложения (веб-серверы, API-сервисы, базы данных).

Выполнение дополнительных действий, таких как перезапуск сервисов или вызов сервисов, необходимых для работоспособности новых изменений.

Выполнение тестов и откат изменений окружения в случае провала тестов.

Логирование и отправка оповещений о состоянии поставки.

Например, в Jenkins конвейер определяется в файле Jenkinsfile, в котором описываются различные этапы, такие как сборка (build), тестирование (test) и развертывание (deploy). Там же описываются переменные окружения, секретные ключи, сертификаты и другие параметры, которые можно использовать в этапах конвейера. В разделе post настраивается обработка ошибок и уведомления.

В более сложном CD-конвейере могут быть дополнительные этапы, такие как синхронизация данных, архивирование информационных ресурсов, установка обновлений и патчей. CI/CD-инструменты обычно поддерживают плагины. Например, у Jenkins есть более 1500 плагинов для интеграции со сторонними платформами, для расширения пользовательского интерфейса, администрирования, управления исходным кодом и сборкой.

При использовании CI/CD-инструмента разработчики должны убедиться, что все параметры сконфигурированы вне приложения через переменные окружения. CI/CD-инструменты позволяют устанавливать значения этих переменных, маскировать пароли и ключи учетных записей, а также настраивать их во время развертывания для конкретного окружения.

Также в CD-инструментах присутствуют дашборды и отчетность. В случае сбоя сборки или поставки они оповещают об этом. При интеграции CD с системой контроля версий и agile-инструментами облегчается поиск изменений кода и пользовательских историй, вошедших в сборку.

Реализация CI/CD-конвейеров с Kubernetes и бессерверными архитектурами

Многие команды, использующие CI/CD-конвейеры в облаках используют контейнеры, такие как Docker, и системы оркестрации, такие как Kubernetes. Контейнеры позволяют стандартизировать упаковку, поставку и упростить масштабирование и уничтожение окружений с непостоянной нагрузкой.

Есть множество вариантов совместного использования контейнеров, инфраструктуры как код и CI/CD-конвейеров. Подробнее изучить это вы можете в статьях Kubernetes with Jenkins и Kubernetes with Azure DevOps.

Архитектура бессерверных вычислений представляет собой еще один способ развертывания и масштабирования приложений. В бессерверном окружении инфраструктурой полностью управляет поставщик облачных услуг, а приложение потребляет ресурсы по мере необходимости в соответствии с его настройками. Например, в AWS бессерверные приложения запускаются через функции AWS Lambda, развертывание которых может быть интегрировано в CI/CD-конвейер Jenkins с помощью плагина.

CI/CD обеспечивает более частое развертывание кода

Итак, подведем итоги. CI упаковывает, тестирует сборки и оповещает разработчиков, если что-то пошло не так. CD автоматически разворачивает приложения и выполняет дополнительные тесты.

CI/CD-конвейеры предназначены для организаций, которым необходимо часто вносить изменения в приложения с надежным процессом поставки. Помимо стандартизации сборки, разработки тестов и автоматизации развертываний мы получаем целостный производственный процесс по развертыванию изменений кода. Внедрение CI/CD позволяет разработчикам сосредоточиться на улучшении приложений и не тратить силы на его развертывание.

CI/CD является одной из DevOps-практик, поскольку направлена на борьбу с противоречиями между разработчиками, которые хотят часто вносить изменения, и эксплуатацией, требующей стабильности. Благодаря автоматизации, разработчики могут вносить изменения чаще, а команды

эксплуатации, в свою очередь, получают большую стабильность, поскольку конфигурация окружений стандартизирована и в процессе поставки осуществляется непрерывное тестирование. Также настройка переменных окружения отделена от приложения и присутствуют автоматизированные процедуры отката.

