**Тема 1. Вступ до курсу. Робочий процес та команда проекту – 4 год.**

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

Вступ до курсу**.**

**Теми курсу:**

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

Тема 2. Основи роботи в Jira,

Тема 3. Робота в Confluence, Bitbucket, Sourcetree

Тема 4. Робота з контрактами в IT

Тема 5. Моделі розробки проектів на основі Agile

Тема 6. Робота над вимогами до продукту

Тема 7. Оцінка та розподіл задач, планування спринту

Тема 8 Робота з бюджетом проекту

Тема 9. Якість, тестування і випуск продукту

Тема 10. Юніт економіка та маркетингові метрики в ІТ.

Тема 11. Інвестування ІС

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.

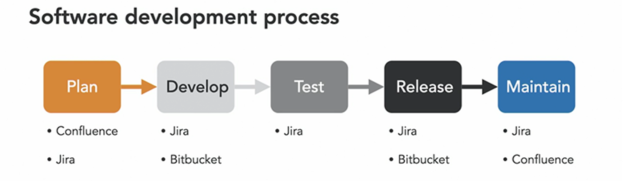
В практичній частині курсу розглянуто кроки по роботі цих інструментів, а також описана технологія налаштування проекту за допомогою Bitbucket рipelines. Кроки pipelines можуть бути налаштовані на запуск автоматично або вручну, залежно від потреб користувача та конкретного етапу розробки, для різних 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).

**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.

ІТ-проекти можна класифікувати за всіма класичними класифікаційними ознаками, хоча специфіка галузі накладає свої особливості.

За функціональним призначенням (орієнтованість продуктів IT-проектів на напрямки діяльності замовника): виробничі; технологічні; фінансові; дослідні; маркетингові; з управління персоналом; з управління проектами; ігрові; комбіновані.

За видом автоматизованих процесів: основні і допоміжні; технологічні та офісні; управлінські; аналітичні; транзакційні; реального часу; з тим чи іншим акцентом на обчислювальну обробку; передача даних; організація зберігання; обробка медіа-контенту; забезпечення безпеки.

За рівнем впливу розробки інтерфейсу на проект:

– низький - результат розробки інтерфейсу має незначний вплив на оцінку проекту в цілому, займає невеликий відрізок часу в життєвому циклі проекту;

– середній - результат розробки інтерфейсу має значний вплив на оцінку проекту в цілому, займає значний відрізок часу в життєвому циклі проекту;

– високий - результат розробки інтерфейсу має критично важливе вплив на оцінку проекту в цілому, займає значний відрізок часу в життєвому циклі проекту, може вплинути на прийняття рішення про закриття проекту.

**Так, *за видом продукту розрізняють такі ІТ-проекти як:***

**1. Проект по створенню ІТ-системи, що включає поєднані між собою ІТ пристрої і програмне забезпечення. *ПЗ в свою чергу поділяється на:***

**1.1. Системне (базове) програмне забезпечення слугує для організації й ефективної роботи комп'ютера та охоплює:**

**1) операційні системи (ОС) основною функцією яких є керування ресурсами і процесами обчислювальних систем;**

*Ось декілька прикладів операційних систем:*

*Windows - операційна система, створена компанією Microsoft, що використовується на більшості персональних комп'ютерів у всьому світі.*

*MacOS - операційна система, розроблена компанією Apple, яка використовується на комп'ютерах Mac.*

*Linux - операційна система з відкритим вихідним кодом, яка є популярною серед розробників програмного забезпечення та використовується у серверному середовищі.*

*Android - операційна система для мобільних пристроїв, розроблена компанією Google, що використовується на більшості смартфонів і планшетів.*

*iOS - операційна система для мобільних пристроїв, розроблена компанією Apple, що використовується на iPhone, iPad і iPod Touch.*

**2) мережеве ПЗ, призначене для керування спільними ресурсами в розподілених обчислювальних системах;**

*Ось декілька прикладів базового мережевого програмного забезпечення:*

*Операційна система мережевих серверів, наприклад, Windows Server або Linux Server, яка дозволяє керувати мережевою інфраструктурою, обслуговувати користувачів та забезпечувати безпеку мережі.*

*Маршрутизатори та комутатори, наприклад, Cisco IOS, Juniper JunOS або HP Comware, які дозволяють забезпечувати маршрутизацію даних в мережі та керувати комутацією даних між пристроями.*

**3) сервісні програми, до яких належать файлові менеджери (наприклад, Total Commander 7.55); утиліти, тобто системні програми для виконання службових функцій (антивіруси, архіватори, програми для обслуговування дисків тощо). Ці програми створюють додаткові зручності під час роботи користувача на комп'ютері;**

