**Тема 1. Вступ до курсу. Робочий процес та команда проекту.**

1. **Вступ до курсу**
2. **Сутність проекту та основні його види.**
3. **Робочий процес проекту**
4. **Визначення складу команди**
5. **Огляд практичної роботи.**

Вступ до курсу**.**

В практичній частині курсу розглядаються інструменти Atlassian, такі як Jira та Bitbucket в інтеграції з Sourcetree в контексті налаштування автоматизації процесу CI/CD проекту веб-додатку.

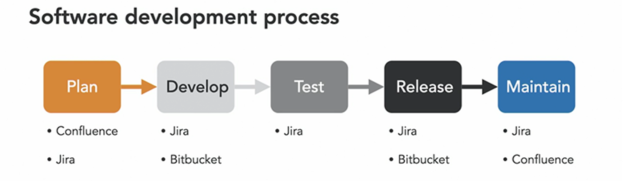
В практичній частині курсу розглянуто кроки по роботі цих інструментів, а також описана технологія налаштування проекту за допомогою Bitbucket рipelines. Кроки pipelines можуть бути налаштовані на запуск автоматично або вручну, залежно від потреб користувача та конкретного етапу розробки, для різних мов програмування та фреймворків. Щоб налаштувати Pipelines, необхідно створити файл bitbucket-pipelines.yml, де описати кроки процесу CI/CD. Після успішного виконання Bitbucket Pipeline, оновлення коду стають доступні в репозиторії, що дає змогу працювати з оновленим кодом. Налаштування автоматизації процесу CI/CD в Jira передбачає автоматичне відстеження пов’язаних процесів збірки та розгортання з проблемами Jira на основі отриманого посилання на ключі проблем в Jira.

На ринку представлено багато інструментів для вирішення таких задач, зокрема найбільш відомими є сервіси від AWS, Atlassian, Azure, кожна з яких має свої переваги і недоліки. Проведені дослідження показують, що саме інтеграція таких інструментів як Jira, Bitbucket та Sourcetree відрізняється простотою використання та конфігурацією, що надають інтуїтивний спосіб організації проектів, управління задачами та налаштування процесу CI/CD.

Інструменти Atlassian, такі як Jira, Bitbucket, є добре встановленими і широко використовуваними на ринку. Вони мають велику спільноту користувачів та розробників, що забезпечує доступність підтримки, документації та розширень, системи легко інтегруються з іншими популярними інструментами розробки та управління проектами. Велика популярність допомагає забезпечити стабільність та надійність системи. Використання цієї системи може бути економічно вигіднішим порівняно з використанням окремих сервісів від AWS або Azure.

Інтеграція Jira, Bitbucket та Sourcetree сприяє впровадженню найкращих практик управління якістю та контролю якості в процес розробки. За допомогою цих інструментів можна встановлювати стандарти кодування, проводити автоматичну перевірку якості, тестування та розгортання програмного забезпечення. Це допомагає знизити кількість помилок, підвищити стабільність продукту та задоволеність клієнтів.

Jira та Bitbucket це найбільш відомі продукти, які належать компанії Atlassian Corporation Plc. Це австралійська компанія з розробки програмного забезпечення для IT індустрії. Більше інформації про продукти Atlassian можна отримати за на офіційному сайті компанії [3]. На рисунку 1 відображено основні продукти Atlassian, які застосовуються в процесі розробки програмного забезпечення.



*Рис. 1. Atlassian Tools в процесі розробки програмного забезпечення*

З представленого на рисунку 1 процесу бачимо, що основними продуктами, які використовуються в системі управління проектами розробки програмного забезпечення є: Jira, Confluence та Bitbucket. Jira – інструмент управління проектами, який дозволяє стежити за завданнями, багами та планувати релізи програмного забезпечення (з 2002 року). Програмне забезпечення Jira може використовуватися як для бізнесу, так і для особистого використання і розраховане переважно на три типи команд: бізнесу, розробки ПЗ та підтримки і обслуговування. Jira також забезпечує зручну інтеграцію з Bitbucket – системою керування версіями (VCS) та репозиторієм Git або Mercurial. Це інструмент для Continuous Integration та спільної роботи розробників над проектами. Bitbucket дозволяє зберігати та керувати кодовими базами, дозволяє створювати та зливати гілки (branches) коду, робити pull-requests та інші дії, які допомагають команді розробників спільно працювати над проектом. Крім того, Bitbucket надає можливість здійснювати рецензування коду, відстежування проблем, організацію спільної роботи над проектами, а також інтеграцію з іншими інструментами.

В даному дослідженні застосовується також графічна програма користувацького інтерфейсу від Sourcetree, яка дозволяє легко працювати з репозиторіями Git. Ця програма, що є клієнтом Git для Windows та Mac, розвиває навики роботи з командним рядком і дає можливість створювати гілки та нові версії, збирати файли, придатні до використання, здійснювати запит для перегляду перед злиттям в головну гілку.

**GitLab, GitHub і Bitbucket** - це всі платформи для керування версіями програмного забезпечення та спільної розробки коду, але вони мають деякі відмінності:

**GitLab:**

GitLab є комбінованою платформою, яка надає вам можливість розміщувати Git-репозиторії (код), керувати завданнями, вести CI/CD, слідкувати за змінами в коді, розробляти інші проекти та багато іншого.

Ви можете встановити GitLab на своєму власному сервері або використовувати хмарну версію GitLab.com.

Є безкоштовна версія GitLab Community Edition, а також платні версії з розширеними можливостями для корпоративних потреб.

**GitHub:**

GitHub в основному спеціалізується на розміщенні та спільній розробці Git-репозиторіїв. Він надає засоби для співпраці, обговорення коду, відстеження проблем, створення власних версійних гілок та багато іншого.

GitHub надає безкоштовні облікові записи для відкритих джерел коду та платні плани для приватних проектів та команд.

**Bitbucket:**

Bitbucket також дозволяє розміщувати Git-репозиторії, проте він відзначається більшою спеціалізацією на засобах спільної розробки для команд.

Окрім Git, Bitbucket також підтримує репозиторії, засновані на Mercurial.

Є безкоштовні плани для обмеженої кількості користувачів та платні плани з розширеними функціями.

Вибір між цими платформами залежатиме від ваших конкретних потреб і обставин. Кожна з них має свої переваги та функції, які можуть відповідати вашому проекту або команді краще за інші.

AWS та Microsoft Azure також надають послуги для неперервної інтеграції та неперервної доставки (CI/CD).

Проекти в галузі розробки та впровадження нових технологій в автомобілебудуванні

**Розробка системи автономного керування автомобілем:** Створення алгоритмів, розробка сенсорної системи, інтеграція з механічною частиною автомобіля.

**Створення системи моніторингу та діагностики стану автомобіля:** Розробка програмного забезпечення для збору даних з датчиків, аналізу даних та виявлення потенційних проблем.

