МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Центральноукраїнський національний технічний університет

Кафедра кібербезпеки та програмного забезпечення

На правах рукопису

Хільченко Тетяна Юріївна

Програмне забезпечення системи дистанційного навчання контролю якості програмних застосунків

Спеціальність: 123 «Комп'ютерна інженерія»

Освітній ступінь: бакалавр

Науковий керівник:

Бісюк Віктор Анатолійович _____

(підпис) (дата)

викладач

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри

_____ О.А. Смірнов

(підпис) ПІБ

«____»_____2021 р.

Кафедра_КбПЗ_2021рік

Кропивницький – 2021

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Спеціальність *123 Комп'ютерна інженерія*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____О.А.Смірнов
« 11 » січня 2021 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Хільченко Тетяні Юріївні
(прізвище, ім'я, по батькові)

1.Тема роботи *Програмне забезпечення системи дистанційного навчання контролю якості програмних застосунків*

керівник роботи *Бісюк Віктор Анатолійович, викладач*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 204-02 від 28.12.2020 року

2. Строк подання студентом роботи до захисту *22.05.2021 р.*

3. Мета та завдання кваліфікаційної бакалаврської роботи*: Метою розробки є програмне забезпечення системи дистанційного навчання контролю якості програмних застосунків*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

Кафедра - КБПЗ - 2021рік

6. Дата видачі завдання « 11 » січня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів кваліфікаційної бакалаврської роботи	Примітка
1.	Аналіз існуючих систем	10.03.2021 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2021 р.	
3.	Розробка моделі компонента	20.03.2021 р.	
4.	Розробка структур даних	25.03.2021 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2021 р.	
6.	Програмування алгоритмів	10.04.2021 р.	
7.	Оформлення ПЗ	17.04.2021 р.	
8.	Попередній захист роботи	22.05.2021 р.	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

Кафедра_КбПЗ_2021рік

АНОТАЦІЯ

Хільченко Т.Ю. Програмне забезпечення системи дистанційного навчання контролю якості програмних застосунків. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній кваліфікаційній бакалаврській розроблено програмне забезпечення системи дистанційного навчання контролю якості програмних застосунків.

Метою розробки є програмне забезпечення системи дистанційного навчання контролю якості програмних застосунків.

Результат роботи – портал дистанційного навчання тестувальників програмного забезпечення.

В процесі роботи над програмною моделлю виконано аналіз існуючих програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма працю на мобільних пристроях, планшетах та персональних комп'ютерах. Підтримує роботу у всіх популярних браузерах, а саме: Chrome, Safari, FireFox, Opera.

Програму розроблено на базі php-фреймворку Laravel.

Ключові слова: комп'ютерна інженерія, QAMentoring портал, дистанційна освіта, безкоштовні курси.

Кафедра_КбПЗ_2021рік

ABSTRACT

Khilchenko T.Y. Software for distance learning system quality control of software applications. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

In this bachelor's qualification the software of the system of distance learning of quality control of software applications is developed.

The purpose of the development is the software of the distance learning system for quality control of software applications.

The result is a distance learning portal for software testers.

In the process of working on the software model, an analysis of existing software was performed. All components of the developed software are fully described.

Developed user-friendly interface. Instructions for working with software are given.

The program works on mobile devices, tablets and personal computers. Supports work in all popular browsers, namely: Chrome, Safari, FireFox, Opera.

The program is developed on the basis of the php-framework Laravel.

Keywords: computer engineering, QAMentoring portal, distance education, free courses.

Кафедра_КОПЗ_2021рік

ВСТУП

Актуальність теми. Дистанційне навчання - це отримання освіти за допомогою інтернету і сучасних інформаційних і телекомунікаційних технологій. Це область спілкування, інформації і знань. Виходячи з того, що професійні знання старіють дуже швидко, необхідно їх безперервне вдосконалення. Дистанційна форма навчання дає сьогодні можливість створення систем масового безперервного самонавчання, загального обміну інформацією, незалежно від наявності тимчасових і просторових поясів. При дистанційному навчанні відбувається обмін навчальною інформацією за допомогою сучасних засобів на відстані.

Також виходячи з ситуації яка склалась в крайні на даний момент, дистанційне навчання є універсальним засобом здобуття освіти віддалено дотримуючись карантинних вимог.

На допомогу учням, викладачам, та загалом людям які працюють у галузі освіти прийшла в нагоді СДН.

Мета й завдання дослідження. Метою роботи є веб-застосунок дистанційного навчання QA інженерів.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Створення та організація бази даних.

– Створення Back-end частини порталу на фреймворку Laravel.

– Створення візуальної частини за допомогою гіпертекстовій розмітці HTML, створення стилів порталу за допомогою роботи з каскадною таблицею стилів CSS та її фреймворком Bootstrap.

Кафедра_КбПЗ_2021рік

Практична цінність отриманих результатів полягає в тому що за допомогою завантаження матеріалу іншого предмету в БД, можна легко створити цілу систему дистанційного навчання по будь-якій спеціальності. При незначних змінах користувач може замість статусу “Done”, обрати отримання оцінок і встановити необхідну бальну систему. Система повністю гнучка й можлива налаштованість під потреби користувача.

Кафедра_КбПЗ_2021рік

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Вперше про відкриті онлайн-курсах заговорили майже десять років тому, коли канадським педагогам з університету Манітоби вдалося запустити курс «Connectivism and Connective Knowledge», який зібрав більше двох тисяч передплатників. З ростом кількості і популярності перших онлайн-курсів, стало зрозуміло, що це не новий модний інтернет-тренд, а сучасний відповідь на багато недоліків і упущень традиційної освітньої системи. Наприклад, в США навчання в університетах завжди було недешевим задоволенням і отримати якісну вищу освіту вдавалося далеко не всім. На сьогоднішній день розмір загального боргу американських студентів перевищує $ 1,3 трильйона і можна спостерігати в реальному часі, як ця сума виростає на $ 2,726 в секунду. Оскільки рейтинг і ім'я університету також надзвичайно важливі для роботодавців, від вибору вузу може прямо залежати подальша кар'єра випускника. Ці та інші проблеми - нестача компетентних викладачів, неактуальне матеріал і застарілі методи роботи з інформацією універсальні не тільки для американських, а й для багатьох світових університетів. Вирішувати такі масштабні завдання взялися не міністерства і уряди, а викладачі та активісти, які вирішили використовувати головний метод поширення знань в сучасному світі - інтернет.

Перша відкрита масова освітня платформа - Khan Academy (Академія Хана) - з'явилася в онлайн-просторі в 2006 році. Сьогодні Khan Academy надає більше п'яти тисяч абсолютно безкоштовних курсів, а її команда з однієї людини зросла в колектив з 80-ю співробітниками.[1]

Але справжній ривок онлайн-освіти відбувся в 2012 році, який газета The New York Times назвала «роком масових відкритих онлайн курсів».

Кафедра_КОПЗ_2021рік

Причиною тому стала поява сайтів edX, Coursera і Udacity, які з моменту заснування отримали істотну фінансову підтримку - одні від іменитих університетів, а інші - від венчурного капіталу.[2]

І якщо ще десять років тому записатися на курси Массачусетського технологічного інституту або вивчати основи програмування під керівництвом викладачів Гарвардського університету могла тільки жменька обраних студентів, яким таке навчання обходилося в десятки тисяч доларів, з появою університетських курсів онлайн, безкоштовний доступ до кращих кладезем знань на планеті став відкритий усім, у кого є комп'ютер і підключення до інтернету.

Лише через кілька років після того, як університетські лекції онлайн стали набирати популярність в світі, все більше вищих навчальних закладів почали говорити про перспективність так званої змішаної освіти (blended education), суть якої полягає в тому, що курси кращих викладачів стають основою для навчання в інших вузах, а успіхи студентів в онлайні переносяться в дипломи. Такий підхід покликаний об'єднати краще від офлайна і онлайну в академічному освіту. За словами експертів, таким чином можна вивести освіту на якісно новий рівень - адже з лекторами світових університетів зможуть конкурувати тільки кращі локальні викладачі та вузи.

Створення власного порталу для отримання повноцінних знань для Junior працівників стане вагомим мотивом для вступу до університету, та допоможе студентам якісніше фільтрувати інформацію з інтернету.

1.2 Область застосування

Програмне забезпечення системи дистанційного навчання контролю якості програмних застосунків можливо розширювати, додаючи просто нові

курси. Також даний курс є повністю скомпонований для підготовки QA інженерів рівня Junior, що забезпечує кваліфіковану підготовку даних фахівців університетом.

