вищі Scratch для групи дітей віком 10–12 років, у групах не більше 12 дітей для онлайн груп. Особливістю даного модуля стала інтеграція вивчення азів програмування одночасно з англійською мовою.

Апробацію розробленого модуля було проведено в період з 19 лютого по 26 квітня 2021 року, власне саме під час карантину. Із 12-ти дітей, які навчалися у групі, успішно завершили навчання та отримали сертифікати 10 дітей, а це – 83%. Оскільки навчання повністю добровільне і не передбачає оцінювання, то такий показник вказує на ефективність розробки. За відгуками батьків, навчання мало позитивний ефект та зацікавило дітей до подальшого вивчення Scratch, за наступними рівнями складності.

Надалі планується розробити тестовий контроль базових навичок зі Scratch та знань з англійської мови для того, щоб здійснювати диференційований набір у групи Code Club за програмою Scratch Newbie, що досі не було передбачено.

1. Офіційний сайт організації “Code Club”. URL: https://codeclub.org/en.
2. Офіційний сайт українського представництва організації “Code Club”. URL: https://codeclub.com.ua.
3. Програми Code Club. URL: https://projects.raspberrypi.org/en/codeclub.

ПРОЕКТУВАННЯ ТА РОЗРОБКА МЕДІА РЕСУРСУ “SHPALTA IF”. РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ ЗАСТОСУНКУ

Полатайко Ігор,
IV курс ОР бакалавр,
факультет математики та інформатики.
Науковий керівник – Іщеряков С.М.,
кандидат технічних наук, доцент.

Програмне забезпечення “Шпальта ІФ” призначене бути централізованим медіа-ресурсом для розміщення публікацій інформаційного та розважального характеру, що має гнучку та персоналізовану систему сповіщень для поширення найбільш релевантної інформації серед груп зацікавлених людей.

Процес розробки даного програмного забезпечення складається з наступних фаз:

- збору та аналізу бізнес вимог;
- визначення функціональних та нефункціональних вимог;
- розробки архітектурного дизайну;
- імплементації та тестування програмного забезпечення.

На фазі збору бізнес вимог було проведено аналіз області застосування даного програмного забезпечення та потреб і побажань замовників. Результатом даного етапу став набір бізнес вимог [1], що описували потребу в даному програмному забезпеченні та його основне призначення. Загалом, список бізнес вимог можна підсумувати визначенням: медіа-ресурс для поширення релевантних публікацій серед зацікавленої аудиторії.

Після проведеного аналізу бізнес вимог, наступним етапом було прийняття рішень щодо способів задовільнити дані бізнес вимоги, та визначення функціональних та нефункціональних вимог до програмного забезпечення. Основними рішеннями прийнятими на даному етапі були рішення, що стосуються формату продукту, який розроблятиметься.

Медіа ресурс представляє собою веб-застосунок, що поділяється на публічну та адміністративну частини. На публічній частині користувачі мають змогу переглянути стрічку з постами, з можливістю фільтрації за категорією чи автором та можливістю пошуку. Також, користувачі можуть переглядати індивідуальні пости, залишати коментарі, переглядати акаунти авторів статей чи залишати відгуки, видимі тільки адміністраторові. На адміністративній частині, користувачі, в залежності від ролі в системі мають різний доступний для них функціонал. Так, наприклад, користувач з роллю "автор", може створити нову публікацію, а, також, оновити будь-яку з власних, вже існуючих, публікацій. В той час, як користувач з роллю "редактор", може оновляти чи видаляти будь-які пости на ресурсі, але не може створювати власних. Найширший функціонал на адміністративній частині доступний користувачеві з роллю "адміністратор", який відповідальний за менеджмент користувачів, категорій, постів та має змогу переглядати статистичну інформацію і відгуки користувачів.

Оскільки, більшість людей регулярно користуються менеджером Телеграм, що спричиняє все більший ріст популярності, так званих, телеграм-ботів: Телеграм-акаунтів, що контролюються програмним забезпеченням, було прийнято рішення побудувати телеграм-бот для сповіщення релевантної аудиторії про нові публікації на ресурсі. З метою забезпечення персоналізації сповіщень, користувачі мають змогу обрати категорії, сповіщення з який вони хочуть отримувати.

Наступним етапом побудови даного програмного забезпечення був етап проектування архітектури та дизайну програмного забезпечення.