Эффект от внедрения CI/CD-конвейеров можно измерить в виде ключевых показателей эффективности (KPI) DevOps. Такие KPI как частота поставки (deployment frequency), время реализации изменений (change lead time) и среднее время восстановления после инцидента (mean time to recovery) часто улучшаются при внедрении CI/CD с непрерывным тестированием. Однако CI/CD — это лишь один из процессов, который может способствовать этим улучшениям. Есть и другие условия для увеличения частоты поставки.

Для начала работы с CI/CD команде разработчиков и эксплуатации необходимо совместно определиться с технологиями, практиками и приоритетами. Команды должны выработать консенсус в отношении правильных подходов к своему бизнесу и технологиям, чтобы после внедрения CI/CD команда постоянно придерживалась выбранных практик.

Отчет по проекту.

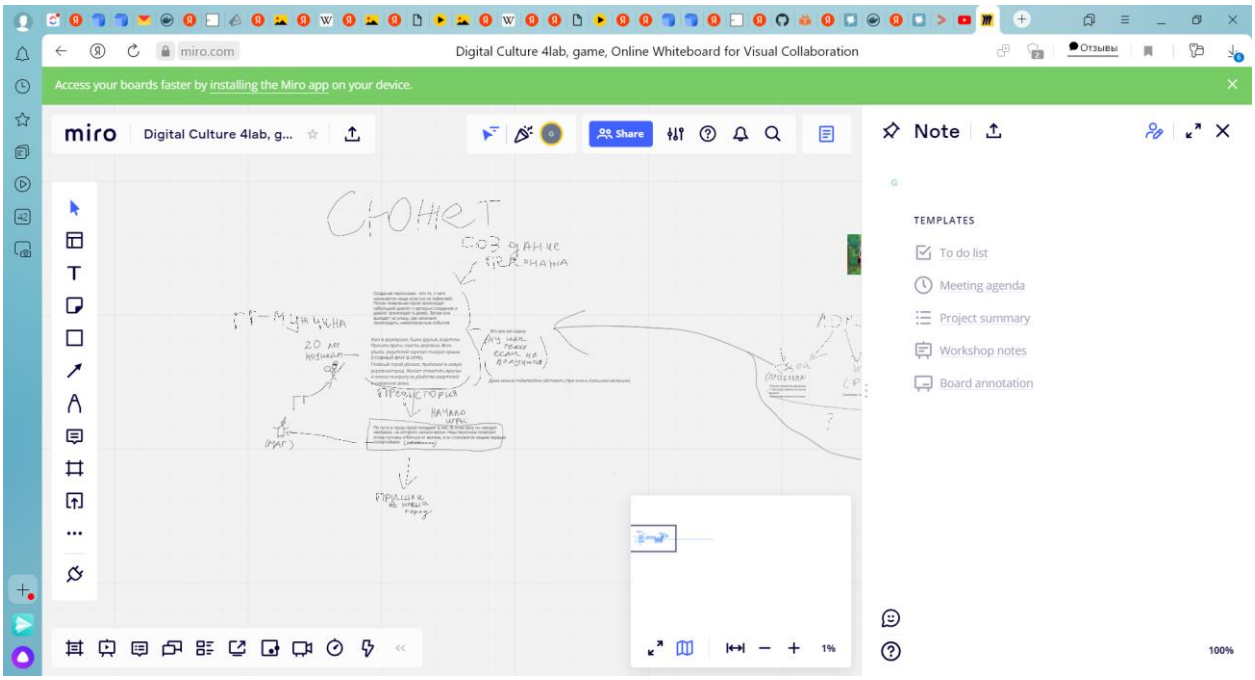
Проект: игра (2d rpg)

Название: Ведьмак

Жанр: пошаговая фэнтези

Игровой движок: rpg maker MV

Для обсуждений и написания сюжета игры была создана доска в Miro в которой производилось написание сюжета геймдизайнерами, а также его корректирование.