Кафедра_КОПЗ_2021рік

Таким чином, виходячи з вищеперерахованого, системи дистанційного навчання контролю якості програмних застосунків, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі. Оскільки учню не потрібно вивчати семестр предмет, а також навантажувати викладача додатковими конспектами, а можливо просто пройти вже повністю готовий курс.

Кафедра_КбПЗ_2021рік

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень

Сучасне дистанційне навчання будується на використанні наступних основних елементів:

- середовища передачі інформації (пошта, телебачення, радіо, інформаційні комунікаційні мережі);
- методів, залежних від технічного середовища обміну інформацією.

У XXI столітті перспективним є інтерактивна взаємодія з учням за допомогою інформаційних комунікаційних мереж, яка масово виділяється серед інтернет-користувачів. У 2003 році ініціативна група ADL почала розробку стандарту дистанційного інтерактивного навчання SCORM, який передбачає широке застосування інтернет-технологій. Введення стандартів сприяє як поглибленню вимог до складу дистанційного навчання, так і вимог до програмного забезпечення.

На даний момент є актуальними, такі форми дистанційного навчання:

- Чат-заняття - навчальні заняття, здійснювані з використанням чат-технологій. Чат-заняття проводяться синхронно, тобто всі учасники мають одночасний доступ до чату. В рамках багатьох дистанційних навчальних закладів діє чат-школа, в якій за допомогою чат-кабінетів організується діяльність дистанційних педагогів і учнів;
- Веб-заняття - дистанційні уроки, конференції, семінари, ділові ігри, лабораторні роботи, практикуми та інші форми навчальних занять, що проводяться за допомогою засобів телекомунікацій та інших можливостей

Кафедра_КБПЗ_2021рік

«Всесвітньої павутини». Для веб-занять використовуються спеціалізовані освітні веб-форуми - форма роботи користувачів з певної теми або проблеми за допомогою записів, що залишаються на одному з сайтів з встановленої на ньому відповідною програмою.

Від чат-занять веб-форуми відрізняються можливістю більш тривалої (багатоденної) роботи і асинхронним характером взаємодії учнів і педагогів;

- Телеконференція - проводиться, як правило, на основі списків розсилки з використанням електронної пошти. Для навчальних телеконференцій характерно досягнення освітніх завдань. Також існують форми дистанційного навчання, при якому навчальні матеріали висилаються поштою в регіони.

В основі такої системи закладений метод навчання, який отримав назву «Природний процес навчання» (англ. Natural learning manner). Дистанційне навчання - це демократична проста і вільна система навчання. У 21 столітті активно використовується жителями Європи для отримання додаткової освіти. Студент, постійно виконуючи практичні завдання, набуває стійкі автоматизовані навички. Теоретичні знання засвоюються без додаткових зусиль, органічно вплітаючись в тренувальні вправи. Формування теоретичних і практичних навичок досягається в процесі систематичного вивчення матеріалів і прослуховування і повторення за диктором вправ на аудіо і відеоносіях (при наявності);

- Телеприсутність. Існує багато різних способів дистанційного навчання. Наприклад, дистанційне присутність за допомогою робота R.Bot 100. У наш час технологій, в одній зі шкіл, йде експеримент по такому виду дистанційного навчання. Хлопчик-інвалід, перебуваючи вдома за комп'ютером, чує, бачить, розмовляє за допомогою робота. Учитель задає йому питання, він відповідає. При цьому і вчитель бачить учня, тому що на роботі знаходиться монітор.

Кафедра_КбПЗ_2021рік

При цьому у хлопчика створюється майже повне враження, що він знаходиться в класі разом зі своїми однолітками на уроці. На перервах він може також спілкуватися зі своїми однокласниками. [3]

У Європі в кінці XVIII століття, з появою регулярної і доступною поштового зв'язку, виникло «кореспондентське навчання». Учні поштою отримували навчальні матеріали, листувалися з педагогами і здавали іспити довіреній особі або у вигляді наукової роботи.

Початок XX століття характеризується бурхливим технологічним зростанням, наявністю телеграфу і телефону. Але достовірних фактів про їх використання в навчанні немає. У той же час триває епоха «кореспондентського навчання», безліч ВНЗ у всьому світі вели і ведуть його до сих пір.

Поява радіо і телебачення внесло зміни в дистанційні методи навчання. Це був значний прорив, аудиторія навчання зросла в сотні разів. Багато хто ще пам'ятає навчальні телепередачі, які йшли, починаючи з 50-х років. Однак у телебачення і радіо був істотний недолік - в учня не було можливості отримати зворотний зв'язок.

У 1969 році у Великобританії був відкритий перший в світі університет дистанційної освіти - Відкритий Університет Великобританії, він був названий так, щоб показати його доступність за рахунок невисокої ціни і відсутності необхідності часто відвідувати аудиторні заняття.[4]

Інші відомі університети з програмами дистанційного навчання за кордоном: University of South Africa (1946), FernUniversität in Hagen (Німеччина, 1974), Національний технологічний університет (США, 1984) (програми ДО з інженерних спеціальностей), Відкритий університет Хаген (Німеччина), INTEC-коледж Кейптауна (ПАР), Іспанська національний університет дистанційного навчання, Відкрита школа бізнесу Британського відкритого університету, Австралійська територіальна інформаційна мережа.

Кафедра_КбПЗ_2021рік

В кінці 1980-х доступність персональних комп'ютерів дала нову надію, пов'язану зі спрощенням і автоматизацією навчання. Комп'ютерні навчальні програми з'явилися на перших комп'ютерах у вигляді різних ігор.

У 1988 був реалізований американський проект «Шкільна електронна пошта».[5]

У XXI столітті доступність комп'ютерів та Інтернету робить дистанційне навчання ще більш простим, а його поширення більш швидким. Інтернет став величезним проривом, значно більшим, ніж радіо і телебачення. З'явилася можливість спілкуватися і отримувати зворотний зв'язок від будь-якого учня, де б він не знаходився. Поширення «швидкого інтернету» дало можливість використовувати онлайн-семінари (вебінари) для навчання.

У 2020 році через Коронавірусну Інфекцію або COVID-19 понад 1,5 мільярда учнів шкіл і ВНЗ переведені на дистанційне навчання, вперше за всю історію.

2.1.1 Системи дистанційного навчання

Moodle. На сьогоднішній день Moodle безсумнівно одна з найпопулярніших СДО з відкритим вихідним кодом.

Moodle пропонує користувачеві різні панелі інструментів, можливість відстежувати прогрес студентів і підтримку мультимедіа. Система дає можливість створювати курси, адаптовані під мобільні телефони, і досить доброзичливо ставиться до інтеграції доповнень від сторонніх розробників.

Для тих, хто хоче заробити на своїх курсах, Moodle має інтеграцію з платіжною системою PayPal, яка робить простим і зрозумілим процес оформлення замовлень і оплати.

Кафедра_КБПЗ_2021рік

Ще однією важливою перевагою Moodle є спільнота користувачів. На відміну від багатьох інших безкоштовних СДО, тут ви можете практично моментально отримати відповіді на більшість, що цікавлять вас питань, звернувшись до онлайн бази технічної підтримки.

Крім того, сервіс пропонує ряд готових шаблонів, якими ви можете скористатися, щоб заощадити час і не створювати курс з нуля.

Але на перший погляд нажаль це не реальність.А в реальності ця СДО є доволі складною, як для користувача так і для адміністрування.

Е-стадії - Електронна освітня середа. Онлайн-платформа для організації дистанційного навчання Е-стадії – безкоштовна розробка команди

однодумців з розвитку дистанційної освіти.

Для початку роботи необхідно зареєструватися на сайті і створити «робочу область» - персональний простір вашої компанії, в якому розміщуватимуться навчальні матеріали та завдання для ваших учнів.

Відмінність від класичних LMS полягає в тому, що функціонал орієнтований на практичну роботу. Е-стадії, швидко, дозволяє публікувати навчальні матеріали, але велика частина системи призначена для всілякої оцінки знань і тестування.