**1.2. Прикладне програмне забезпечення — це програми, що призначені для реалізації конкретних задач опрацювання даних, які користувач розв’язує в ході своєї діяльності. Наприклад,  текстові процесори (Microsoft Word, WordPad, Блокнот, та ін.); табличні (Microsoft Excel), для графіки (Corel Draw, PageMaker, Adobe Photoshop) програми створення презентацій (Microsoft Power Point, та інші.**

*Ось декілька прикладів прикладного програмного забезпечення (ППЗ):*

*Офісні пакети (Microsoft Office, Google Docs)*

*Редактори зображень та графіки (Adobe Photoshop)*

*Мультимедійні плеєри (Windows Media Player, iTunes)*

*Браузери (Google Chrome)*

*Поштові клієнти (Microsoft Outlook, Gmail)*

*Клієнти соціальних мереж (Facebook, Instagram, Twitter)*

*Месенджери (WhatsApp, Telegram, Facebook Messenger)*

*Програми для роботи з базами даних (Microsoft Access, MySQL, Oracle)*

**Веб-сайти, веб-сервіси, веб-додатки та мобільні додатки відносяться до прикладного програмного забезпечення.**

*Веб-сайт - це колекція веб-сторінок, які можуть містити текст, зображення, відео, аудіо та інші елементи контенту.*

*Веб-додаток - це програмне забезпечення, яке працює в браузері і надає користувачам можливість взаємодіяти зі службами та функціями через Інтернет. Веб-додатки можуть виконувати різні завдання, такі як обробка оплат, збереження даних або доступ до різних інформаційних ресурсів. Прикладами веб-додатків є електронна пошта, онлайн-гри, соціальні мережі та інтернет-банкінг.*

*Веб-сервіс - це програмна система, яка надає стороннім програмам доступ до своїх функцій та послуг через мережу Інтернет.*

***Ось кілька прикладів веб-сервісів:***

***Amazon Web Services (AWS) - платформа хмарних обчислень, яка надає набір веб-сервісів для обробки даних, зберігання, машинного навчання, безпеки та багатьох інших функцій.***

**Jira - це програмне забезпечення для управління проектами, яке надається в форматі веб-сервісу.**

**Програмно-технічні комплекси як вид прикладного програмного забезпечення. ІТ-проект, який передбачає створення продукту з вбудованим програмним забезпеченням.**

**ПТК – це сукупність технічного обладнання, програмного забезпечення та організаційних заходів для реалізації конкретного напряму діяльності, дозволяє будувати на його базі системи (наприклад, інформаційні) різних масштабів і складності.**

*Наприклад:*

*"Прозоро" - комплекс електронних систем для публічних закупівель.*

*"Дія" - комплекс для забезпечення надання адміністративних послуг.*

*"Картка пацієнта" - інформаційно-аналітична система для автоматизації роботи медичних закладів.*

**ІТ-проекти зі створення окремих модулів, блоків, додатків, дизайну, кошторису і т.д. Це матеріали, роботи та послуги, пов’язані з окремими етапами проектних робіт, також відносяться до прикладного ПЗ.**

Отже, **інформаційна система** є комплексом технологій, програмного забезпечення, обладнання, баз даних та інфраструктури, призначених для збору, обробки, збереження та обміну інформацією в межах підприємства.

Інформаційні системи підприємств створюються з метою полегшення управління бізнес-процесами, прийняття рішень та забезпечення ефективного функціонування підприємства.

Орієнтуючись на різні міжнародні стандарти, ІС можна визначити як сукупність таких складових частин:

1) «система баз даних: база даних (БД) разом iз системою управління базами даних (СУБД);

2) прикладне ПЗ;

3) персонал;

4) організаційно–методичне (нормативне) забезпечення;

5) технічні засоби.

Проекти ІС можуть бути розроблені:

**власними потужностями**

**передані в аутсорсинг**

**придбані як готовий продукт.**

Більшість ІС бізнес-структур в Україні стосуються таких складових структур:

1. **Загальнофірмова інфраструктура**

Інформаційна система (ІС) загальнофірмової інфраструктури компанії - це комплекс технічних, програмних і організаційних засобів, які об'єднуються для забезпечення обробки, зберігання, передачі та управління інформацією всієї компанії або організації. Ця система включає в себе різні компоненти та ресурси, які дозволяють організації ефективно використовувати інформацію для своєї діяльності.