**Розробка системи адаптивного круїз-контролю:** Створення алгоритмів для підтримки безпечної дистанції до попереднього автомобіля та автоматичного регулювання швидкості.

**Розробка системи допомоги при паркуванні:** Створення алгоритмів для автоматичного паркування автомобіля, розробка сенсорної системи.

**Розробка системи електромобіля:** Створення силової установки, системи управління батареями, інтеграція з іншими системами автомобіля.

Проекти в галузі управління виробничими процесами в автомобільній промисловості

**Оптимізація виробничих ліній:** Аналіз існуючих процесів, розробка пропозицій щодо їх вдосконалення, впровадження нових технологій (наприклад, роботизація, автоматизація).

**Впровадження системи управління якістю:** Розробка системи контролю якості продукції, встановлення стандартів якості, проведення аудиту.

**Управління логістичними процесами:** Оптимізація маршрутів доставки, управління запасами, впровадження систем відстеження вантажів.

**Управління проектами модернізації виробництва:** Планування, організація та контроль проектів з модернізації обладнання, технологічних процесів.

Проекти в галузі сервісного обслуговування автомобілів

**Розробка системи онлайн-запису на сервісне обслуговування:** Створення веб-платформи або мобільного додатку для зручності клієнтів.

**Впровадження системи управління запасами запчастин:** Оптимізація рівня запасів, автоматизація замовлень.

**Розробка системи дистанційної діагностики автомобілів:** Створення програмного забезпечення для збору даних з автомобіля та проведення діагностики на відстані.

Проекти в галузі управління автопарком

**Розробка системи управління автопарком:** Створення програмного забезпечення для відстеження геолокації транспортних засобів, планування маршрутів, контролю витрат палива.

**Оптимізація маршрутів руху транспорту:** Використання алгоритмів для розрахунку оптимальних маршрутів з урахуванням різних факторів (пробки, відстань, час доставки).

**Управління витратами на паливо:** Аналіз витрат, розробка заходів щодо їх зниження.

**Для прикладу пропонується вибір такого проєкту:**

Розробка системи дистанційної діагностики автомобілів: **Створення програмного забезпечення для збору даних з автомобіля та проведення діагностики на відстані.**

Такий проект не тільки дозволить студентам застосувати свої знання з програмної інженерії та автомобільної електроніки, але й сприятиме розвитку інноваційних рішень в автомобільній галузі.

**Зазвичай, система дистанційної діагностики автомобіля складається з таких компонентів:**

**Апаратне забезпечення:** Встановлюється в автомобіль і збирає дані з різних датчиків (швидкість, оберти двигуна, температура, тиск тощо).

**Програмне забезпечення:** Обробляє зібрані дані, виявляє потенційні проблеми та формує звіти.

**Мобільний додаток:** Забезпечує зручний доступ до інформації про стан автомобіля для власника або сервісного центру.

**2. Сутність проекту та основні йогов види.**

**Що таке проект? Проект – «Projectus» (лат.) задуманий наперед».**

Розглянемо сутність поняття проект та його визначення в різних літературних джерелах, зокрема, ***проект – «Projectus» (лат.) задуманий наперед».***

***Проект*** - діяльність, спрямована на створення певного продукту чи послуги протягом визначеного терміну та за певних фінансових обмежень. ***Іншими словами****,* це діяльність щодо створення продуктів чи послуг, що здійснюється у логічній послідовності та включає визначені етапи, які входять до життєвого циклу проекту.

***Проект -*** це тимчасова діяльність, яка здійснюється з метою створення унікального продукту чи послуги.

***Проект -*** це сукупність дій, в яких людські, матеріальні і фінансові ресурси організовані для виконання унікального набору робіт певного змісту в умовах обмеженої вартості і часу.

***IT-проект*** - це проект, в рамки якого входять роботи, пов'язані з інформаційними технологіями, які в свою чергу спрямовані на створення, розвиток і підтримку інформаційних систем.

****

**Рис. 1. Основні характеристики проекту**

***Довідково. В проекті мають бути розглянуті всі області знань. Рис.1. Основні характеристики проекту***

***Довідни́к з управлі́ння проєктами (***[***англ.***](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) ***A Guide to the Project Management Body of Knowledge або PMBOK Guide) — довідник, який містить набір*** [***процесів***](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81)***, що зазвичай визнані та забезпечують виконання завдань*** [***управління проєктами***](https://uk.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F_%D0%BF%D1%80%D0%BE%D1%94%D0%BA%D1%82%D0%B0%D0%BC%D0%B8) ***незалежно від галузі та організації, орієнтований на тих, хто складатиме екзамен на сертифікат*** [***PMI***](https://uk.wikipedia.org/wiki/PMI) ***(***[***англ.***](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) ***Project management institute, Інститут управління проєктами).***

***PMBOK складається з дев'яти функцій: менеджменту обсягів, витрат, часу, якості, людських ресурсів, комунікацій, контрактів/постачання, ризиків, проєктної інтеграції. Перші чотири функції (спрямовані на управління цілями) заведено називати основними.***



**Рис. Схема основних зацікавлених сторін проекту**

**Завдання керівника проекту** – пошук компромісів між цілями окремих учасників проекту, оскільки часто вони можуть перебувати у протиріччі.

***Наприклад, ціль постачальника*** – отримання максимального прибутку, ***а ціль компанії***, що реалізовує проект, – мінімізація витрат.

Проекти можуть бути розроблені:

**власними потужностями**

**передані в аутсорсинг**

**придбані як готовий продукт.**

### За типом бізнесу існують такі компанії:

* **Продуктові компанії:**
  + Розробляють власні програмні продукти та апаратні рішення для дистанційної діагностики.
  + Як правило, мають власну команду розробників, маркетингу та продажів.
  + Приклади: Tesla, Ford,
* **Аутсорсингові компанії:**
  + Виконують розробку програмного забезпечення та апаратних рішень на замовлення інших компаній.
  + Не мають власного продукту, але володіють необхідними компетенціями для його створення.
  + Приклади: багато українських IT-компаній, які спеціалізуються на розробці програмного забезпечення для автомобільної галузі.
* **Аутстаф компанії:**
  + Надають своїх фахівців в оренду іншим компаніям для участі у проектах.
  + Можуть виконувати як частину проекту, так і весь проект повністю.
* **Системні інтегратори:**
  + Інтегрують різноманітні програмні та апаратні рішення в єдину систему.
  + Зазвичай працюють з продуктами різних виробників.

**2. Робочий процес проекту**

Наш проект: **Створення мобільного додатку для збору даних з автомобіля та проведення діагностики на відстані для нового сервісного центру в США.**

Вихідні умови:

Ти влаштовуєшся на роботу менеджером в аутсорс-компанію CreatorTY, яка займається мобільною розробкою. Онбординг[[1]](#footnote-1) щойно закінчився. Для тебе це перший і новий проект.