Одними з нефункціональних вимог, що мали прямий вплив на архітектуру були вимоги стосовно SEO (Search Engine Optimization) та швидкості завантаження веб сторінки при переході на ресурс за зовнішнім посиланням. Відповідно до даних вимог, веб-ресурс повинен добре індексуватися у пошукових системах, а користувачі, при переході на ресурс ззовні повинні миттєво отримувати веб-сторінку. Як результат було прийнято рішення використати сервер сайд рендеринг: підхід при якому веб сторінки будуються на стороні сервера. Це одночасно забезпечує і хороший рівень SEO, оскільки пошукові системи не запускають JavaScript код, і високу швидкість завантаження веб сторінки при переході на ресурс ззовні.

Після прийняття архітектурних рішень, наступним етапом є процес проектування дизайну програмного забезпечення, а саме, визначення компонентів та взаємодії між ними [2]. На даному етапі, було проаналізовано список функціональних вимог, визначено доменні сутності та пов'язану з ними бізнес логіку. Процес визначення доменних сутностей зводився до аналізу іменників, які зустрічаються у функціональних вимогах: більшість з них представляють певний доменний об'єкт у даному програмному забезпеченні. Кожна з доменних сутностей має відповідний клас-репозиторій, де інкапсульована логіка взаємодії з базою даних. Самі ж функціональні вимоги та бізнес правила реалізовуються в класах-сервісах, призначених для інкапсуляції бізнес логіки.

На етапі реалізації програмного забезпечення, було проведено декомпозицію робіт та визначення технічних завдань [3]. Після чого технічні завдання виконувалися у відповідності до Scrum методології. На початку кожного двотижневого спринта відбувалося попереднє планування, після чого, завдання потрапляли в розробку. В кінці спринта проводилася ретроспектива та огляд робіт, виконаних в його рамках.

Для реалізації даного програмного забезпечення було використано технічний стек мови програмування Java [4] та фремворк Spring Boot (Spring Boot Web MVC, Spring Boot Security, Spring Boot Data JPA), обробник шаблонів Apache Freemarker, інструмент міграцій баз даних Flyway, інструмент збірки проектів Maven та ін. Базою даних було обрано використати MySQL.

У процесі розробки використовувалася система контролю версій Git. Перед тим, як потрапити до основної вітки розробки, код проходив через CI (Continuous Integration) процес [5], налаштований за допомогою Travis CI, та процес рев'ю коду, де перевірялася коректність та якість виконання завдання. Після потрапляння до основної вітки розробки, нова версія автоматично збиралася та розгорталася на тестовому середовищі, завдяки CD (Continuous Deployment) процесу, налаштованому за допомогою платформи Heroku [6].

Для запобігання регрешину програмного забезпечення, після внесення змін у майбутньому, код було покрито інтеграційними тестами, що перевіряють правильність роботи основного функціоналу програмного забезпечення, не прив'язуючись до деталей імплементації.

Отже, в рамках даної роботи, було розроблено веб-ресурс для зручного розміщення публікації, що інтегрований з телеграм-бот акаунтом для поширення сповіщень про нові публікації на ресурсі, серед релевантної аудиторії.

1. Business Requirements Document: a High-level Review. URL: https://www.isixsigma.com/implementation/project-selection-tracking/business-requirements-document-high-level-review.
2. Eric Evans. Domain-Driven Design: Tackling Complexity in the Heart of Software: Addison-Wesley Professional, 2003. 560 p.
3. Applying the work breakdown structure to the project management lifecycle. URL: https://www.pmi.org/learning/library/applying-work-breakdown-structure-project-lifecycle-6979.
4. Іщеряков С. М., Яновський Ю. М., Козленко М. І. Програмування множинного наслідування реалізації на основі анонімних внутрішніх класів Java. Proceedings of the 2019 Scientific Seminar on Innovative Solutions in Software Engineering (м. Івано-Франківськ, 10 грудня 2019 р.). Івано-Франківськ, 2019. С. 26-27. URL: https://doi.org/10.5281/zenodo.4263418.
5. Continuous integration. Wikipedia. URL: https://en.wikipedia.org/wiki/Continuous_integration.
6. Continuous Delivery on Heroku. URL: https://www.heroku.com/continuous-delivery.