Е-Стадії містить велику кількість інструментів для організації навчання і тестування:

- "Робоча область". У робочій області публікуються навчальні матеріали з курсу, оголошення і завдання (кейси). Робоча область створюється викладачем / тьютором / менеджером з навчання і може об'єднувати кілька груп або курсів. Слухачі отримують доступ до робочої області за заявками;
- «Тест». Е-Стадії володіє потужним функціоналом для проведення тестування, тест можна створювати на сайті або імпортувати з * .docx попередньо оформивши відповідно до спеціальними правилами. Доступний докладний звіт про відповіді кожного тестованого. Є можливість обмежити

Кафедра_КОПЗ_2021рік

терміни, час, кількість спроб, можливість перемикатися між вікнами браузера;

- «Файл». Завантаження файлів / документів, який викладач потім може оцінити і прокоментувати. Журнал успішності генерується в робочій області автоматично, на базі створених завдань і дозволяє генерувати Excel-файл.

Е-Стадії це нова безкоштовна електронно-освітнє середовище, орієнтоване для організації навчання персоналу.

Переваги:

- не вимагає інсталяції / налаштування;
- система безкоштовна;
- проста у використанні;
- потужний функціонал для тестування і оцінки;
- не вимагає попередньої розробки курсів;
- є англійська версія.

Недоліки:

- неможливість самостійного доопрацювання;
- відсутність підтримки SCORM;
- обмежений, але достатній функціонал.

В цілому Е-Стадії заслуговує відмінної оцінки і є хорошим рішенням для невеликих компаній, що бажають організувати навчання персоналу без будь-яких витрат на придбання СДО.

ATutor. Ця система дистанційного навчання має безліч корисних функцій: від email-повідомлень до файлового сховища.

Одним з найбільш яскравих переваг ATutor є її клієнтоорієнтованість і легкий і зрозумілий інтерфейс, що робить дану систему ідеальним інструментом для тих, хто тільки починає освоювати світ електронного навчання.

Кафедра_КБПЗ_2021рік

Також Atutor пропонує користувачеві ряд попередньо встановлених тем, що дозволяють прискорити процес створення курсу. І не можна не відзначити різні інструменти оцінки, резервне копіювання файлів, ведення статистики і можливість інтеграції опитувань.

Але система є платною, без можливості допрацювання та переналаштовування.

Eliademy. Для викладачів і кураторів навчання дана система є повністю безкоштовною, невелика плата береться з користувачів, якщо ті захочуть скористатися перевагами преміум аккаунта.

Eliademy пропонує каталоги курсів електронного навчання, інструменти оцінки і навіть мобільний додаток для Андроїда для тих викладачів, які прагнуть розвивати мобільні курси і націлені на людей.

Forma LMS. Від аналізу загального рівня знань до детально статистики та звітності - Forma LMS може сподобатись досить об'ємним набором доступних функцій.

Сервіс також має різні сертифікати, компетентну підтримку керівництва, а також широкий спектр інструментів для управління віртуальною класною кімнатою, включаючи різні календарі і менеджери подій.

Ця система ідеально підходить для корпоративних програм навчання і пропонує доступ в активну онлайн спільноту, де ви зможете знайти безліч корисних порад про те, як отримати максимальну віддачу від даного сервісу.[6]

Dokeos. Якщо ви шукаєте систему дистанційного навчання з уже готовими елементами курсів, то Dokeos, що надається безкоштовно для груп до п'яти користувачів, для вас.

Ця система пропонує безліч готових шаблонів і курсів електронного навчання і звичайно ж авторські інструменти, за допомогою яких ви можете максимально скоротити час, витрачений на створення свого курсу.

Кафедра_КБПЗ_2021рік

На своєму веб-сайті розробники пропонують користувачеві масу корисної інформації, в тому числі і покрокові відео інструкції по створенню власних курсів. Інтуїтивно зрозумілий інтерфейс робить Dokeos відмінним варіантом для новачків у сфері електронного навчання та для тих, хто не хоче витрачати час на довгий вивчення інструкцій.

ILIAS. Цю систему дистанційного навчання можна назвати першою відкритою системою, яка відповідає таким стандартам систем дистанційного навчання, як SCORM 1.2 та SCORM 2004.

Ця гнучка універсальна система відповідає всім основним вимогам, необхідним для успішного продажу авторських курсів.

Слід зазначити, що ILIAS одна з небагатьох систем дистанційного навчання, яку можна використовувати, як повноцінну платформу для електронного навчання, завдяки можливості спілкування всередині команди і передачі і зберігання всіх документів. Система абсолютно безкоштовна для всіх організацій, що займаються електронним навчанням, незалежно від кількості користувачів.

Якщо у вас навчаються сотні, а то й тисячі, людей, ця система допоможе вам значно скоротити статті витрат, так як багато інших СДО призначають плату в залежності від кількості користувачів.

Основним недолік системи що вона є платною без можливості допрацювання.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для реалізації дипломного проекту було обрано такі засоби:

- Мова програмування PHP;
- Фреймворк для php, Laravel;

Кафедра_КбПЗ_2021рік

- Декларативна мова програмування для роботи с базою даних SQL.

Front-end частина виконана на

- Bootstrap;
- HTML;
- CSS.

Оскільки в короткій термін потрібно було реалізувати майже ОРМ систему було обрано скриптову мову програмування PHP. Ця мова має 2 популярні фреймворки Laravel та Cake. Laravel (Ларавел) - це безкоштовний оупенсорс (з відкритим вихідним кодом) PHP-фреймворк загального призначення. Універсальність CMS - це одночасно і плюс, і мінус. Так, ви можете розкачати на ній проект будь-якого типу і призначення - від новинного порталу до інтернет-магазину. Але за це доведеться заплатити істотну ціну. І мова не про гроші (хоча і про них трошки теж), а про те, що разом з CMS ви отримаєте «з коробки» масу інструментів, модулів, скриптів, які з високою часткою ймовірності вам ніколи не знадобляться, однак будуть постійно споживати ресурс .

Якщо ж мова йде про Laravel, то універсальність видається дещо в іншому, на фреймворку розробляються без застережень будь-які проекти. Можливості і функціональність компілюються або створюються індивідуально. У підсумку - нічого зайвого. Утилітарність, як вона є.

Такий підхід обумовлює два наступних переваги Laravel. Що б вам не говорили менеджери з продажу, при використанні CMS і постійному зростанні веб-ресурсу рано чи пізно зіткнення з обмеженнями системи неминуче. І так, грамотні розробники зможуть їх обійти або що-небудь ще придумати, щоб втілити потрібну замовнику функціональність. Але це складно і проблемно, а тому недешево і не завжди в рамках ідеології CMS.

Кафедра_КОПЗ_2021рік

При розробці на Laravel таких проблем немає. Проект легко масштабується в потрібному напрямку. З фреймворком доступні абсолютно будь-які впровадження: складні функції, сервіси. І все нові можливості додаються в рамках логіки фреймворка, без необхідності обходити спочатку закладені обмеження.

Кастомізація проекту на CMS багато в чому обмежується логікою системи. Ще складніше персоналізувати сайт, де впроваджені готові рішення. Але не при використанні PHP-фреймворку. Справа в тому, що навіть готові пакети створені так, що їх можна повністю розібрати і зібрати, використовувати з них тільки необхідне.

Наповнити ресурс потрібними функціями допомагають пакети - аналоги модулів в CMS. Бібліотека встановленого ПЗ знаходяться у вільному доступі. Автори постійно працюють над розвитком і підтримкою, тому знайти перевірене часом готове рішення, яке максимально підійде під вимоги бізнесу, не складе труднощів.

Фреймворк захищає веб-ресурси від двох основних небезпек в мережі:

- XSS-атаки (міжсайтовий скриптинг);
- SQL-ін'єкції.

Потенційно небезпечні html-теги екрануються і виводяться екрановані рядком чистого тексту, який неможливо виконати.

Власна ORM (технологія взаємодії з базою даних шляхом надання методів API для типових операцій: вибірка, додавання, оновлення, видалення і т. д.) Виключає передачу «сирих» SQL-запитів і нормалізує всі параметри при їх побудові. З них видаляється все, що в теорії може нашкодити. [7]

HTML - мова гіпертекстової розмітки. Абревіатура утворилася від перших букв англійських слів HyperText Markup Languge. HTML

Кафедра_КбПЗ_2021рік

застосовується для розмітки веб-сторінок. Вона потрібна браузерам, які перетворять гіпертекст і виводять на екран сторінку в зручному для людини форматі. [8]

CSS - мова опису стилів. Абревіатура утворилася з перших букв англійських слів Cascading Style Sheets - каскадні таблиці стилів. CSS описує зовнішній вигляд HTML-елементів. Тобто розробники за допомогою каскадних таблиць стилів визначають, як має виглядати той чи інший елемент на сторінці.