Тобі потрібно добре знати, як влаштовано розробку, яку методологію вибрати, як побудувати процеси та здати проєкт у визначений термін. Але на цьому сфера компетенцій не закінчується.

Проджект повинен бути в курсі життєвого циклу продукту: розуміти, на якому етапі розвитку він знаходиться, хто його користувачі та яка зараз ситуація на ринку. Це допоможе зрозуміти мету та завдання продукту, донести їх до команди.

На аутсорсі ці знання знадобляться не лише для загальної картини, а й для спілкування із замовником. Проджект стає посередником між бажаннями замовника та командою, якій потрібні чіткі вимоги.

Суть замовлення. Сервісний цент замовив у нас додаток. Він хоче, щоб будь-який клієнт міг отримати інформацію про технічний стан свого автомобіля прямо у своєму телефоні.

*Користувач* - це власники легкових автомобілів, що не мають гарантії на сервісне обслуговування від виробника Авто. Мета — дати їм інструмент для діагностики технічного стану автомобіля.

Потрібен MVP[[2]](#footnote-2), тому що замовник лише перевіряє свою ідею. Робоча назва — DriveEasy. Отже, ми робимо додаток під назвою DriveEasy.

*Про цифри*: приблизно через 4 місяці ми випускаємо MVP *(мінімально життєздатний продукт з базовим мінімальним функціоналом)* і протягом першого місяця збираємо базу з 10 00 активних користувачів. Користуючись інструментами шерингу хочемо подвоїти кількість користувачів на другий місяць.

У перші півроку пробуємо продукт на США в одному штаті, а протягом наступних 6 місяців будемо локалізувати на інші штати. Це основні пункти у розвитку DriveEasy y.

Орієнтовні вартість: $110 000

Монетизація у перші 6 місяців не планується. DriveEasy важливо набрати першу базу лояльних користувачів (1 000 користувачів у перший місяць та 80 00 - на початок 6 місяця). Як тільки замовник визначить product-market fit (*Product-Market Fit (PMF) - важлива концепція в роботі з новим продуктом. Дослівно PMF перекладається як «відповідність продукту ринку»)* і цінність DriveEasy для користувача, він почне поступово впроваджувати платне розширення місця для контенту користувача в хмарному сховищі.

Джон Клайд (власник сервісного центру) мало розуміється на процесі розробки, просто горить бажанням зробити щось класне для своєї індустрії. Ви, яке прожект спілкуватимемося через його менеджера Олівію. З нею можна домовитися про деталі, терміни та вартість проєкту.

Олівія надала список фіч.

Мобільний додаток на iOS. У ньому будуть наступні фічі:

**Для користувача:**

**Реєстрація та профіль:**

Створення особистого кабінету з можливістю вказання даних про автомобіль (марка, модель, рік випуску, VIN-код).

Збереження історії обслуговування, включаючи дати, описи виконаних робіт, вартість та рекомендації майстра.

**Підключення автомобіля:**

Інструкції з підключення OBD-ІІ адаптера (якщо потрібно) або використання вбудованих систем автомобіля для збору даних.

Перевірка сумісності автомобіля з додатком.

**Моніторинг стану автомобіля:**

Відображення основних параметрів автомобіля в режимі реального часу (швидкість, витрата палива, температура двигуна тощо).

Отримання сповіщень про можливі несправності та необхідність проведення технічного обслуговування.

**Запит діагностики:**

Можливість ініціювати детальну діагностику автомобіля за допомогою кількох натискань.

Вибір конкретних систем для діагностики (двигун, коробка передач, гальмівна система тощо).

**Отримання звітів:**

Формування детальних звітів про стан автомобіля з рекомендаціями щодо необхідних ремонтних робіт.

Збереження звітів в особистому кабінеті для подальшого використання.

**Запис на сервіс:**

Онлайн-запис на сервіс з вибором зручного часу та майстра.

Можливість прикріпити отриманий звіт про діагностику до запису на сервіс.

**Оплата послуг:**

Інтеграція з платіжними системами для здійснення оплати онлайн.

**Чат зі службою підтримки:**

Швидкий зв'язок з менеджером сервісного центру для отримання консультацій та вирішення питань.

**Для сервісного центру:**

**Панель управління:**

Огляд підключених автомобілів та їхнього стану.

Моніторинг запитів на діагностику та обслуговування.

Управління персоналом та розподіл замовлень.

**База клієнтів:**

Збереження детальної інформації про кожного клієнта та його автомобіль.

Історія взаємодії з клієнтом.

**База знань:**

Доступ до технічної документації, схем, кодів помилок.

Можливість додавання власних нотаток та рекомендацій.

**Звіти та аналітика:**

Генерація звітів про популярність послуг, ефективність роботи майстрів, фінансові показники.

Додаткові функції:

**Геолокація:** Визначення місцезнаходження автомобіля в режимі реального часу.

**Нагадування:** Створення нагадувань про планове технічне обслуговування, заміну масла тощо.

**Програма лояльності:** Накопичення бонусів та знижок для постійних клієнтів.

**Інтеграція з іншими системами:** Можливість інтеграції з системами управління запасами, обліку та іншими.

Замовник не надав специфікації, а значить писати вимоги до проєкту та збирати команду тобі доведеться самостійно.

Наш проект:

Проджект отримує Vision проєкту від Олівії.

**Vision проєкту**— це велика ідея, яку замовник реалізує через продукт. Зазвичай цей документ складається лише з однієї сторінки. У ньому містяться основні тези про продукт: яким він буде, яку мету переслідує бізнес, хто ваш користувач. Завдяки Vision усі учасники проєкту розуміють, в якому напрямку рухатися та якого результату потрібно досягти.

Vision допомагає побудувати беклог і визначити пріоритети завдань: що потрібно зробити зараз, а на які фічі закласти час в кінці. Vision потрібно обов'язково обговорити із замовником. Він уточнить, які фічі важливіші для продукту.

У Vision від Олівії є інформація про ринок, користувачів, монетизацію та ідейні цілі DriveEasy y. Також у документі перераховані фічі, які необхідно реалізувати.

**Завдання.** Для пріоритетизації визначених фіч, які будуть включені в MVP, розглянемо їх типи та відповідно їх просортуємо.

## Типи продуктових фіч

## 

**1. Must-have фічі** - це "життєво необхідна" функціональність. Наприклад, у пошуковій системі обов'язково має бути пошук, у соціальній мережі — можливість додавати друзів. У фітнес-додатку – програми тренувань та інструкції з виконання вправ, у фотододатку – інструменти для редагування фотографій.

Без Must have фічі, користувачі не зможуть задовольнити свої потреби. я функціональність обов'язково має бути у першій версії. Але цього замало, щоб перемогти на конкурентному ринку.

