**Дмитрів Юрій**

**КБ-201**

**Процес розробки програмного забезпечення на стороні розробника**

**Вступ**

Процес розробки програмного забезпечення — сукупність ряду послідовних дій, спрямованих на розробку програмного забезпечення (ПЗ).

Існує кілька моделей такого процесу, кожна з яких описує свій підхід, у вигляді завдань і/або діяльності, які мають місце в ході процесу. Вибір тієї або іншої моделі здійснюється відповідно до обраної методології розробки програмного забезпечення.

Розробка програмного забезпечення є складним і многогранним процесом, який вимагає не лише технічних знань і експертизи, але і чіткої організації та структури. Кожен програмний продукт починається з ідеї, яка потребує детального розгортання в планів розробки, дизайну, програмування, тестування і впровадження. Цей процес структурований з метою забезпечення систематичного підходу до кожного етапу розробки, з урахуванням вимог замовника, специфікацій та стандартів безпеки та якості. Крім технічної частини, ключовим елементом успішної розробки є також організація комунікації в команді розробників, ефективне управління ресурсами та контроль якості продукту на кожному етапі його життєвого циклу.

**Backend-розробка**

Бекенд, потрібен кожній компанії, яка взаємодіє з клієнтами через сайт чи веб-сервіс. Наприклад, в інтернет-магазині він відповідає за ідентифікацію покупців, зберігання та пошук інформації в базі даних, роботу полів реєстрації та кошика, взаємодію з банком, автоматичне розсилання листів та оновлення контенту.

Іншими словами, бекенд забезпечує роботу всіх тих функцій, які споживачеві надає інтерфейс. У проекті, що вже існує, бекенд-розробники зазвичай покращують сервіс: додають функціональність і нові можливості, виправляють помилки. Якщо проект розробляють з нуля, то завдання бекендера будуть такими:

1. Вивчення контексту задачі. Ще одна особливість бекенда в тому, що він вимагає більшого порівняно з фронтендом занурення у конкретні бізнес-процеси. Наприклад, якщо ви не знаєте, що таке лійка продажів, то навряд чи зможете розробити її логіку разом з усіма фільтрами та переходами.
2. Пошук рішення. Насправді головне завдання бекенд-розробника – вирішення бізнес-завдань клієнта програмними засобами. І написання коду – це лише один із способів досягнення цієї мети. Досвідчений розробник пропонує клієнту найбільш практичне рішення на основі шаблонів програмування, веб-стандартів та напрацьованого роками досвіду. У великих проектах для цього проводять планерки за участю фахівців, які відповідають за різні фрагменти сервісу.
3. Вибір мови. В рамках поставленого йому завдання бекендер може вибирати будь-яку з серверних мов. Головне, щоб він забезпечив зручність, швидкість та продуктивність роботи. Універсального вибору немає. Так для великих проектів зазвичай вибирають Java, для середніх – PHP, для стартапів, що швидко розвиваються, – Ruby або Go. Докладніше про мови для backend ми поговоримо нижче.
4. Вибір інструментів. Крім мови програмування на початковому етапі розробки необхідно вибрати інструменти. Це можуть бути бази даних, наприклад, MySQL, PostgreSQL або MongoDB. А також найбільш підходящий під ваші цілі фреймворк. Термін «фреймворк» перекладається з англійської, як каркас, структура. Це набір готових рішень, що прискорює процес розробки програм. Усередині фреймворків реалізовано код для вирішення безлічі найпоширеніших завдань: ідентифікації користувачів, роботи з базами даних тощо. Приклади популярних у бекенді фреймворків: Spring Framework, Django, Express.js .
5. Написання коду. Це основне завдання бекендера, яке займає найбільше часу. У процесі написання коду велике значення приділяється спільної роботи з іншими розробниками. Спростити її дозволяють системи контролю версій, або СКВ. Вони записують на хмарному сервері всі зміни коду і дозволяють за потреби повернутися до певного варіанту.

**Мови програмування в бекенді**

****Рисунок 1 – мови для backend розробки.

Набір технологій, на основі яких розробляють веб-проект, включає операційну систему, тип веб-сервера, систему управління базами даних, мову програмування та фреймворк. Усе разом це називають стеком технологій. При вказуванні мови до неї часто додають назву вибраного фреймворку. Наприклад, JavaScript + Node.js або PHP + Yii .