Включають:

Інформаційну базу даних

ІТ-інфраструктура:

Програмні засоби:

Комунікаційна інфраструктура:

Інструменти аналізу та звітності:

Безпека та захист інформації:

2. **Транзакційні системи** ефективно обслуговують бізнес-процеси. Така система зазвичай відповідає за обробку рутинних операцій, таких як облік продажів, операції з клієнтами, фінансові транзакції тощо. Наприклад для авіакомпаній це бронювання, продаж білетів, реєстрація, розклад літаків та екіпажем, а також обслуговування повітряних суден.

3. **Стратегічні системи** – програми, які забезпечують конкурентні переваги. Приклад, для авіакомпанії це програма для ідентифікації цінних клієнтів, щоб вони могли бути забезпечені диференційованим сервісом для підвищення їх лояльності - це приклад стратегічної ІС системи. Мета ІС – покращення продаж підвищення конкурентоспроможності або покращення ринкової позиції.

В Україні існують різноманітні види інформаційних систем, які використовуються в різних галузях, включаючи бізнес, органи влади, освіту, охорону здоров'я та інші сфери. Ось деякі з найпоширеніших видів інформаційних систем в Україні:

**Інформаційні системи управління підприємствами (ERP)**: Використовуються для автоматизації бізнес-процесів, обліку ресурсів, фінансів та управління цілими підприємствами.



**Рис. Прикладом такої системи для великих підприємств є програмний продукт класу ERP Парус-Підприємство 8.**

**Системи управління взаємодією з клієнтами (CRM)**: Використовуються для ведення бази клієнтів, взаємодії з ними та покращення обслуговування клієнтів.

**Системи управління логістикою та постачанням (SCM)**: Використовуються для оптимізації логістичних процесів, складського обліку та постачання товарів.

**Системи управління проектами (Project Management)**: Системи для управління проектами дозволяють компаніям планувати та виконувати стратегічні проекти, враховуючи ресурси, строки та бюджет.

В сучасному світі більшість проектів стосуються побудови ІС, які інтегрують в собі декілька вже існуючих ІС, а не побудові принципово нових, а також обміну даними між різними формами ІС та гаджетами.

В даному курсі для прикладу обрано стратегічну ІС, що забезпечує покращення конкурентоспроможності замовника. Замовник – відомий музикант. Який бажає створити додаток-відеоредактор для молодих талантів.

Веб-додаток-відеоредактор може бути включений в поняття інформаційних систем, але це залежить від рівня складності, функціональності та масштабу цього додатку. Інформаційні системи, як правило, включають в себе збір, зберігання, обробку та надання інформації для підтримки прийняття рішень в організаціях або підприємствах. Якщо цей веб-додаток здатний збирати аналітичні дані про користувачів, надавати доступ до ресурсів для навчання або створення власного вмісту, а також надавати статистику про використання додатку, то він може мати елементи інформаційної системи, які допомагають управляти цим додатком.

**2. Робочий процес ІТ-проекту**

Наш проект: стосується створення стратегічної ІС, яка забезпечить власнику конкурентну перевагу.

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. Відомий музикант Джон Клайд замовив у нас додаток-відеоредактор. Він хоче, щоб будь-який виконавець-початківець міг створити крутий кліп прямо у своєму телефоні. Його головне завдання і, мабуть, перевага — допомагати молодим талантам.

*Користувач* - це молодий музикант від 16 до 35 років, який цікавиться створенням кліпмейкінгу. Відповідно, цільова аудиторія — музиканти-початківці. Вони намагаються потрапити в індустрію, але не мають доступу до професійної техніки. Мета — дати їм інструмент для створення та поширення якісних музичних кліпів.

*Конкуренти*. Нині на ринку багато простих редакторів. Вони підходять для постінгу відео в соцмережі, але не більше. Є **Splice** зі складнішим функціоналом. Це наш головний конкурент.

Потрібен MVP[[2]](#footnote-2), тому що замовник лише перевіряє свою ідею. Робоча назва — Filmy. Отже, ми робимо додаток під **назвою Filmy** для молодих музикантів, які не мають доступу до дорогого обладнання, але є величезне бажання створювати музичні кліпи відмінної якості.

*Про цифри*: приблизно через 4 місяці ми випускаємо MVP *(мінімально життєздатний продукт з базовим мінімальним функціоналом)* і протягом першого місяця збираємо базу з 10 000 активних користувачів. Користуючись інструментами шерингу хочемо подвоїти кількість користувачів на другий місяць.

У перші півроку пробуємо продукт на США, а протягом наступних 6 місяців будемо локалізувати на Європу та Китай. Це основні пункти у розвитку Filmy.