**2. Performance фічі** - функціональність, завдяки якій продукт відрізняється від інших конкурентів. Наприклад, для пошукача це може бути швидкість пошукового алгоритму, для соціальної мережі розумна стрічка новин. Для фітнес-додатку – ступінь деталізації програми тренувань, можливість трекінгу кроків, розрахунок калорій, плани харчування. Для фотододатка — набори фільтрів, унікальні інструменти редагування фотографій тощо. Для AirPods це миттєве з'єднання з iPhone.

Через Performance фічі продукт відбудовується від конкурентів, формуючи свою унікальну ціннісну пропозицію.

**3. Wow фічі** - це функціональність, що викликає у користувача приємні емоції. Наприклад, для фітнес-додатку - спеціальні тренування від зірок спорту (як Кріштіану Роналду в Nike Training Club). Для фотопрограми — функціональність, яка автоматично обробляє фотографії, виходячи з уподобань користувача. Такі фічі роблять твій продукт особливим.

Wow фічі - це вишенька на торті.

Користувач завжди вибирає продукт за деякими параметрами. Це і є Performance фічі. Тому більшість зусиль команди має бути приділена саме цим фічам. Користувач замовить таксі в тому додатку, в якому машину подають найшвидше.

При цьому не можна забувати про Must-Have фічі. Якщо користувач не отримуватиме головну цінність, то йому буде не до Performance фіч. Добре, що AirPods можуть моментально підключатися до iPhone. Але в цьому немає жодного сенсу, якщо динамік не працює.

Wow фічі покращують сприйняття продукту загалом, що призведе до лояльності та рекомендацій. Ніхто не розповідатиме друзям про просто нормальний продукт.

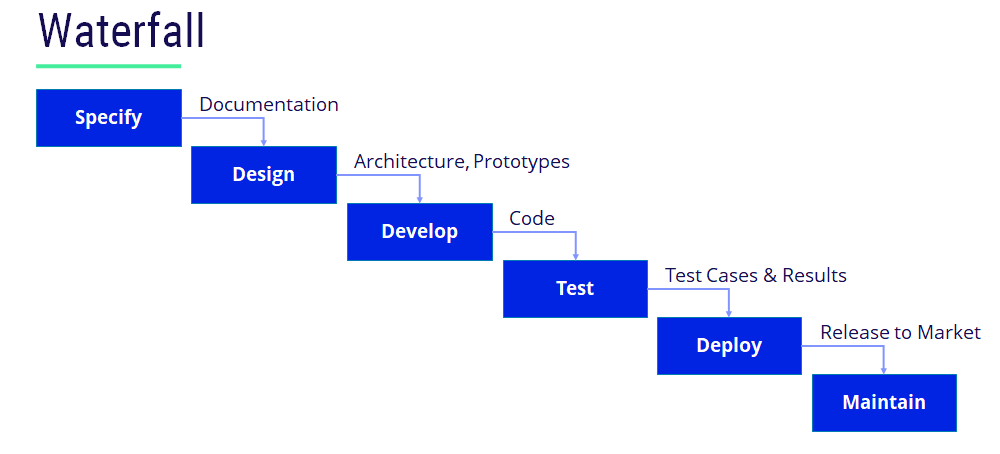
Дом.**Завдання.** Оберіть 3 фічі, які варто включтти до MVP

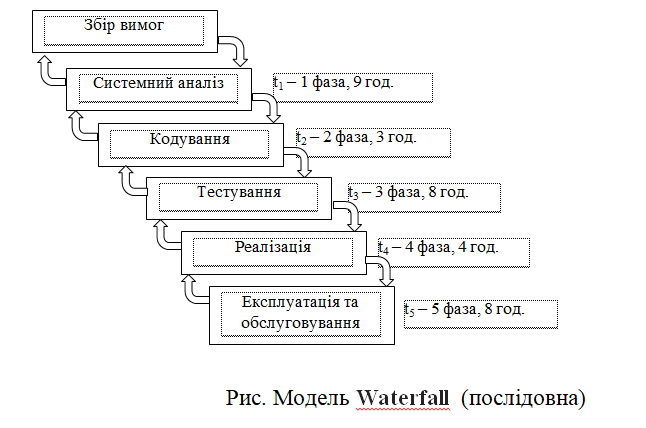
**Повернемось до процесів.**

Процеси або фази життєвого циклу розробки продукту можуть бути організовані послідовно або паралельно.

Модель життєвого циклу — це структура, що складається із процесів, робіт та задач, які включають в себе розробку, експлуатацію і супровід програмного продукту; охоплює життя системи від визначення вимог до неї до припинення її використання. На сьогодні найбільшого розповсюдження набули такі моделі:

1) модель – **Waterfall** (каскадна, послідовна), в основі якої покладено початкове планування проекту (як правило, кожна фаза належить окремому колективу) (рис). Однією з перших з'явилась каскадна модель, в якій кожен етап роботи виконується лише раз. На кожному етапі робота виконується настільки ретельно, щоб потреби повертатись до попереднього не виникало. Результат виконання кожного етапу, перед передачею в наступний, піддається верифікації.





Послідовна характеризується тим, що кожна наступна операція (задача, функція) починається тільки після її повного завершення на попередній операції (стадії процесу або задачі).

Паралельно-послідовна модель характеризується тим, що задачі (функції продукту) передаються на наступну стадію процесу (спринту) таким чином, щоб забезпечити безперервність всього процесу розробки програмного продукту.

При запарелелюванні процесів, графік побудови технологічного процесу розробки проекту буде мати наступний вигляд (рис.2).

|  |  |  |  |
| --- | --- | --- | --- |
| № задачі | задача | тривалість, дні | порядковий номер задачі, яка передує поточній |
| 1 | планування та моделювання проекту | 20 | 1 |
| 2 | аналіз вимог проекту | 10 | 2 |
| 3 | прототипування проекту | 20 | 1,2 |
| 4 | кодування фічі 1 | 5 | 3 |
| 5 | кодування фічі 2 | 10 | 3 |
| 6 | кодування фічі 3 | 7 | 3 |
| 7 | тестування | 2 | 4,5,6 |

**V-модель є варіацією моделі водоспаду у V- формі**, складеної навпіл на найнижчому рівні розкла­дання (рис.2.2). Ліва нога V-форми представляє еволю­цію вимог користувача до все більш дрібних компо­нентів через процес декомпозиції і визначення. Права нога являє інтеграцію та перевірку системних компо­нентів у послідовні рівні впровадження та складання. Вертикальна вісь зображує рівень розкладання від системного рівня, у верхній частині, до найнижчого рівня деталізації на рівні компонентів внизу.