У європейському середовищі фахівця з HTML і CSS часто називають верстальником, а створення веб-сторінок за допомогою цих мов - версткою. В англомовному середовищі таких фахівців називають веб-дизайнерами.

Як зазначалося вище, верстальники використовують HTML і CSS для створення веб-сторінок, тобто сторінок в інтернеті. За даними We Are Social, в січні 2020 року в світі налічувалося 4,54 млрд користувачів інтернету. Це 59% від загального населення планети. Тобто потенціал зростання інтернет-користувачів обчислюється мільярдами людей.

Ці дані дозволяють припустити, що інтернет в доступному для огляду майбутньому буде зростати. Тобто будуть з'являтися нові сайти, а для створення цих сайтів знадобиться HTML і CSS. Очевидно, ці мови будуть розвиватися, а їх можливості будуть рости.

Веб-розробники, як фронтендери, так і бекендери, так чи інакше стикаються з версткою під час роботи. Не кожному веб-програмісту, особливо бекенд-розробнику, потрібно вміти верстати сторінки «з точністю до пікселя». Але без розуміння принципів HTML і CSS працювати в веб-розробці практично неможливо.

Тобто, для створення повноцінного порталу потрібна не тільки мова програмування, що буде відповідати за розмітку але й мови візуалізації якими є HTML та CSS.

Кафедра_КОПЗ_2021рік

Але працювати в чистому вигляді з ними не дуже зручно. На даний момент існує більш продвинута реалізація поєднання HTML, CSS та JS це фреймворк Bootstrap.

Bootstrap - це відкритий і безкоштовний HTML, CSS і JS фреймворк, який використовується веб-розробниками для швидкої верстки адаптивних дизайнів сайтів. Веб-фреймворк Bootstrap використовується по всьому світу не тільки незалежними розробниками, але іноді й цілими компаніями. На Bootstrap створено дуже багато різних сайтів, подивитися їх можна на сторінці Bootstrap Expo . [9]

Основна область його застосування - це фронтенд розробка сайтів та інтерфейсів адмінок . Серед аналогічних систем (Foundation, UIkit, Semantic UI, InK і ін.) Фреймворк Bootstrap є найпопулярнішим .

Чому Bootstrap такий популярний? Це пов'язано з тим, що він дозволяє верстати сайти в кілька разів швидше , ніж на «чистому» CSS і JavaScript. А в нашому світі, час - це дуже цінний ресурс. Ще один його аспект - доступність. Вона зводиться до того, що надає можливість навіть початківцю веб-розробнику (без глибоких знань і достатньої практики) створювати досить якісні макети .

Фреймворк Bootstrap - це набір CSS і JavaScript файлів . Щоб його використовувати ці файли необхідно просто підключити до сторінки . Після цього вам стануть доступні інструменти даного фреймворку: блочна система (сітка Bootstrap), класи і компоненти додатків.

Проаналізувавши потреби порталу для кваліфікаційної роботи було обрано вищеперелічені мови програмування, фреймворки, та технології.

Кафедра_КоПЗ_2021рік

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на кваліфікаційну бакалаврську роботу, реалізації підлягає програмне забезпечення, яке призначено для дистанційного навчання контролю якості програмних застосунків.

В процесі розробки кваліфікаційної бакалаврської роботи необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

Кафедра_КОПЗ_2021рік

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Сайти містять різну статику, яка як і HTML-файл не генерується на льоту. Найчастіше це картинки, CSS-файли, JS-скрипти, але можуть бути і будь-які інші файли: mp3, mov, csv, pdf.

Блоги, візитки з формою для контакту, лендінги з купою ефектів я теж відношу до простих сайтів. Хоча на відміну від зовсім статичних сайтів, вони вже включають в себе якусь бізнес-логіку.

А веб-додаток - це щось технічно більш складне. Тут HTML-сторінки генеруються на льоту в залежності від запиту користувача. Поштові клієнти, соцмережі, пошукові системи, інтернет-магазини, онлайн-програми для бізнесу, це все веб-додатки.

Веб-додатки можна розділити на кілька типів, в залежності від різних поєднань його основних складових:

- Backend (бекенд або серверна частина програми) працює на віддаленому комп'ютері, який може знаходитися де завгодно. Вона може бути написана на різних мовах програмування: PHP, Python, Ruby, C# та інших. Якщо створювати додаток використовуючи тільки серверну частину, то в результаті будь-яких переходів між розділами, відправок форм, оновлення даних, сервером буде генеруватися новий HTML-файл і сторінка в браузері перезавантажуватиметься;
- Frontend (фронтенд або клієнтська частина програми) виконується в браузері користувача. Ця частина написана на мові програмування Javascript. Додаток може складатися тільки з клієнтської частини, або ускладнено, де потрібно зберігати дані користувача довше однієї сесії. Це можуть бути, наприклад, фоторедактори або прості іграшки;

Кафедра_КбПЗ_2021рік

- Single page application (SPA або односторінкове додаток). Більш цікавий варіант, коли використовуються і бекенд і фронтенд. За допомогою їх взаємодії можна створити додаток, який буде працювати зовсім без перезавантажень сторінки в браузері. Або в спрощеному варіанті, коли переходи між розділами викликають перезавантаження, але будь-які дії в розділі обходяться без них.

У розробці фреймворк - це набір готових бібліотек і інструментів, які допомагають створювати веб-додатки. Для прикладу опишу принцип роботи фреймворка Laravel, написаного на мові програмування PHP.

Першим етапом запит від користувача потрапляє в роутер (URL dispatcher), який вирішує яку функцію для обробки запиту треба викликати. Рішення приймається на основі списку правил, що складаються з регулярного виразу і назви функції: якщо такий-то урл, то ось така функція.

Функція, яка викликається роутером, називається в'ю (view). Всередині може міститися будь-яка бізнес-логіка, але найчастіше це одне з двох: або з бази беруться дані, готуються і повертаються на фронт; або прийшов запит з даними з якоїсь форми, ці дані перевіряються і зберігаються в базу.

Дані програми зберігаються в базі даних (БД). Найчастіше використовуються реляційні БД. Це коли є таблиці з наперед заданими колонками і ці таблиці пов'язані між собою через одну з колонок.

Дані в БД можна створювати, читати, змінювати і видаляти. Іноді для позначення цих дій можна зустріти абревіатуру CRUD (Create Read Update Delete). Для запиту до даних в БД використовується спеціальна мова SQL (structured query language).

У Laravel для роботи з БД використовуються моделі (model). Вони дозволяють описувати таблиці і робити запити у звичному для розробника форматі, що набагато зручніше. За це зручність доводиться платити: такі

Кафедра_КОПЗ_2021рік

запити повільніше і обмежені в можливостях у порівнянні з використанням чистого SQL.

Отримані з БД дані готуються у в'ю до відправки на фронт. Вони можуть бути підставлені в шаблон (template) і відправлені в вигляді HTML-файлу. Але в разі односторінкового додатки це відбувається всього один раз, коли генерується HTML-сторінка, на який підключаються всі JS-скрипти. В інших випадках дані серіалізуются і відправляються в JSON-форматі.

MVC - скорочення від „Model View Controller”. Він представляє архітектуру, яку розробники приймають під час створення додатків. В архітектурі MVC ми розглядаємо структуру програми щодо того, як працює потік даних в ній.

MVC - це програмна архітектура ... яка відокремлює логіку домену / програми / бізнесу ... від решти користувальницького інтерфейсу. Це робиться шляхом поділу програми на три частини: модель, вигляд та контролер.

Модель керує фундаментальною поведінкою та даними програми. Вона може реагувати на запити інформацію, реагувати інструкції щодо зміни стану своєї інформації і навіть сповіщати спостерігачів у системах, керованих подіями, коли інформація змінюється. Це може бути база даних або будь-яка кількість структур даних або систем зберігання. Це дані та управління даними програми.

Вигляд ефективно забезпечує елемент інтерфейсу користувача програми. Дані з моделі відображатимуться у формі, яка підходить для інтерфейсу користувача.

Контролер отримує введені користувачем дані та здійснює виклики модельних об'єктів та подання для виконання відповідних дій.

Загалом, ці три компоненти працюють разом, щоб створити три основні компоненти MVC.