1. *PHP*

За даними ресурсу W3techs, понад 77% всіх веб-сайтів у світі написані мовою PHP. Він простий у вивченні, але погано масштабується та досить повільно працює. Тому часто можна почути, що PHP вмирає. Так це чи ні, покаже час. А поки що PHP-розробники забезпечені роботою на багато років вперед, адже мільйони сайтів на PHP нікуди не подінуться і їх потрібно обслуговувати.

1. *C# + ASP.NET*

Універсальний C# у зв’язці з потужним фреймворком від Microsoft займає друге місце в рейтингу популярності серверних мов для веб-сайтів.

1. *Ruby + Ruby on Rails*

Мова загального призначення Ruby дозволяє швидко створити працюючий проект. Тому її часто обирають за основу стартапи, серед яких Kickstarter, Airbnb та інші.

1. *Java*

Головні переваги мови Java – кросплатформність та продуктивність. Її можна використовувати для створення програм на Android, десктопних та веб-додатків. А висока продуктивність Java дозволяє створювати великі проекти з великим навантаженням. Наприклад, програми для роботи з Big Data.

У рейтингу популярності на основі пошукових запитів TIOBE за листопад 2022 Java займає третє місце. Крім цього, у 70% вакансій для розробників знання цієї мови є обов’язковою умовою для працевлаштування.

1. *Python*

Зручний і нескладний у вивченні Python підходить для вирішення багатьох завдань. А завдяки хорошим фреймворкам, наприклад Django, дозволяє швидко створити блог, форум чи інтернет-магазин. У рейтингу TIOBE мова програмування Python стабільно посідає перше місце.

1. *Go (Golang)*

Швидка і легка мова від Google ще не така популярна, як перелічені вище, але активно розвивається. Тому фахівці з Go потрібні у багатьох компаніях і досить непогано заробляють.

1. *JavaScript + Node.js*

Мову програмування JavaScript широко використовують для фронтенд-розробки, але вона знаходить застосування і в бекенді. Платформа Node.js дозволяє запускати JS на серверній частині сайту. З одного боку, це забезпечує гнучкість налаштувань. З іншого, ускладнює масштабування та уповільнює роботу.

**Frontend**

Що таке фронтенд? Frontend - це публічна частина web-додатків (вебсайтів), з якою користувач може взаємодіяти та контактувати безпосередньо. У Frontend входить відображення функціональних завдань, інтерфейсу користувача, що виконуються на стороні клієнта, а також обробка користувальницьких запитів. По суті, фронтенд це все те, що бачить користувач при відкритті web-сторінки.

У свою чергу, web-додаток - клієнт-серверний додаток, в якому клієнтом виступає в основному браузер, а сервером - web-сервер. Логіка web-додатка розподілена між сервером та клієнтом, зберігання даних здійснюється переважно на сервері, обмін інформацією відбувається через мережу. Простіше кажучи, це те, що бачить користувач і які дії виконує щоразу, коли підключається до Інтернету і відкриває будь-який браузер.

Frontend-розробка - це робота зі створення публічної частини web-додатку, з якою безпосередньо контактує користувач, та функціоналу, який зазвичай виконується на стороні клієнта. Тобто, фронтенд розробник працює над тим, щоб на сайті кожна кнопочка, іконка, текст і вікно не тільки стояли на своєму місці, не перекривали один одного і виглядали цілісно (це веб-верстка), а й щоб вони виконували своє пряме призначення. робили якусь дію (наприклад, щоб кнопка "купити" відкривала кошик, а "play" - запускала відтворення фільму чи музики).

З метою створення затребуваного та доступного продукту (веб-додатку) фронтенд-розробнику необхідно взаємодіяти з іншими програмістами, дизайнерами, маркетологами, аналітиками та іншими фахівцями.

Компоненти фронтенд:

1. HTML (HyperText Markup Language) кажучи простими словами - це мова розмітки всіх елементів та документів на сторінці, та їх взаємодія у структурі сторінки.
2. CSS (Cascading Style Sheets) - це мова характеристики та стилізації зовнішнього вигляду документа. За допомогою CSS-коду браузер розуміє, як необхідно відображати елементи. CSS створює шрифти, кольори, визначає розташування блоків сайту та інше. Також адаптує той самий документ у різних стилях, виводить передачу на екран або для читання голосом.
3. JavaScript — мова, створена для оживлення веб-сторінок. Завдання JavaScript - відгукуватися на дії користувача, обробляти натискання клавіш, переміщення курсору, кліки мишкою. JavaScript також дає можливість вводити повідомлення, надсилати запити на сервер, а також завантажувати дані без перезавантаження сторінки, і так далі.