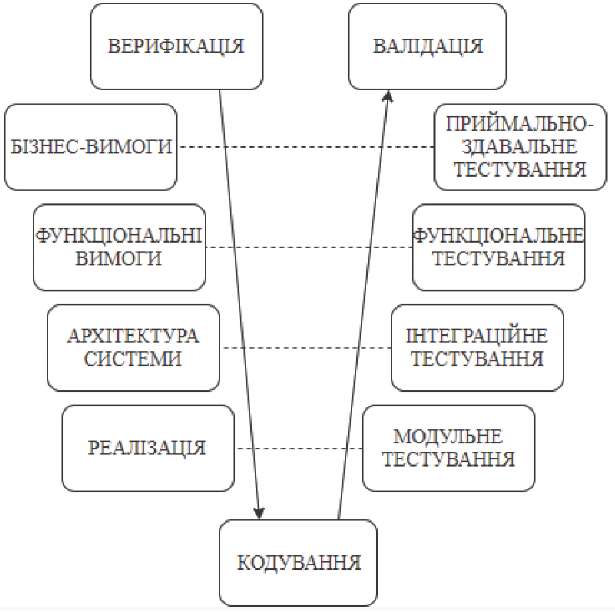


Рис.2.2. V-модель

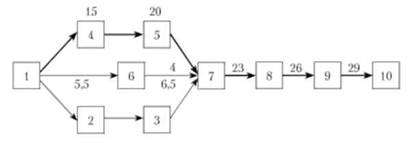
Що стосується великих, складних проектів, як правило не повязаниз розробкою ПЗ, то при роботі з ними найчастіше використовуються більш складні способи планування. Одним з таких методів є метод мережевого планування.

**Мережеве планування** – це такий підхід до проектного планування, який дозволяє побачити окремі етапи (роботи) виконання складного проекту і отримати уявлення про шляхи досягнення цілей.

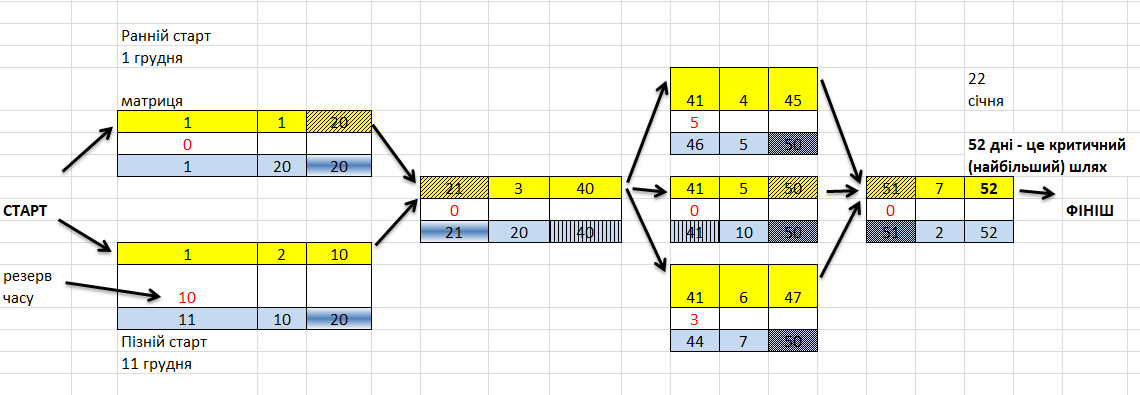
Шлях від початкового до кінцевого події називається ***повним.***

***Критичний шлях*** *–* найдовший за часом шляху від початку виконання завдання до його завершення; на графіку критичний шлях позначається жирними стрілками





*Рис..* Мережевий графік



Тільки після процесу аналітичного планування можна переходити до вибору інструментів роботи по проекту.

1. **Інкрементна модель (Incremental model)**

Відповідно до інкрементної моделі (англ. Increment - збільшення, прирощення) програмне забезпечення розробляється з лінійною послідовністю стадій, але в кілька інкрементів (версій). Таким чином поліпшення продукту проходить заплановано весь час, поки життєвий цикл розробки ПЗ не завершиться.

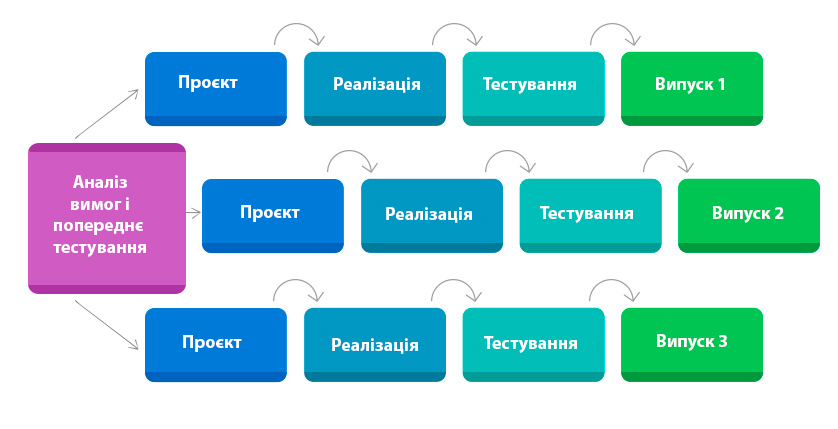
Вимоги до системи визначаються на самому початку роботи, після чого процес розробки проводиться у вигляді послідовності версій, кожна з яких є закінченим і працездатним продуктом.

Плюси:

* замовник може дати свій відгук щодо кожної версії продукту;
* є можливість переглянути ризики, які пов'язані з витратами і дотриманням графіка;
* звикання замовника до нової технології відбувається поступово.

Мінуси:

* функціональна система повинна бути повністю визначена на початку життєвого циклу для виділення ітерацій;
* при постійних змінах структура системи може бути порушена;
* терміни здачі системи можуть бути збільшені через обмеженість ресурсів (виконавці, фінанси).



1. **Гнучка модель (Agile model)**

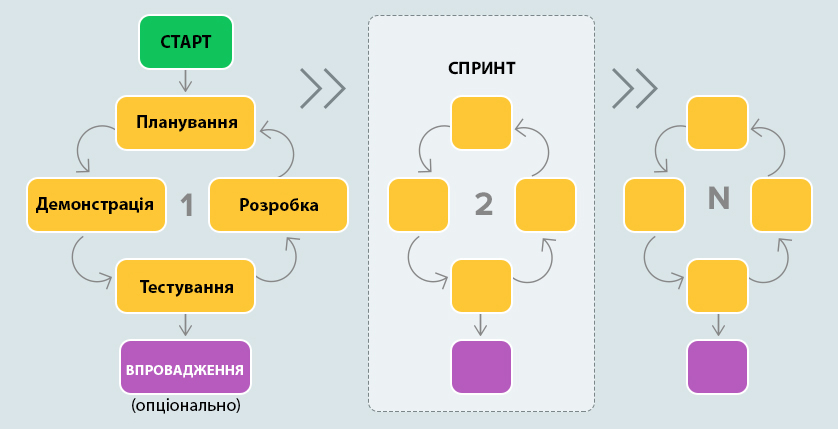
Являє собою сукупність різних підходів до розробки ПЗ. Включає серії підходів до розробки програмного забезпечення, орієнтованого на використання ітеративної розробки (в Scrum ітерації називаються спринтами), динамічне формування вимог і забезпечення їхньої реалізації в результаті постійної взаємодії всередині самоорганізованих робочих груп, що складаються з фахівців різного профілю. Окрема ітерація являє собою мініатюрний програмний проєкт. Однією з основних ідей Agile є взаємодія всередині команди і з замовником напряму.