Орієнтовні вартість: $110 000

Монетизація у перші 6 місяців не планується. Filmy важливо набрати першу базу лояльних користувачів (10 000 користувачів у перший місяць та 80 000 - 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 Filmy для користувача, він почне поступово впроваджувати платне розширення місця для контенту користувача в хмарному сховищі.

Джон Клайд 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 відеоредактор на iOS. У ньому будуть наступні фічі:

- Додавання відео з галереї

- Логін через Facebook, Apple та Google

- Створення нового проекту

- Обрізка відео

- Накладання колірних фільтрів

- Додавання музики

- Додавання тексту

- Конвертування в інші формати

- Зміна формату відео

- Публікація відео у соцмережі

- Таймлайни

- Picture in Picture

- Хмарне сховище для файлів.

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 Filmy. Також у документі перераховані фічі, які необхідно реалізувати.

**Filmy: vision and scope** від Олівії

Filmy — це мобільний відеоредактор, який дозволяє створювати яскраві та незабутні відеоролики, зберігати їх у хмарному сховищі, ділитися з іншими та створювати дуети.

Ми робимо Filmy для молодих музикантів, які не мають доступу до дорогого обладнання, але є величезне бажання створювати музичні кліпи відмінної якості. Наш користувач - це молодий музикант від 16 до 35 років, який цікавиться створенням кліпмейкінгу. Він невідомий виконавець чи мікроінфлюєнсер, що створює відеоконтент для свого музичного портфоліо.

Наш головний конкурент це Splice, а також інші мобільні відеоредактори. Головна відмінність Filmy від конкурентів — максимально мінімалістичний інтерфейс, а також чіткий орієнтир на музикантів-початківців.

Позиціонування Filmy – це мобільність, легкість та професіоналізм. Ми хочемо донести до користувача, що займатися музикою можна в будь-якій точці планети, у будь-який час і використовуючи різні інструменти. Ми повинні чітко транслювати, що зняти свій кліп може кожен без особливих навичок.

Filmy орієнтований насамперед на ринок США і через три місяці після випуску MVP планується вихід на європейський та китайський ринки.

Перша версія продукту дозволить користувачам створювати відео, відео дуети та ділитися ними онлайн. Далі ми плануємо розширювати функціональність за рахунок великої бібліотеки відео та аудіо ефектів.

Монетизація у перші 6 місяців не планується. Filmy важливо набрати першу базу лояльних користувачів (10 000 користувачів у перший місяць та 80 000 - на початок 6 місяця). Як тільки ми визначимо product-market fit [[3]](#footnote-3)і цінність Filmy для користувача, почнемо поступово впроваджувати платне розширення місця для контенту користувача в хмарному сховищі.

**Фічі.** Користувачі Filmy можуть додавати у свої відео кольорові фільтри, музику та текст. Доступні формати відео: 4:3 та 16:9. Готовий ролик можна експортувати в різній роздільній здатності: 360p, 480p та 720p.

Коли кліп готовий, можна використовувати функцію Picture in Picture та з'єднати два відео в одному. Або просто зберегти результат на телефоні або у хмарі Filmy.

У хмарі Filmy користувач може створювати резервні копії своїх проектів та редагувати їх із будь-якого пристрою.

**Функції Filmy:**

Додавання відео з галереї

Створення нового проекту

Логін

Обрізання відео

Накладання кольорових фільтрів

Додавання музики (ITunes API)

Додавання тексту

Таймлайни: можливість бачити всі доріжки ефектів на одному екрані та змінювати їх тривалість

Хмарне сховище для файлів

Зміна розміру кадру (4:3 або 16:9)

Експорт відео в різній якості (360p, 480p та 720p)

Публікація відео в соцмережі

Picture in Picture

Аналітика в режимі реального часу

**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

Мобільний відеоредактор на iOS. У ньому будуть наступні фічі:

- Додавання відео з галереї

- Логін через Facebook, Apple та Google

- Створення нового проекту

- Обрізка відео

- Накладання колірних фільтрів

- Додавання музики

- Додавання тексту

- Конвертування в інші формати

- Зміна формату відео

- Публікація відео у соцмережі

- Таймлайни

- Picture in Picture

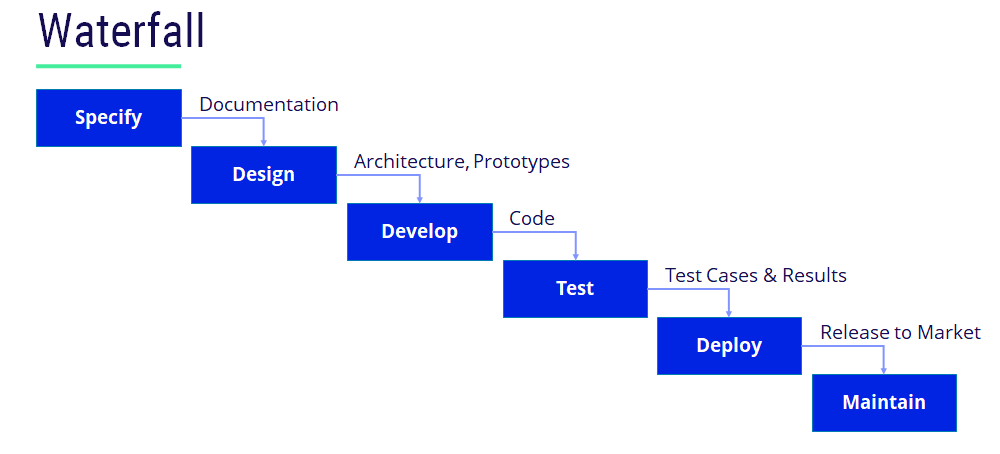
- Хмарне сховище для файлів.

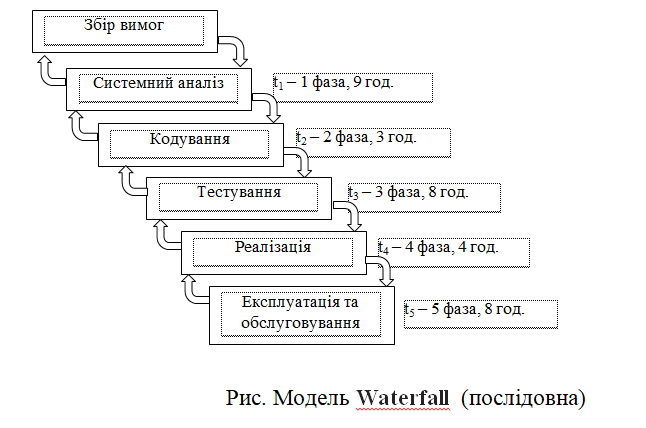
**Повернемось до процесів.**

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).

Завершила лекцію 1

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.

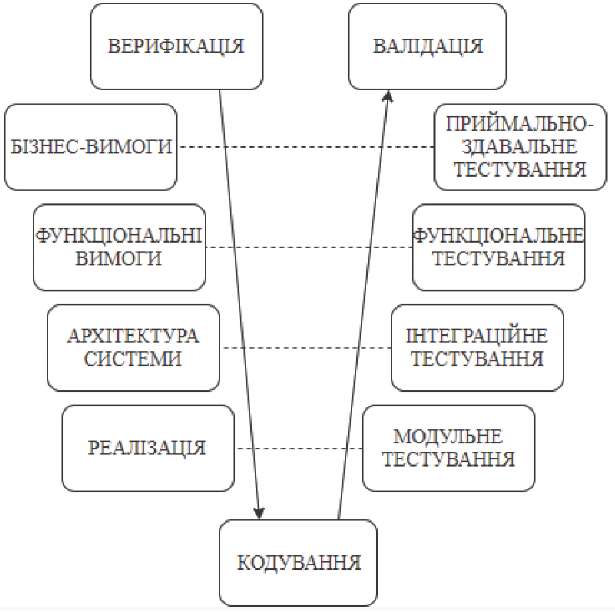


Рис.2.2. V-модель

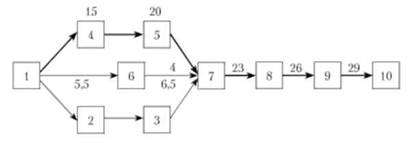
Що стосується великих, складних проектів, як правило не повязаниз розробкою ПЗ, то при роботі з ними найчастіше використовуються більш складні способи планування. Одним з таких методів є метод мережевого планування.

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

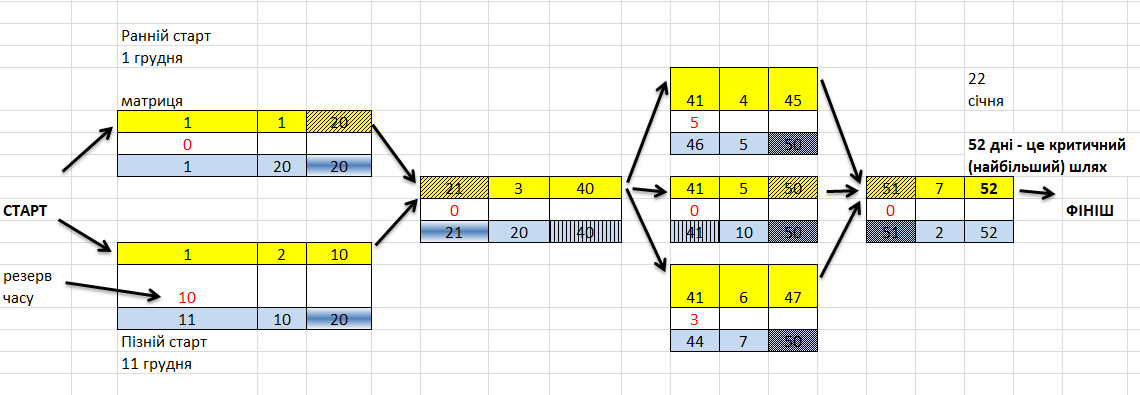
Шлях від початкового до кінцевого події називається ***повним.***