Кафедра_КбПЗ_2021рік

Для більшої наглядності та зрозумілості пропоную ознайомитись з Рисунку 3.1:

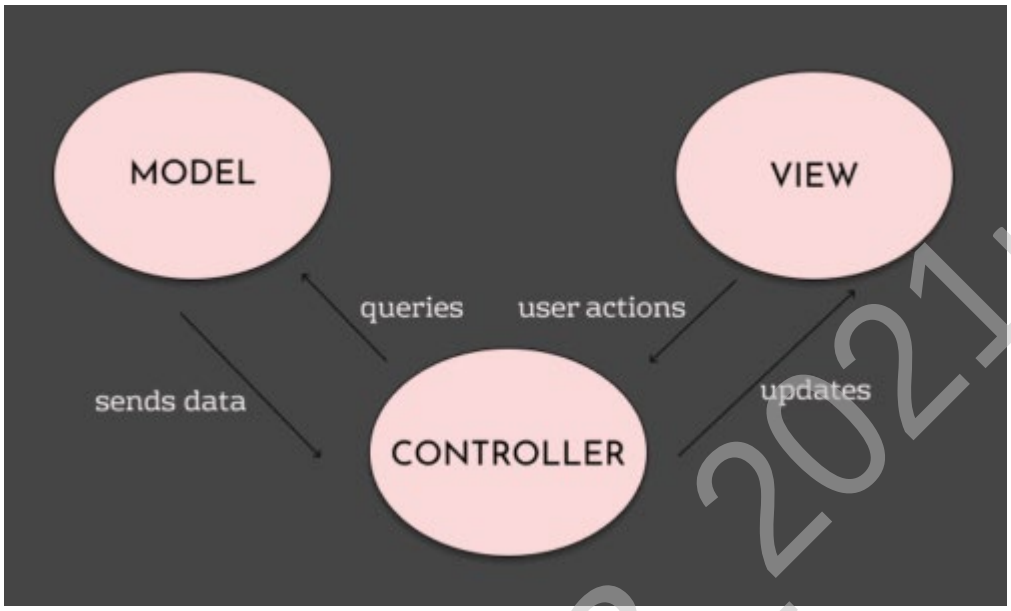


Рисунок 3.1 - Діаграма, що ілюструє концепції MVC

Модель - це представлення реального екземпляру або об’єкта в нашій базі коду. Вид представляє інтерфейс, за допомогою якого користувач взаємодіє з нашим додатком. Коли користувач виконує дію, контролер обробляє дію та при необхідності оновлює Модель.

Якщо ви перейдете на веб-сайт електронної комерції, різні сторінки, які ви бачите, надаються як перегляд. Коли ви натискаєте певний продукт, щоб переглянути більше, шар Controller обробляє дії користувача. Це може включати отримання даних із джерела даних за допомогою рівня Модель. Далі дані об’єднуються, розташовуються в рівні View і відображаються користувачеві. [10]

Коли створюється новий проект Laravel (його можливо створити, запустивши команду laravel new project-name), проект має таку структуру як зображено на Рисунку 3.2:

Кафедра_КБПЗ_2021рік

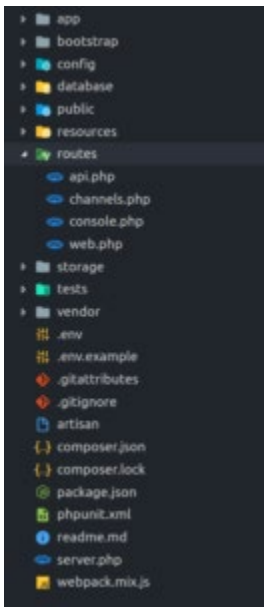


Рисунок 3.2 - Структура проекту Laravel

У каталозі маршрутів / головного каталогу є файл web.php. У цьому файлі ви обробляєте запити, коли користувачі відвідують додаток. Файл виглядає так:

```
<? php
  / *
  | ---------------------------------------------- ---------------
---------
  | Веб-маршрути
  | ---------------------------------------------- ---------------
---------
  |
  | [...]
  |
  * /
  Route :: get ('/', function () {
      повернення ("ласкаво просимо");
  });
```

У цьому файлі можливо направляти URL-адреси до контролерів у програмі, наприклад, перехід відбувається коли коричтувач взаємодіє зі сторінками порталу, роут перенаправляє користувача на потрібну сторінку.

Кафедра_КбПЗ_2021рік

Коли користувач тисне на Home роут перенаправляє його на головну сторінку /index.

Для того щоб побачити роботу MVC моделі у фреймворку, потрібно створити базовий проект.

Щоб створити новий проект, запустіть команду нижче у своєму терміналі:

```
$ laravel new qamentoring
```

Щоб побачити роботу початкової програми, необхідно оновити APP_URL у .env-файлі до http: // localhost: 8000, а потім запустити у терміналі команду:

```
$ php artisan serv
```

Усю роботу з проектом необхідно роботи у корені проекту, тому перед роботою з консолью потрібно удостовіритись що користувач знаходиться у корені проекту.

Після запуску початкової програми порталу, побачити її вид можна буде за адресою 127.0.0.1:8000. Головний єкран пустого проекту просто вітає користувача, тому потрібно повернутись до роботи з проектом а саме натиснути Ctrl+C.

Для роботи з базою потрібно створити Model. У Laravel модель є класом із властивостями, які відповідають стовпцям бази даних.

У базі порталу студент матиме такі властивості:

- id - особливий ідентифікатор студента;
- fullname – призвіще та ім’я студента;
- email – емейл адреса;
- password – токен паролю;
- level – поточний рівень;
-status – статус готовності домашнього завдання.

```
$ php artisan make: model User
```

Кафедра_КОПЗ_2021рік

Після запуску цієї команди Laravel створить файл User.php у каталозі програми. Це буде клас PHP з назвою User, і це буде моделлю таблиці студентів у базі даних.

У моделі User потрібно додати властивість $ fillable:

```
<?php

 namespace App;

 use Illuminate\Database\Eloquent\Model;

 class Product extends Model {

     protected $fillable = [
        'id’,
        'fullname',
        'email',
        ‘password',
‘level’,
‘status’,
        ];
 }?
```

Властивість заповнення використовується для представлення масових призначуваних атрибутів для моделі.

Міграція дозволяє розробнику вносити та скасовувати зміни до бази даних проекту. Міграцію можна використовувати, щоб зробити управління базами даних простим та передбачуваним. Щоб створити міграцію, необхідно наступне запустити в терміналі:

```
$ php artisan make: переміщення create_user_table
```

Коли команда виконується, користувач бачить новий файл * _create_products_table.php у каталозі бази даних / міграцій, і також він може редагувати його, щоб мати схему таблиці користувачів, як приклад:

Кафедра_КбПЗ_2021рік

```
<?php

    [...]

    class CreateuserTable extends Migration
    {
        public function up()
        {
            Schema::create('products', function (Blueprint $table)
{
                $table->increments('id');
                $table->string('name');
                $table->text('description');
                $table->integer('count');
                $table->integer('price');
                $table->softDeletes();
                $table->timestamps();
            });
        }

        public function down()
        {
            Schema::dropIfExists('user');
        }
    }
```

Файл міграції готовий, але тепер потрібно перевірити чи налаштована база даних. Раніше було описано що для реалізації бази було обрано MySQL.

Для того щоб далі працювати над проектом необхідно встановити базу даних. Також необхідно створити порожній файл бази даних / database.mysql.

Потім необхідно скопіювати файл .env.example у кореневій частині проекту в .env, а потім у скопійованому файлі замінити такі рядки:

```
DB_CONNECTION=mysql
    DB_DATABASE=homestead
    DB_USERNAME=username
    DB_PASSWORD=password

  DB_DATABASE=/full/path/to/database.mysql
```

Кафедра_КбПЗ_2021рік

Це все для налаштування бази даних. Тепер для запуску міграцій необхідно запустити команду нижче у терміналі:

```
$ php artisan migrate
```

Перед запуском команди потрібно налаштувати базу даних та встановити з'єднання деталі знаходяться у файлі .env у кореневій частині проекту.

Раніше вже згадувалось, що контролери відповідають за виконання дій користувача та управління бізнес-логікою додатків. Для дипломного проекту запропоновано використання Resource Controller.

Маршрутизація ресурсів Laravel призначає типові маршрути “CRUD” контролеру з одним рядком коду. Наприклад, можливо створити контролер, який обробляє всі HTTP-запити на "фотографії", що зберігаються у додатку.

Щоб створити контролер ресурсів у Laravel, необхідно виконати таку команду:

```
$ php artisan make:controller ProductController -r
```

Прапор -r робить його контролером ресурсів і таким чином створює всі методи, необхідні для роботи з CRUD.

Після запуску команди Laravel необхідно створити новий файл у каталозі app/Http/Controllers з назвою UserController.php.