Плюси:

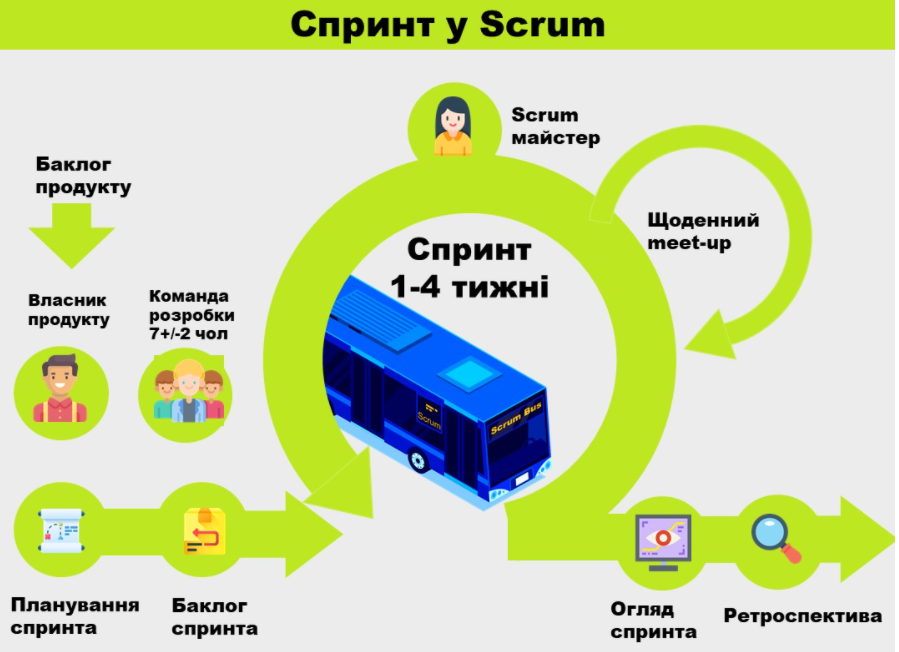
* швидке прийняття рішень завдяки постійним комунікаціям;
* мінімізація ризиків;
* полегшена робота з документацією.

Мінуси:

* велика кількість мітингів і обговорень, що може збільшити час розробки продукту;
* складно планувати процеси, так як вимоги постійно змінюються;
* рідко використовується для реалізації великих проєктів.



Якщо замовник не має специфікації й незрозуміло, як робити проєкт. А ще якщо проєкт зовсім новий та не має аналогів застосовуються гнучкі практики, однією з яких є Scrum. Наприклад, зробити клон соцмережі Twitter можна і за Waterfall — у нього зрозуміла функціональність. Але написати двигун рекомендацій, як у Tiktok, — вже інше завдання, тоді буде Scrum



Ти – менеджер проекту і застосовуєш гнучкий підхід:

Не всі раніше працювали по Scrum, тому потрібно чітко проговорити правила ведення проекту з командою. Визнач довжину спринтів та встанови періодичність зустрічей: коли відбуватимуться стендапи, грумінги, ретро та інші Scrum-події.

Першочергові завдання краще озвучити прямо на першій зустрічі, щоб одразу команда займалася проєктом: налаштовувала сервери або проєктувала архітектуру.

Методологія: Scrum: спринти завдовжки 2 тижні.

Багтрекер: Jira

Канали зв'язку: Telegram, e-mail, meet

Графік зустрічей:

Планування спринту – перший день спринту, 11.00

Рев'ю спринту - останній день спринту, 16.00

Ретроспектива – другий день спринту, за результатами попереднього спринту 19.00

Стендап – щодня, 11.00

Грумінг (пріоритеризація та перегляд, оцінка) беклогу - за потребою.

Зустрічі з клієнтом – за домовленістю.

Приклад робочого флоу[[3]](#footnote-3)





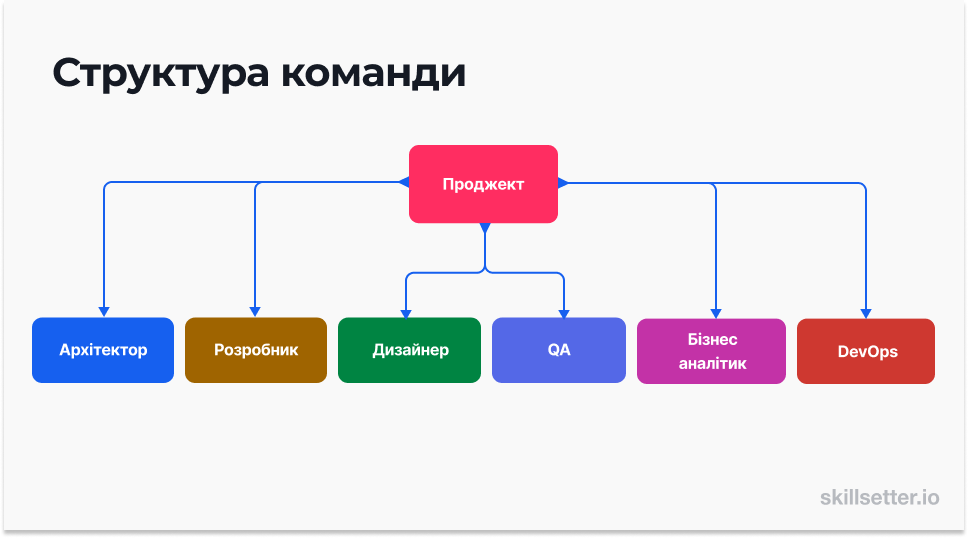
Рис. Робочий процес DriveEasy

Зазвичай у розробці фічі чотири фази: дизайн, розробка, тестування та постачання. Такі етапи роботи почнуться лише через 4 місяці, після випуску MVP. А доти, наприкінці кожного спринту готуватимемо робочий білд, який можна протестувати.

Одна ітерація – 1 фіча, демонстрація результату у вигляді того самого білда, влаштовуємо демо, і ти залишаєш фідбек.

**3. Визначення складу команди**

Команда розробки на аутсорсі може складатися із різних спеціалістів. У будь-якому проєкті у вас будуть: розробники, QA-інженери та проджект-менеджер. Залежно від специфіки також можуть бути бізнес-аналітик, архітектор програмного забезпечення, UI/UX-дизайнер, DevOps і розробник баз даних.



На склад проєктної команди впливають кілька факторів: рейти, терміни проєкту, технології та обсяг роботи.

**Рейти** — це оплата робочого часу вашої команди (N $ / годину розробки). Рейти прописуються у контракті. Чим вони вищі, тим більше висококваліфікованих інженерів можна залучити.