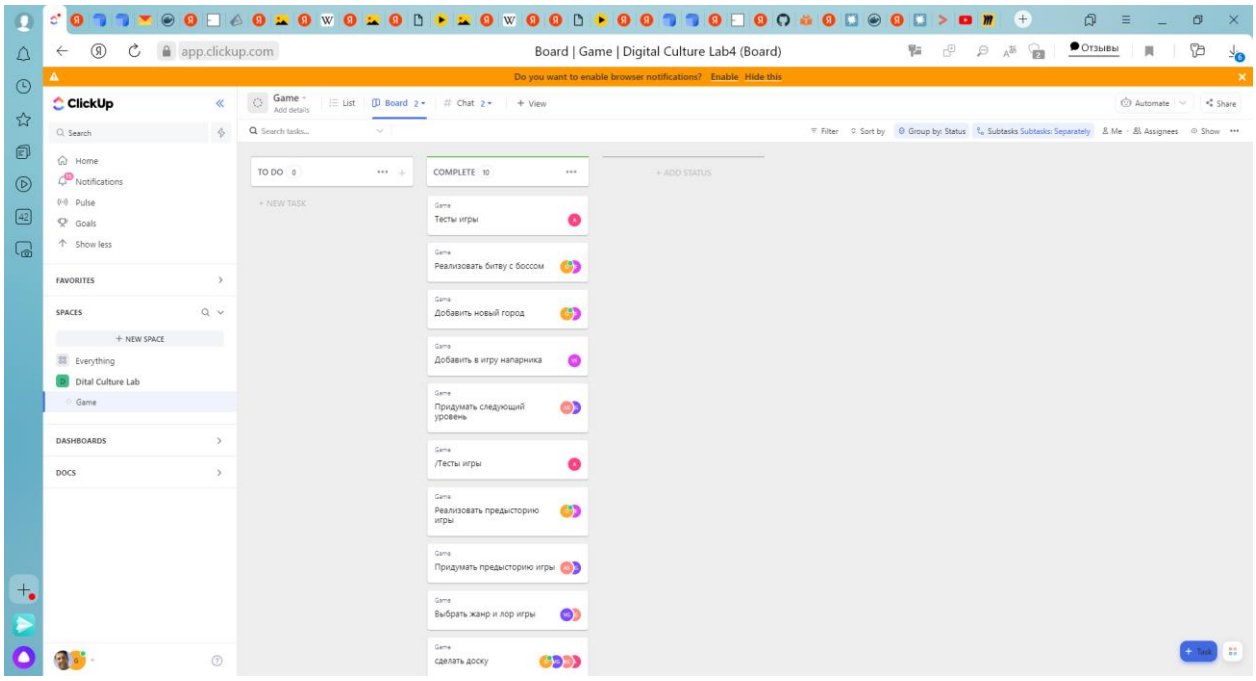


Также в первую очередь были определены роли в команде.