**Головне завдання Frontend-розробника**

Якщо ви маєте творчий підхід до роботи, хочете розробляти і створювати динамічні інтерфейси, вам однозначно дорога у фронтенд .

Вся фронтенд розробка виконується на стороні користувача, вона не менш важлива ніж бекенд розробка (про неї ми також згадаємо трохи нижче), тому що це те, що користувач бачить і з чим взаємодіє. Основне завдання фронтенд спеціаліста - це пов'язати представлені дизайнером графічні макети Web-додатки (сторінки сайту) з бекендом і при необхідності реалізувати обчислювальний функціонал на стороні користувача. Основні технології в арсеналі фронтендера це HTML, CSS та JavaScript. Звичайно, при роботі в команді потрібно знати і розбиратися в багатьох процесах, суміжних із роботою фронтенд-розробки. Будучи вже досвідченим фронтенд-розробником, потрібно бути знайомим з бекенд-технологіями і розуміти принципи взаємодії користувача та додатків (UX).

**Frontend і Backend розробка - відмінності та взаємодія між собою**

Розглянемо процес взаємодії frontend та backend:

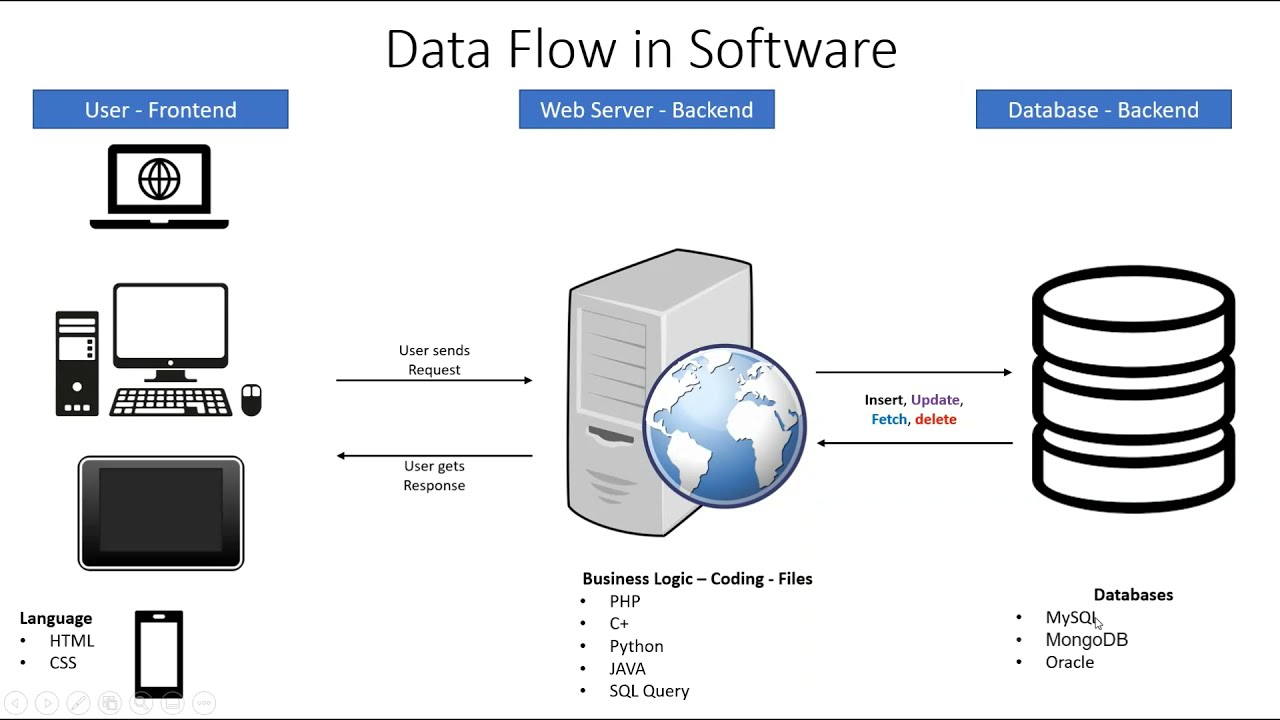
* фронтенд відправляє інформацію користувача в бекенд;
* інформація обробляється;
* інформація повертається назад, прийнявши цілісну форму та виконавши опрацьований запит.

Всі ці завдання виконує кілька спеціалістів одночасно, це взаємодоповнююча командна робота.