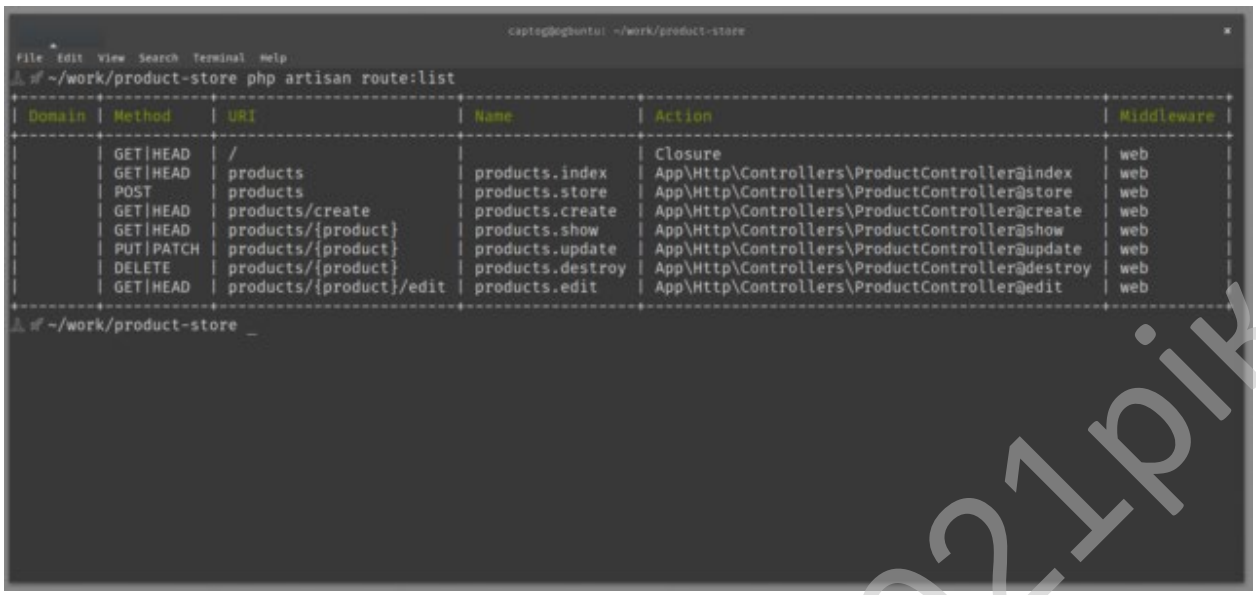
Перш ніж писати логіку до контролера, необхідно перейти до файлу routes / web.php і додати маршрут:

```
Route::resource('/dashboard', 'UserController');
```

Це говорить Laravel створити всі маршрути, необхідні для контролера ресурсів, і зіставити їх із класом UserController.

Для того щоб побачити всі можливі маршрути, необхідно запустити команду php artisan route: list у терміналі. Користувач може бачити такий результат як на Рисунку 3.3, що показує маршрути та типи запитів, які слід використовувати при доступі до маршрутів.

Кафедра_КбПЗ_2021рік



Domain	Method	URI	Name	Action	Middleware
	GET|HEAD	/		Closure	web
	GET|HEAD	products	products.index	App\Http\Controllers\ProductController@index	web
	POST	products	products.store	App\Http\Controllers\ProductController@store	web
	GET|HEAD	products/create	products.create	App\Http\Controllers\ProductController@create	web
	GET|HEAD	products/{product}	products.show	App\Http\Controllers\ProductController@show	web
	PUT|PATCH	products/{product}	products.update	App\Http\Controllers\ProductController@update	web
	DELETE	products/{product}	products.destroy	App\Http\Controllers\ProductController@destroy	web
	GET|HEAD	products/{product}/edit	products.edit	App\Http\Controllers\ProductController@edit	web

Рисунок 3.3 - Список маршрутів Laravel

Звернувши увагу на додане зображення можливо побачити дію, на яку позначається кожен URI. Це означає, що коли користувач переходить на 127.0.0.1:8000/dashboard/create, функція створення в UserController обробляє запит користувача.

Необхідно додти до файлу контролеру методи з початковою логікою:

Метод створення:

```
public function create()

    {
        return view('createuser');
    }
```

Вищезазначений для створення (C у CRUD). Контролер завантажує подання (V у MVC), яке називається create user, і служить відповіддю на будь-який час, коли хтось відвідує маршрут /dashboard/create за допомогою методу GET HTTP.

Кафедра_КБПЗ_2021рік

Спосіб магазину:

```
public function store(Request $request) {
        \App\Product::create([
          'name' => $request->get('name'),
          'description' => $request->get('description'),
          'price' => $request->get('price'),
          'count' => $request->get('count'),
        ]);

        return redirect('/products');
    }
```

Метод store викликається, коли користувач надсилає запит POST HTTP до кінцевої точки /user. Ця логіка вище отримує дані із запиту та зберігає їх у базі даних за допомогою моделі User.

Індексний метод:

```
  public function index()
    {
        $products = \App\Product::all();

        return view('viewproducts', ['allProducts' => $products]);
    }
```

Індексний метод викликається, коли / user маршрут завантажується методом GET HTTP. У цьому методі отримуютьсявсі дані, доступні в таблиці користувачів, використовуючи модель User і передаючи їх у подання як змінну. Це означає, що у поданні буде доступна змінна $ allUseers.

У Laravel всі подання зберігаються в каталозі ресурси/подання. У визначених поданнях зазвичай зберігається HTML-код створенної сторінки та є презентаційним рівнем архітектури MVC.

Для створення сторінки перегляду необхідно виконати ряд дій. Оновити файл welcome.blade.php у каталозі resources/views, щоб включити такий код всередину тегу основного HTML:

Кафедра_КБПЗ_2021рік

```
[...]

    <div class="flex-center position-ref full-height">
      <div class="content">
        <div class="title m-b-md">Product Store</div>
        <div class="links">
          <a  href="{{  config('app.url')}}/products/create">Create
Product</a>
          <a      href="{{      config('app.url')}}/products">View
Products</a>
        </div>
      </div>
    </div>

    [...]
```

Laravel використовує Blade як шаблонний механізм. Blade - це майже HTML, але з деяким ін'єкційним PHP-подібним синтаксисом.

Якщо повернутись до маршрутів / web.php, користувач побачить, що там зазначено, що привітальний вигляд повинен бути наданий користувачеві під час відвідування/. Якщо відвідати URL-адресу веб-сторінки http://127.0.0.1:8000/, можливо побачити цю сторінку як на Рисунку 3.4:

Рисунок 3.4 – Стандартний вигляд головної сторінки Laravel

Кафедра_КБПЗ_2021рік

Для подальшої роботи необхідно створити View “User”. Створити файл dashboarduser.blade.php у каталозі ресурси/подання нашого проекту. Туди додати:

```
<!doctype html>
    <html lang="{{ app()->getLocale() }}">
    <head>
      <title>Create Product | Product Store</title>
      <!-- styling etc. -->
    </head>
    <body>
        <div class="flex-center position-ref full-height">
            <div class="content">
                <form method="POST" action="{{
config('app.url')}}/products">

    <h1> Enter Details to create a product</h1>
                    <div class="form-input">
                        <label>Name</label> <input type="text"
name="name">
                    </div>

                    <div class="form-input">
                        <label>Description</label> <input
type="text" name="description">
                    </div>

                    <div class="form-input">
                        <label>Count</label> <input type="number"
name="count">
                    </div>

                    <div class="form-input">
                        <label>Price</label> <input type="number"
name="price">
                    </div>

                    <button type="submit">Submit</button>
                </form>
            </div>
```

Кафедра_КбПЗ_2021рік

```
</div>
 </body>
 </html>
```

Це проста форма для того щоб редагувати користувачів. Коли форму надіслано, надсилається запит POST до маршруту /dashboard програми, який обробляється методом store у нашому UserController.