**Терміни проєкту**, як і рейти, прописуються у контракті. Якщо часу на розробку замало, в команду можна набрати більше інженерів і виконувати кілька завдань паралельно.

**Технології проєкту** впливають на вибір спеціалістів у команду. Там можуть бути fullstack-розробники або спеціалісти конкретної галузі — iOS-розробники, Android-розробники, джавісти та інші.

**Обсяг роботи** впливає на кількість розробників та їх експертизу. На невеликий та зрозумілий проєкт можна взяти 1-2 програмістів початкового рівня, а на комплексну та складну роботу краще найняти більш досвідчених інженерів.

Ти – менеджер проекту:

Розумієш, що у будь-якому разі ці фахівці повинні розумітися на технологіях твого проєкту. У CreatorTY ти створюєш мобільний додаток на iOS. Отже, тобі знадобляться мобільні розробники, знайомі з фреймворками та інструментами цієї платформи. Вони повинні знати мову Swift[[4]](#footnote-4) чи objective C[[5]](#footnote-5).

Ти звертаєш до наставника.

Наставник:

Сеньйорам потрібні складні завдання, а у нас є і прості, які теж хтось має робити.

Мідлам можна довірити зрозумілі завдання, які вони вже вирішували або бачили в інших проєктах.

А з простими завданнями типу «зробити інтерфейс для запиту доступу до даних на пристрої» впорається й джун.

Якщо фіч багато — то треба людей багато. Не потрібно ставити десять розробників на одну невелику фічу. Вони тільки гальмуватимуть один одного. А ось у месенджері типу Telegram — основних функцій мало: повідомлення, налаштування користувача і голосовий чат.

На трьох розробників потрібний приблизно один QA-інженер.

Оптимальний сетап (склад) команди:

25% сеньйорів, 25% джунів та 50% мідлів. Але тут є нюанси.

**Команда проекту:**

Оптимальний склад команди залежатиме від масштабу проекту, його складності та наявних ресурсів. Ось приблизний склад команди, який можна розглядати:

**1. Продакт-менеджер**

Визначає стратегію розвитку продукту.

Формулює вимоги до продукту.

Координує роботу команди.

Відповідає за взаємодію з клієнтами та замовниками.

**2. Програмісти**

Frontend-розробник: Створює інтерфейс користувача мобільного додатку.

Backend-розробник: Розробляє серверну частину, яка відповідає за обробку даних, взаємодію з базами даних та іншими системами.

Мобільний розробник: Розробляє мобільний додаток під конкретні платформи (iOS, Android).

**3. Тестувальники**

Перевіряють функціональність додатку.

Виявляють помилки та недоліки.

Забезпечують високу якість продукту.

**4. Дизайнер інтерфейсу**

Створює дизайн інтерфейсу мобільного додатку.

Розробляє прототипи.

Забезпечує інтуїтивність та зручність використання додатку.

**5. Спеціаліст з баз даних**

Розробляє структуру баз даних.

Забезпечує ефективне зберігання та обробку даних.

**6. Спеціаліст з безпеки**

Забезпечує захист даних користувачів.

Проводить аудит безпеки.

Розробляє заходи для запобігання кібератак.

**7. Спеціаліст з автомобільної електроніки**

Консультує щодо протоколів зв'язку, датчиків та інших автомобільних систем.

Допомагає у розробці алгоритмів аналізу даних.

**8. Спеціаліст з машинного навчання (опціонально)**

Розробляє алгоритми машинного навчання для аналізу даних та прогнозування несправностей.

Пропозиція:

Три розробники — сеньйора, мідла та джуна. Всю роботу веди паралельно, щоби не затягувати з термінами.

На QA — сеньйора на 0.25 ставки (проєкт простий, тому задач для сеньйора буде замало для повної ставки) та мідл - на повну ставку. Тестуватимуть написане розробниками, не поспішаючи. Наприклад, потрібно буде тестувати продуктивність, один мідл не впорається. А фічі Picture in Picture і хмарне сховище будуть з нюансами: точно знадобиться досвідчений фахівець. Загалом, сеньйор наглядатиме за мідлом і виконуватиме функцію QA-ліда, вона на цьому проєкті особливо важлива.

Є хмарне сховище — потрібен бекендер мідл-рівня не на повний день.

Твоє рішення:

Отже, початковий склад команди може бути наступний:

**1 PM менеджер - ти**

**1 дизайнер**, щоб відмалювати UI всієї програми.

**3 iOS розробника**: 1 сеньйор (Тімлід команди розробки) 1 мідл та 1 джун. Команді потрібен хоча б один сеньйор, щоб реалізувати фічу Picture in Picture та попередити можливі труднощі. Мідл на рутинні завдання та джун — на найпростіші.

**1.25 QA-інженерів**: 1 сеньйор на 0.25 та 1 мідл.

**Бекендер** мідл-спеціаліст на 0.5 ставки, тому що в проєкті є фіча «Хмарне сховище», що потребує роботи з серверами.



**В додатку А** відображена таблиця кадровий склад ІТ-компанії, у якій ти працюєш. У ній перераховані всі фахівці CreatorTY та їхня зайнятість. Подивися, хто зараз вільний, та забронюй на свій проєкт. По таблиці:

0 — це вільний, 0.5 — це зайнятий на пів ставки, а 1 — повністю у проєкті.

Дивись на дати проєктів, у яких беруть участь фахівці. Хтось, може, уже закінчує роботу і буде ось-ось вільний.

## Завдання

Відкрий таблицю та вибери фахівців, які підходять тобі по завантаженню та кваліфікації. (**Ця Угода набирає чинності з понеділка, 20.04.2024 року, а проект планується завершити в середу, 1 вересня 2024 року).** Стеж, щоб люди зі списку були повністю вільні чи могли прийти до тебе на пів ставки саме у цей період.

**Завдання на передпроектний період (7 днів)**

Перший спринт розпочнеться за 7 днів, і за цей час потрібно встигнути підготувати всі процеси.

Наприкінці спринту ми зберемося на спільну зустріч з усією командою та покажемо результати роботи замовнику. Швидше за все, це буде інтерактивний прототип у Figma. Ми обговоримо результат та можливі коригування на наступний спринт. Якщо все добре — почнемо розробку першої фічі.

Результат аналізу кадрового складу компанії:

*Наза етапу - адміністрування проекту*

Виконавці :

Ти – менеджер проекту:

*Запланувати всі зустрічі п*о Scrum

*Завести проект у багтрекері* у Jira та дати доступ всій команді та замовнику.