Варіанти взаємодії frontend та backend:

* HTTP-запит відправляється на сервер, сервер у процесі пошуку інформації, вбудовує її в шаблон і повертає назад у вигляді HTML-сторінки.
* Випадок із застосуванням інструментарію AJAX (Asynchronous JavaScript and XML). У цьому випадку запит надсилає JavaScript, який завантажений у браузер, відповідь надходить у форматі XML або JSON.
* Односторінкові програми, які завантажують дані без поновлення сторінок. Це робиться за допомогою AJAX або фреймворків Angular та Ember.
* Ember або бібліотека React надають допомогу у використанні програми в клієнтській частині та на сервері. Frontend та backend взаємодіють через AJAX та HTML-код, який обробляється на сервері.

Робота та обов'язки frontend та backend девелоперів найчастіше розділені, але іноді виникає необхідність у програміста вирішувати проблеми як на стороні сервера, так і в клієнтській частині. Досить часто можна зустріти фахівців, які можуть поєднувати frontend і backend, вони абсолютно впевнено почуваються як з одного, так і з іншого боку медалі.

  
Рисунок 2 – приклад взаємодії між собою(frontend - backend).

**Database**

**Яку роль відіграє база даних у розробці пз.**

Роль баз даних у розробці програмного забезпечення надзвичайно важлива і охоплює кілька ключових аспектів.

По-перше, бази даних забезпечують ефективне зберігання інформації, необхідної для роботи програмного продукту. Це включає в себе дані користувачів, конфігураційні налаштування, тексти і зображення, що використовуються в інтерфейсі програми.

Далі, бази даних забезпечують надійний механізм для організації і зберігання даних, що використовуються в програмі. Це може включати структури даних, такі як таблиці, схеми та індекси, що дозволяють ефективно здійснювати доступ і операції з даними.

Третій аспект - бази даних є основою для забезпечення інтеграції та співпраці між різними компонентами програмного продукту. Вони дозволяють різним частинам програми звертатися до одних і тих самих даних, що спрощує розробку і підтримку програми.

Також, бази даних забезпечують механізм для здійснення транзакцій і забезпечення цілісності даних. Це важливо для забезпечення надійності і консистентності даних під час одночасних операцій, які здійснюються користувачами програми.

* Забезпечення ефективного зберігання усієї необхідної інформації, що використовується програмним продуктом.
* Створення структур даних, які дозволяють ефективний доступ і операції з даними, такі як таблиці, схеми і індекси.
* Забезпечення можливості взаємодії різних частин програми з одним джерелом даних.
* Здійснення транзакцій та забезпечення цілісності даних для забезпечення надійності і консистентності даних під час роботи програми.

Призначення баз даних в контексті розробки програмного забезпечення також включає кілька важливих аспектів, які сприяють успішному впровадженню і підтримці програмних продуктів.

Бази даних служать основою для зберігання не лише структурованої інформації, але і надання можливості для проведення різних аналітичних операцій і отримання цінної інформації для прийняття управлінських рішень. Вони дозволяють збирати, організовувати і аналізувати дані з різних джерел, що робить їх незамінним інструментом у сучасному бізнес-середовищі.

Крім того, бази даних підтримують безпеку даних, забезпечуючи захист від несанкціонованого доступу та зберігання інформації відповідно до вимог законодавства і стандартів безпеки даних. Це включає контроль доступу до даних, шифрування і аудит дій з даними для запобігання інцидентам і витокам інформації.

Крім того, бази даних забезпечують механізм для інтеграції з іншими системами і програмним забезпеченням, що дозволяє створювати комплексні рішення та забезпечувати сумісність інформаційних технологій у великих організаціях.

1. Аналітика і звіти: Використання даних для створення аналітичних звітів і інформаційних панелей для прийняття управлінських рішень.
2. Безпека даних: Захист інформації від несанкціонованого доступу і забезпечення відповідності стандартам безпеки.
3. Інтеграція з іншими системами: Забезпечення можливості обміну даними із зовнішніми і внутрішніми системами.
4. Скорочення часу на розробку: Використання готових рішень і засобів баз даних для прискорення процесу розробки програмного забезпечення.