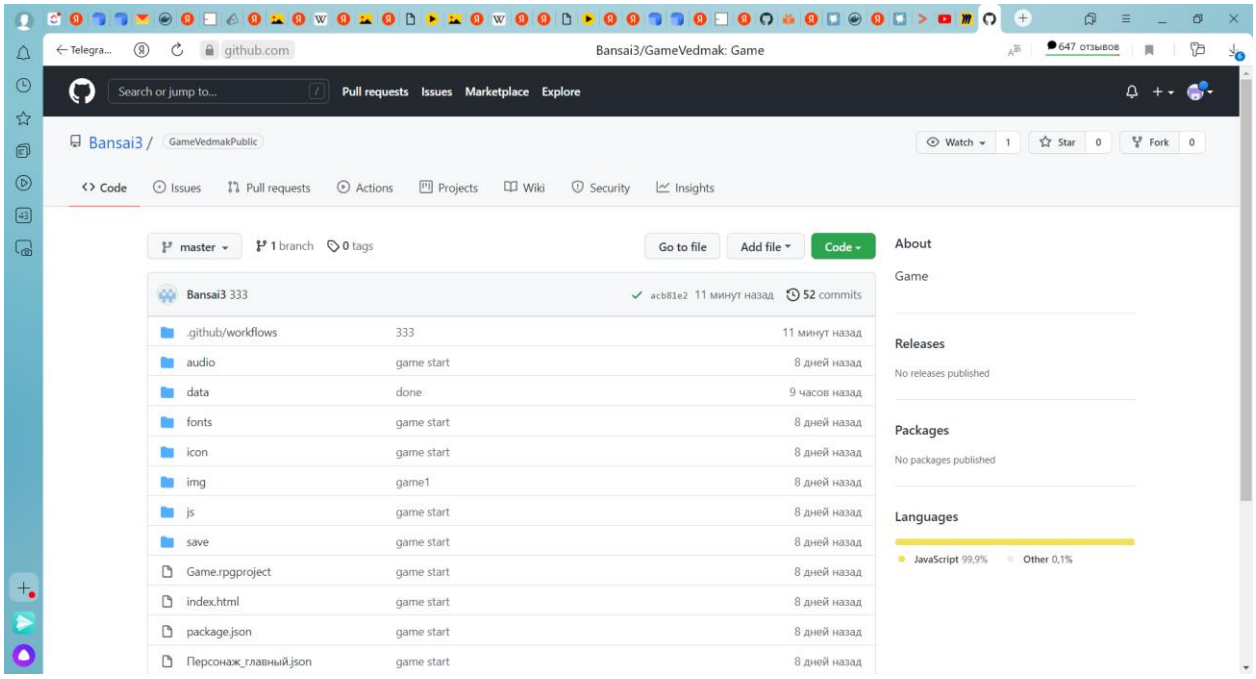
	Георгий Хлучин	Вадим Федотов	Артем Телушкин	Влад Гумбатов	Андрей Соколов

разработчики	+	+	+		
тестировщики	+	+	+	+	+
Гейм дизайнеры				+	+
Создатели моделей				+	+

Для планирования задач была создана доска на Clickup.



Так же был создан репозиторий на котором лежит готовая версия проекта.



Кто что сделал и узнал:

1. Левл дизайнеры (гейм дизайнеры) досконально изучили как работает повышение уровня в рпг: 1) это автолевелинг (плюсы: не позволяет наскучить) (минусы: может убить интерес у игрока на бональных вещах(обесценивает приобретенные навыки)); 2) постоянные уровни (позволяют скорректировать темп игры и поведение игрока)
2. Левл дизайнерами (гейм дизайнерами) были изучены методы работы интуитивного левл дизайна (примеры: 1)специальные объекты окрашиваются в более яркий цвет; 2) можно дать возможность игроку потеряться, чтобы он лучше изучил игру)
3. Левл дизайнерами (гейм дизайнерами) была изучена работа процедурной генерации миров в играх (миры в играх могу генерироваться, например по какому-то специальному алгоритму) (главный плюс - это индивидуальность такого мира)
4. Левл дизайнерами (гейм дизайнерами) был изучен нарративный дизайн в играх (к примеру можно с помощью повествования управлять действиями игрока)
5. Тестировщиками были изучены методы тестирования игр (QA; Фокус-группы; Тестирование UX; Плей тесты; Unit тесты)
6. Разработчиками были изучены способы оптимизации игр
7. Разработчиками были изучены работа микротранзакций и психологические приемы в играх.

Сюжет игры:

Создание персонажа – это то, с чего начинается игра (но не геймплей). После появления героя происходит небольшой диалог с матерью(создание и диалог происходят в доме). Затем они выходят на улицу, где начинают происходить нижеописанные события.

Жил в деревушке, были друзья, родители. Пришли враги, сожгли деревню. Всех убили, родителей зарезал генерал армии (ГЛАВНЫЙ ВРАГ В ИГРЕ). Главный герой убежал, прибежал в новую деревню/город. Желает отомстить врагам и лично генералу за убийство родителей и сожжение дома.

По пути в город герой попадает в лес. В этом лесу он находит человека, на которого напали волки. Наш персонаж помогает этому путнику отбиться от волков, и он становится нашим первым напарником.