*Завести базу знань* у Jira

*Вибрати скоуп завдань та провести грумінг беклогу для першого спринту (залучення тімліда)*

Дизайнер Саша Мілхауз

*Визначити, які екрани програми малювати в першу чергу (залучення тімліда - розробника сеньйора Кері Вулф і менеджера)*

*Створити UХ-схему програми*

Тімлід команди розробки. Розробник сеньйор Кері Вулф

*Намалювати схему архітектури*

Розробник мідл Даррен Рачинскі

*Налаштувати систему випуску білдів*

Розробник джуніор Джо Батлер

*Визначитися з правилами для пул реквестів[[6]](#footnote-6) та гітфлоу[[7]](#footnote-7).*

Інженер QA мідл Нік Стенджерс

*Вибрати тип тестової документації і зберігання (*тест кейси).

QA-інженер сеньйор 0.25 Ед Сендман

*Опціонально: допомогти проджекту з написанням вимог до проекту*

Бекендер 0.5 Джо Батлер

*Налаштувати сервери* для тестування, розробки, демо та продакшену.

**Додаток А**

**кадровий склад ІТ-компанії**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Працівник | спеціалізація | Рівень кваліфікації | ПРОЕКТИ | ЗАйНЯТі з | **ЗАйНЯТі ДО** | ЗАНЯТІСТЬ |
| Джастин Гринберг | Бекенд розробник | сеньйор | Guitar dreamer | 05.02.2022 | **05.02.2023** | 0,25 |
| Анджела Джонс | Бекенд розробник | мідл | Guitar dreamer | 05.02.2021 | **05.02.2023** | 0,5 |
| Питер Гласс | Бекенд розробник | сеньйор | Face recognition | 05.03.2022 | **06.08.2022** | 1 |
| Джо Батлер | Бекенд розробник | мідл | Junga app | 09.09.2021 | **25.06.2023** | 0,5 |
| Синди Питерхофф | Android-розробник | сеньйор | Guitar dreamer | 05.02.2021 | **05.02.2023** | 1 |
| Бритни Голдс | Android-розробник | мідл | - |  |  | 0 |
| Патрик Доллстай | Android-розробник | мідл | - |  |  | 0 |
| Елена Джонс | Android-розробник | джуніор | - |  |  | 0 |
| Лили Свифт | iOS-розробник | мідл | Guitar dreamer | 05.02.2021 | **05.02.2023** | 1 |
| Еван Тимми | iOS-розробник | джуніор | Guitar dreamer | 05.02.2021 | **05.02.2023** | 1 |
| Уилл Симпсон | iOS-розробник | сеньйор | Face recognition | 05.03.2022 | **06.08.2022** | 0,5 |
| Даррен Рачински | iOS-розробник | мідл | Face recognition | 05.03.2022 | **15.04.2023** | 1 |
| Кери Вульф | iOS-розробник | сеньйор | Junga app | 09.09.2021 | **15.04.2023** | 1 |
| Патрик Харауей | iOS-розробник | джуніор | - |  |  | 0 |
| Хелен Стенджерс | QA-інженер | сеньйор | Guitar dreamer | 05.02.2021 | **05.02.2023** | 0,5 |
| Хлоя Спирс | QA-інженер | мідл | Guitar dreamer | 05.02.2021 | **05.02.2023** | 1 |
| Брендон Ковальски | QA-інженер | джуніор | Guitar dreamer | 05.02.2021 | **05.02.2023** | 1 |
| Алекс Джонсон | QA-інженер | мідл | Face recognition | 05.03.2022 | **06.08.2022** | 0,5 |
| Кери Денсон | QA-інженер | мідл | Face recognition | 05.03.2022 | **06.08.2022** | 0,5 |
| Джанет Еллиот-Свансон | QA-інженер | джуніор | Junga app | 09.09.2021 | **30.04.2022** | 1 |
| Ед Сендман | QA-інженер | сеньйор | Junga app | 09.09.2021 | **30.04.2023** | 1 |
| Ник Стенджерс | QA-інженер | мідл | Junga app | 09.09.2021 | **15.04.2023** | 0,5 |
| Даррен Соланж | UI-дизайнер | - | Guitar dreamer | 05.02.2021 | **05.02.2023** | 1 |
| Кристи Ноулз | UI-дизайнер | - | Face recognition | 05.03.2022 | **06.08.2022** | 0,25 |
| Штефан Лоренц Зоргнер | UI-дизайнер | - | Face recognition | 05.03.2022 | **06.08.2022** | 0,5 |
| **Саша Миллхаус** | **UI-дизайнер** | **-** | **Junga app** | **09.09.2021** | **15.04.2023** | **1** |

1. Це процес інтеграції нового співробітника в компанію і колектив, який починається ще під час найму і триває до півроку, а іноді – і того більше, в залежності від програм адаптації та корпоративної культури роботодавця [↑](#footnote-ref-1)
2. MVP, мінімально життєздатний продукт ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) *Minimum viable product* — MVP) — продукт з мінімальним функціоналом, який можна дати користувачам для використання. Використовується для тестування ідей у розробці програм з мінімальними затратами ресурсів. Зазвичай реалізується шляхом вдалого планування релізів, коли у перший реліз входять лише основні функції (або одна найважливіша), що дає можливість користувачам уже почати працювати, не очікуючи остаточної версії. [↑](#footnote-ref-2)
3. це послідовність робочих операцій. Включає загальні завдання команди, кроки по їх реалізації, комунікацію зі стейкхолдерами, інструменти, постановку цілей і результати роботи [↑](#footnote-ref-3)
4. **Swift** — [багатопарадигмова](https://uk.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) [компільована](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%96%D0%BB%D1%8C%D0%BE%D0%B2%D0%B0%D0%BD%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) [мова програмування](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), розроблена компанією [Apple](https://uk.wikipedia.org/wiki/Apple) для того, щоб співіснувати з [Objective C](https://uk.wikipedia.org/wiki/Objective_C) і бути стійкішою до помилкового коду. [↑](#footnote-ref-4)
5. **Objective-C** — ("Обджектів Сі") [рефлективна](https://uk.wikipedia.org/w/index.php?title=%D0%A0%D0%B5%D1%84%D0%BB%D0%B5%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F&action=edit&redlink=1), [високорівнева](https://uk.wikipedia.org/wiki/%D0%92%D0%B8%D1%81%D0%BE%D0%BA%D0%BE%D1%80%D1%96%D0%B2%D0%BD%D0%B5%D0%B2%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) [об'єктно-орієнтована](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%E2%80%99%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) [мова програмування](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) загального призначення, розроблена у вигляді набору розширень стандартної [С](https://uk.wikipedia.org/wiki/%D0%A1%D1%96_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)). [↑](#footnote-ref-5)
6. це запит на вливання змін із гілки-фічі в основну гілку вихідного репозиторію. Наприклад, розробник закінчив зміни у своїй гілці та створює пул реквест. Так він повідомляє інших, що закінчив роботу і потрібно перевірити зміни в його гілці, щоб влити їх в основну [↑](#footnote-ref-6)
7. це модель робочого процесу [Git](https://git-scm.com/) (Системи контролю версій). Вона визначає, які гілки потрібно створити і як проводити їхнє злиття (мерджити) [↑](#footnote-ref-7)