**Контейнеризація**

Контейнеризація, зокрема з використанням Docker і Kubernetes, відіграє значну роль у розробці, розгортанні і управлінні програмним забезпеченням. Ось детальна інформація про кожен з цих інструментів:

Docker:

Docker - це платформа для розробки, доставки та запуску програмного забезпечення у вигляді контейнерів. Контейнери Docker використовуються для упаковки програм, разом з усіма їхніми залежностями, в один стандартизований пакет. Основні аспекти Docker включають:

1. Docker дозволяє упаковувати програми та їх залежності у віртуалізовані контейнери, які забезпечують ізольоване середовище для виконання програмного коду.
2. Docker забезпечує стандартизовану платформу для розробки і розгортання програм, що полегшує розробникам перенесення і запуск додатків між різними середовищами.
3. Docker дозволяє швидко розгортати нові версії програмного забезпечення і керувати залежностями середовища.

Kubernetes:

Kubernetes (K8s) - це відкрита система для автоматизації розгортання, масштабування і керування контейнеризованими додатками. Основні функції Kubernetes включають:

1. Kubernetes керує і автоматизує розгортання та керування багатьма контейнерами Docker, включаючи їх масштабування, надійність і доступність.
2. Kubernetes використовує декларативний підхід до керування, що дозволяє описувати бажаний стан системи і автоматично приводити систему до цього стану.
3. Kubernetes підтримує різноманітні архітектурні шаблони для складних додатків, таких як мікросервісна архітектура, що дозволяє ефективно керувати ресурсами і завданнями.
4. Kubernetes є розширюваною і підтримує різні платформи обчислення, що дозволяє використовувати його для різних видів додатків і середовищ.

Використання Docker і Kubernetes у розробці ПЗ:

* Швидке розгортання і відладка: Docker дозволяє розробникам швидко створювати і запускати локальні середовища для розробки і тестування.
* Kubernetes забезпечує управління цими контейнерами у виртуальному чи хмарному середовищі.
* Масштабування і висока доступність: Kubernetes дозволяє автоматично масштабувати додатки залежно від навантаження і забезпечувати високу доступність за рахунок розміщення контейнерів на різних вузлах і автоматичного відновлення в разі відмов.
* Інтеграція з іншими інструментами: Docker і Kubernetes інтегруються з іншими інструментами розробки, такими як системи управління версіями коду, CI/CD системи, моніторинг і логування, що забезпечує повний цикл життєвого циклу розробки ПЗ.

Загалом, Docker і Kubernetes стали важливими інструментами у розробці сучасного програмного забезпечення, спрощуючи розгортання, управління і масштабування контейнеризованих додатків у будь-якому середовищі.

**Development operation**

DevOps (Development and Operations) - це підхід до розробки програмного забезпечення, який поєднує практики розробки (Dev) і управління інфраструктурою (Ops) для поліпшення швидкості впровадження змін і якості розробленого продукту. Основні аспекти DevOps включають культурні зміни, автоматизацію процесів, застосування CI/CD практик, моніторинг і логування, безпеку і управління конфігураціями.

Культурні зміни в DevOps орієнтовані на створення спільної культури і взаєморозуміння між розробниками програмного забезпечення і технічним персоналом, що відповідає за інфраструктуру. Це включає спільні практики, засоби комунікації та взаємну підтримку у вирішенні завдань.

Автоматизація є ключовим елементом DevOps і використовується для автоматизації всіх аспектів процесу розробки і розгортання програмного забезпечення. Це включає автоматизацію тестування, збору інформації про продуктивність, контроль версій і розгортання змін.

CI/CD (Continuous Integration / Continuous Deployment) практики спрощують і прискорюють процеси інтеграції нового коду в основний репозиторій, його автоматичного тестування та автоматичного розгортання у виробниче середовище. Це дозволяє знижувати час розгортання нового функціоналу і знижувати ризики пов'язані з ним.

Моніторинг і логування є важливою частиною DevOps, оскільки вони забезпечують можливість вчасно виявляти проблеми у виробничому середовищі, вирішувати їх і вдосконалювати продукт.

Безпека є невід'ємною частиною DevOps і включає в себе керування доступом, моніторинг вразливостей і захист даних, що є критичними для забезпечення безпеки в процесі розробки і експлуатації програмного забезпечення.

Управління конфігураціями в DevOps включає в себе зберігання коду та конфігураційних файлів у системах контролю версій, що дозволяє легко відслідковувати зміни і управляти різними версіями програмного забезпечення.

Загалом, DevOps спрямований на поліпшення якості, швидкості та безпеки розробки і експлуатації програмного забезпечення шляхом інтеграції культурних змін, автоматизації процесів і використання сучасних інструментів і практик.