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





*Рис..* Мережевий графік



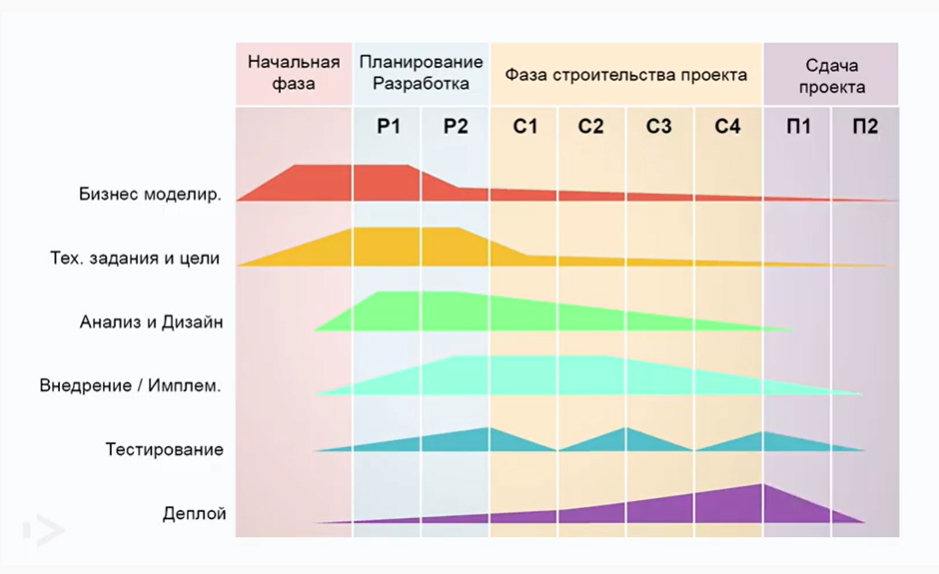


Рис. 2. Приклад паралельного підходу (перехідна модель) робочого процесу

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.

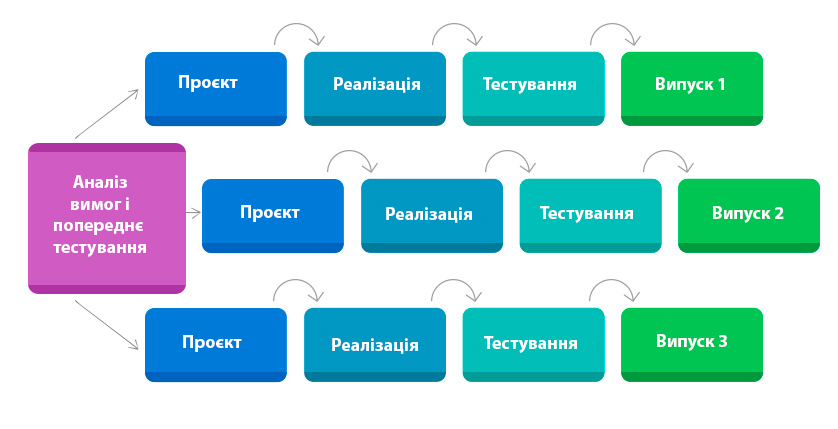
Вимоги до системи визначаються на самому початку роботи, після чого процес розробки проводиться у вигляді послідовності версій, кожна з яких є закінченим і працездатним продуктом.

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. **Спіральна модель**

Розробка ітераціями відображає об'єктивно існуючий спіральний цикл створення системи. Неповне завершення робіт на кожному етапі дозволяє переходити на наступний етап, не чекаючи повного завершення роботи на поточному. При ітеративному способі розробки відсутню роботу можна буде виконати на наступній ітерації. Головне ж завдання — щонайшвидше показати користувачам системи працездатний продукт, тим самим активізуючи процес уточнення і доповнення вимог.