Ось як виглядатиме маршрут /user/dashboard після додавання View та відвідування маршруту:

Наступним View, який необхідно додати, є view viewprogressdashboard.blade.php. Його також необхідно створити у каталозі resources/views, та додати до нього код:

```
<!doctype html>
  <html lang="{{ app()->getLocale() }}">
  <head>
      <title>View Products | Product Store</title>
      <!-- Styles etc. -->
  </head>
  <body>
      <div class="flex-center position-ref full-height">
          <div class="content">
              <h1>Here's a list of available products</h1>
              <table>
                  <thead>
                      <td>Name</td>
                      <td>Description</td>
                      <td>Count</td>
                      <td>Price</td>
                  </thead>
                  <tbody>
                      @foreach ($allProducts as $product)
                          <tr>
                              <td>{{ $product->name }}</td>
                              <td class="inner-table">{{
$product->description }}</td>
                              <td class="inner-table">{{
$product->count }}</td>
```

Кафедра КБПЗ 2021 рік

```
<td class="inner-table">{{ $product->price }}</td>
                                </tr>
                            @endforeach
                        </tbody>
                    </table>
                </div>
            </div>
        </body>
        </html>
```

3.2 Розробка структурної схеми

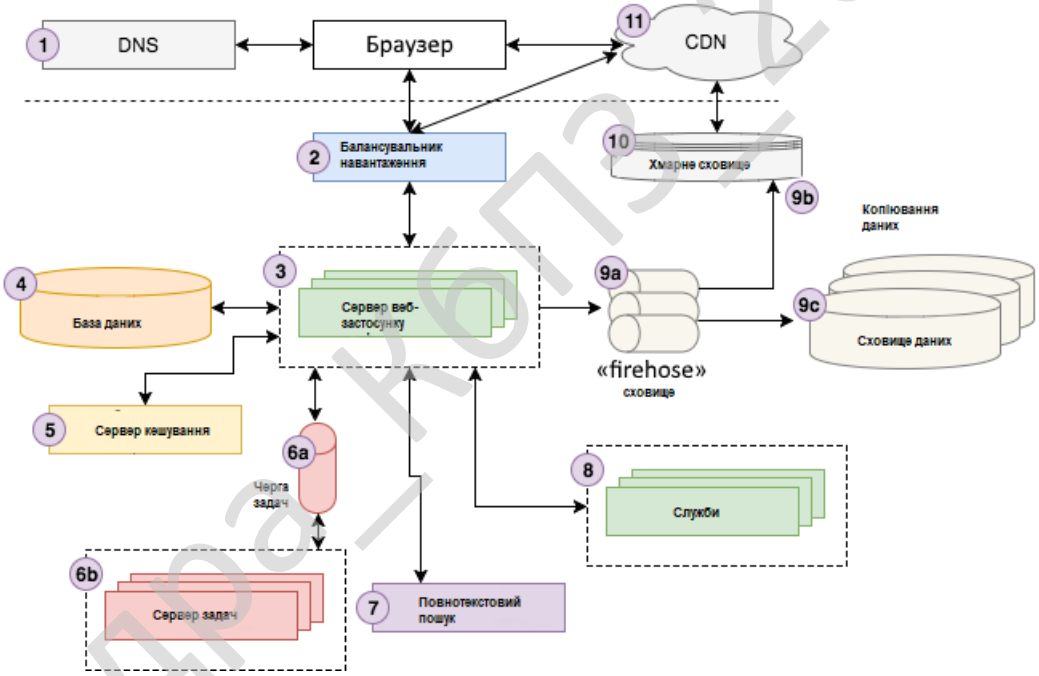


Рисунок 3.5 – Структурна схема роботи веб-застосунку

Для того щоб більш детально ознайомитись з роботою веб-застосунку необхідно розглянути Рисунок 3.5. Користувач шукає в Google «Strong Beautiful Fog And Sunbeams In The Forest». Перший результат відправляє його на Storyblocks. Користувач натискає на посилання, яка перенаправляє його браузер на сторінку з зображенням. Тим часом браузер відправляє запит на

Кафедра_КБПЗ_2021рік

DNS-сервер, щоб встановити з'єднання зі Storyblock, і, отримавши відповідь, відправляє запит на сайт.

Запит потрапляє на балансувальник навантаження, який випадковим чином вибирає для обробки запиту один з десяти (або близько того) веб - серверів, на яких запущено сайт. Веб-сервер дістає деяку частину інформації про зображення зі служби кешування, а решта - з основної бази даних. Якщо колірний профіль для зображення ще не був обчислений, завдання «визначити колірний профіль» буде додана в чергу завдань. Ці завдання сервери обробляють асинхронно і відповідним чином оновлюють базу даних з результатами.

Потім відбувається пошук схожих фотографій: в службу повнотекстового пошуку відправляється запит з заголовком фотографії в якості вхідних даних. Якщо користувач авторизувався в Storyblocks, інформація про його облікового запису завантажується з бази даних облікових записів. Нарешті, дані про перегляд сторінки відправляються в firehose-сховище для подальшого запису в хмарну систему зберігання. В остаточному підсумку, ці дані будуть завантажені в сховище даних, яке аналітики використовують для обробки.

Тепер сервер рендерить сторінку в HTML і відправляє її назад браузеру користувача, проходячи спочатку через балансувальник навантаження. Сторінка містить Javascript- і CSS-файли, які завантажені в хмарну систему зберігання, підключену до CDN, тому браузер зв'язується з CDN для отримання вмісту. Нарешті, браузер відображає сторінку користувачеві.

DNS

DNS розшифровується як «Domain Name System» (система доменних імен). Це базова технологія, яка робить можливою роботу інтернету. На самому базовому рівні DNS забезпечує пошук пари з доменного імені та IP-адреси (наприклад, google.com і 85.129.83.120), що дозволяє комп'ютеру відправити запит на відповідний сервер. За аналогією з телефонними

Кафедра_КбПЗ_2021рік

номерами різниця між доменним ім'ям і IP-адресою така ж, як і між викликом Джону Доу і дзвінком за номером 201-867-5309. Для пошуку номера Джона потрібна телефонна книга, а для пошуку IP-адреси домену - DNS. Таким чином, можна вважати DNS телефонною книгою інтернету.

Балансувальник навантаження

Перш ніж почати обговорення балансування навантаження, потрібно зробити крок назад, щоб обговорити горизонтальне і вертикальне масштабування додатків. Що це і в чому різниця? Ця тема добре пояснена в пості на StackOverflow: «горизонтальне» масштабування характеризується додаванням нових машин в пул ресурсів, тоді як «вертикальне» має на увазі, що нарощується потужність (наприклад, збільшується CPU або RAM) існуючої машини.

У веб-розробці проект масштабується горизонтально як мінімум тому, що все ламається. Сервери падають з незрозумілих причин. Мережі деградують. У деяких ЦОД-ах час від часу відключається світло. Кілька серверів дозволить переживати незаплановані відключення без порушення роботи програми. Іншими словами, додаток стає «ВІДМОВОСТІЙКИМ». Горизонтальне масштабування дозволяє мінімально пов'язувати різні частини проекту (веб-сервер, базу даних і т. Д.), Тому що кожна з них запускається на різних серверах. Нарешті, може наступити такий момент, коли вертикальне масштабування більш неможливо, так як в світі немає достатньо потужного комп'ютера для виконання всіх обчислень. Пошукова платформа Google є типовим прикладом, хоча це відноситься і до компаній з набагато меншими масштабами. Наприклад, Storyblocks зазвичай використовує від 150 до 400 AWS-машин EC2 в будь-який момент часу. Було б складно отримати всю цю обчислювальну потужність за допомогою вертикального масштабування.

Давайте повернемося до балансувальник навантаження. Завдяки їм можливо горизонтальне масштабування. Вони направляють вхідні запити на

Кафедра_КОПЗ_2021рік

один з безлічі серверів додатків, які зазвичай є дзеркальними копіями один одного, і відправляють відповідь назад користувачеві. Будь-сервер обробляє запити однаково, так що балансувальник займається розподілом завдань, щоб жоден із них не був перевантажений.

От і все. Концептуально балансувальник навантаження досить прості і зрозумілі.

Сервери веб-додатків

Якщо дивитися здалеку, сервери веб-додатків відносно прості. Вони виконують основну бізнес-логіку, яка обробляє запит користувача і відправляє HTML назад браузеру. Щоб виконувати свою роботу, вони зазвичай спілкуються з різними бекенда-інфраструктурами, такими як бази даних, сервери кешування, черги завдань, служби пошуку і т. д. Як згадувалось вище, зазвичай є як мінімум два, а то і більше, сервера, підключених до балансувальник навантаження для обробки запитів користувачів.

Для реалізації сервера додатків потрібно вибрати конкретну мову (Node.js, Ruby, PHP, Scala, Java, C #, .NET і т. Д.) І MVC-фреймворк для цієї мови (Express для Node.js, Ruby on Rails, Play для Scala, Laravel для PHP і т. д.).