Виходячи з можливості внесення змін, як в процес, так і в проміжний продукт було створено спіральну модель ЖЦ (рис.3).

Внесення змін орієнтоване на задоволення потреби користувачів одразу, як тільки буде встановлено, що створені артефакти або елементи документації не відповідають дійсному стану розробки.

Дана модель ЖЦ допускає аналіз продукту на витку розробки, його перевірку, оцінку правильності та прийняття рішення про перехід на наступний виток або повернення на попередній виток для доопрацювання на ньому проміжного продукту.

Відмінність цієї моделі від каскадної полягає в можливості багато разів повертатися до процесу формулювання вимог і до повторної розробки версії системи з будь-якого процесу моделі.

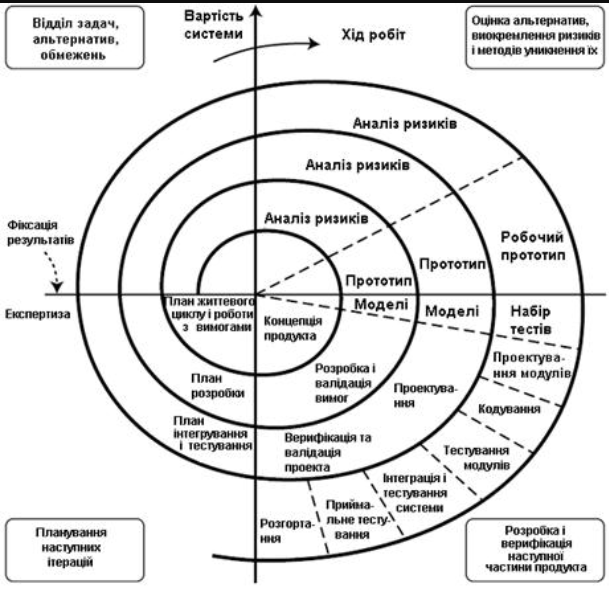


Рис.3. Спіральна модель ЖЦ розробки програмних систем

Для програмного продукту така модель не дуже підходить з декількох причин. По-перше, висловлення вимог замовником носить суб'єктивний характер, вимоги можуть багаторазово уточнюватися протягом розробки ПС і навіть після завершення та випробовування, і часом може з'ясуватися, що замовник «хотів зовсім інше». По-друге, змінюються обставини та умови використання системи, тому загальновизнаним законом програмної інженерії є закон еволюції, який сформулюємо так: кожна діюча ПС з часом потребує внесення змін або виводиться з експлуатації.

При необхідності внесення змін до системи на кожному витку з метою отримання нової версії системи обов'язково вносяться зміни в заздалегідь зафіксовані вимоги, після чого повертаються на попередній виток спіралі для продовження реалізації нової версії системи з урахуванням усіх змін.

1. **Гнучка модель (Agile model)**

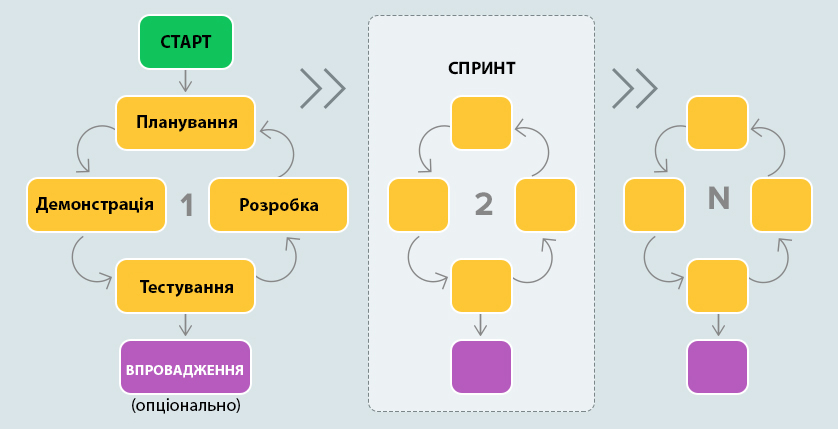
Являє собою сукупність різних підходів до розробки ПЗ. Включає серії підходів до розробки програмного забезпечення, орієнтованого на використання ітеративної розробки (в Scrum ітерації називаються спринтами), динамічне формування вимог і забезпечення їхньої реалізації в результаті постійної взаємодії 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:

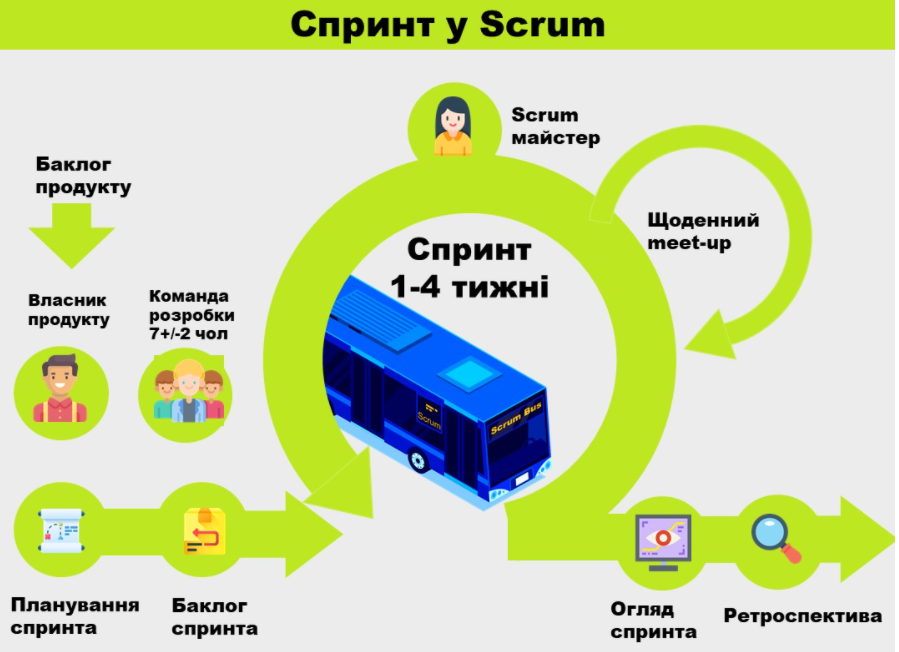
* швидке прийняття рішень завдяки постійним комунікаціям;
* мінімізація ризиків;
* полегшена робота з документацією.

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[[4]](#footnote-4)



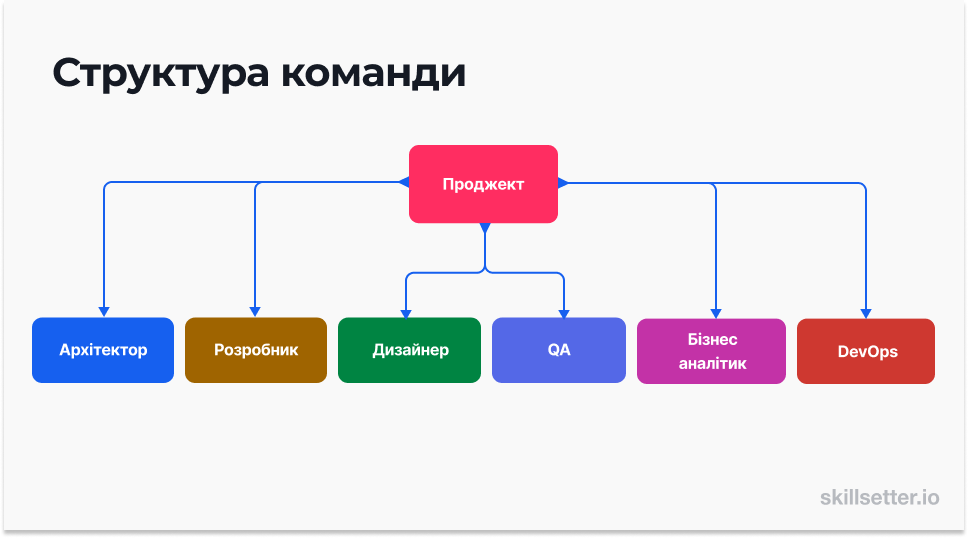


Зазвичай у розробці фічі чотири фази: дизайн, розробка, тестування та постачання. Такі етапи роботи почнуться лише через 4 місяці, після випуску MVP. А доти, наприкінці кожного спринту готуватимемо робочий білд, який можна протестувати.

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[[5]](#footnote-5) чи objective C[[6]](#footnote-6).

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.

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, 20.04.2023 року, а проект планується завершити в середу, 1 вересня 2023 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[[7]](#footnote-7) та гітфлоу[[8]](#footnote-8).*

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. Product-Market Fit (PMF) - важлива концепція в роботі з новим продуктом. Дослівно PMF перекладається як  
   «відповідність продукту ринку». [↑](#footnote-ref-3)
4. це послідовність робочих операцій. Включає загальні завдання команди, кроки по їх реалізації, комунікацію зі стейкхолдерами, інструменти, постановку цілей і результати роботи [↑](#footnote-ref-4)
5. **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-5)
6. **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-6)
7. 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-7)
8. 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-8)