Сервер баз даних

Кожен сучасний веб-додаток використовує одну або кілька баз даних для зберігання інформації. Бази даних надають інструменти для організації, додавання, пошуку, оновлення, видалення та виконання обчислень над даними. У більшості випадків сервери веб-додатків безпосередньо спілкуються з серверами завдань. Крім того, у кожної серверної служби може бути відповідна база даних, ізольована від решти частини програми.

Тут варто згадати SQL і NoSQL.

Кафедра_КбПЗ_2021рік

SQL розшифровується як «Structured Query Language» (мова структурованих запитів). Він був винайдений в 1970-х роках, щоб створити стандартний спосіб запитів до реляційних наборам даних, доступних широкій аудиторії. SQL-бази даних зберігають дані в таблицях, які пов'язані між собою загальними ключами. Такі ключі зазвичай представлені цілими числами.

Розглянемо типовий приклад зберігання інформації про історію адрес користувачів. Вийдуть дві таблиці - user і user_addresses, - пов'язані один з одним ідентифікатором користувача (id в таблиці user). Їх можна побачити на зображенні нижче. Таблиці пов'язані, тому що стовпець user_id в user_addresses є «зовнішнім ключем» в стовпці id в таблиці users.

Ця технологія використовується в веб-розробці майже всюди, тому варто хоча б розуміти основи для правильної побудови додатків.

NoSQL розшифровується як "не-SQL» і являє собою більш новий набір технологій баз даних. Він був розроблений для обробки дуже великих обсягів інформації, які можуть генеруватися великомасштабними веб-додатками. Більшість варіантів SQL погано масштабуються горизонтально, а масштабуватись вертикально можуть тільки до певного моменту. Бажано перед розробкою ознайомитись з терміном NoSQL.

Також хочеться відзначити, що індустрія повсюдно покладається на SQL навіть як на інтерфейс для баз даних NoSQL, тому потрібно вивчати SQL, навіть якщо він не потрібно у роботі. У сучасних реаліях майже неможливо цього уникнути.

Кешування

Служба кешування надає просте сховище даних в форматі ключ-значення, яке дозволяє зберігати і шукати інформацію за час, близький до лінійного (O (1)). Зазвичай додатки використовують функції кешування, щоб зберігати результати дорогих обчислень і скористатися ними пізніше з кешу, а не перераховувати їх ще раз. Додаток може кешувати результати запиту в

Кафедра_КбПЗ_2021рік

бази даних, результати звернення до зовнішніх службам, HTML для заданого URL-адреси і багато іншого. Ось деякі приклади з реального світу:

- Google кешує результати пошуку для популярних пошукових запитів, таких як «собака» або «Тейлор Свіфт», а не шукає їх щоразу заново;
- Facebook кеширует більшу частину даних, які ви бачите при вході в систему, наприклад, списки постів або друзів. Детальніше, про технології кешування використовується в Facebook, можна почитати в цій статті на Medium;
- Storyblocks кеширует HTML-сторінки від React, результати пошуку та інше.

Двома найбільш поширеними технологіями кешування є Redis і Memcache.

Черги завдань

Більшості веб-додатків потрібно виконувати деяку роботу, безпосередньо не пов'язану з відповіддю на запити користувачів, асинхронно, в фоновому режимі. Наприклад, Google повинен сканувати та індексувати весь інтернет, щоб повертати релевантні результати пошуку. Він не робить це при кожному запиті, а сканує мережу асинхронно, оновлюючи пошукові індекси «по дорозі».

Виконувати асинхронну роботу дозволяють різні архітектури, але найбільш поширеною є архітектура «черга завдань». Вона складається з двох компонентів: черги «завдань», які необхідно виконати, і одного або декількох робочих серверів (часто званих «працівниками»), які обробляють завдання з черги.

У чергах завдань зберігаються списки завдань, які потрібно виконати асинхронно. Щоразу раз, коли з додатком потрібно виконати якесь завдання, яке повинно виконуватися за розкладом або відповідно до дій користувача, воно додає її в чергу. Найпростіше організовані черги FIFO - «першим

Кафедра_КоПЗ_2021рік

прийшов - першим пішов», але більшість програм в кінцевому підсумку потребують якоїсь системі балансування черг.

Наприклад, в Storyblocks чергу завдань використовується для виконання багатьох прихованих робіт, необхідних для підтримки проекту: кодування відео і фотографій, обробка CSV-файлів.[11]

3.3 Розробка функціональної схеми

У веб застосунку існує 2 суті: вчитель та учень. Для того щоб запланувати навчання вчитель має встановити дату у особистому кабінеті. Встановлена дата буде відображатись на головній сторінці , а також учням у особистому кабінеті після авторизації.

Кафедра_КбПЗ_2021рік

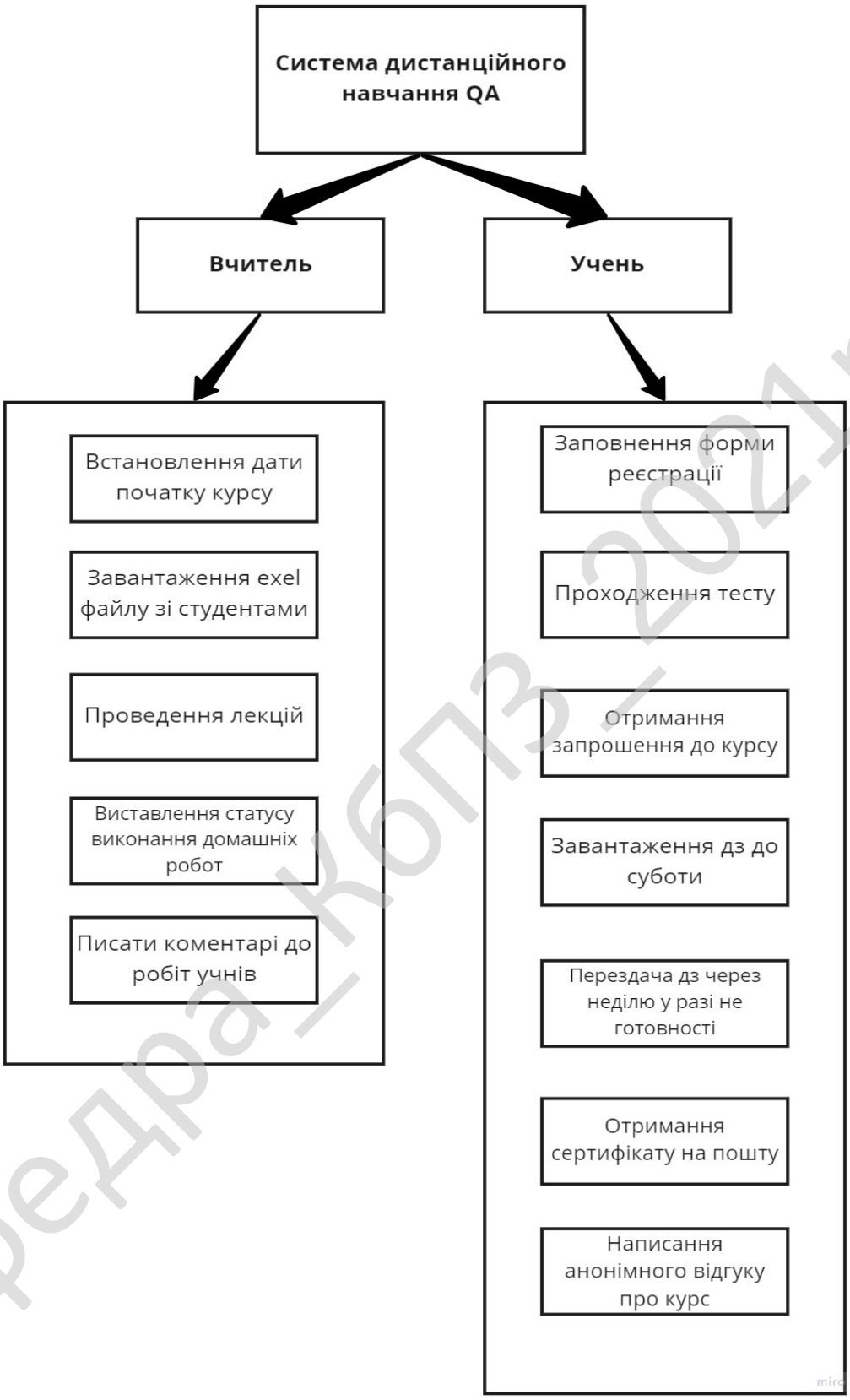


Рисунок 3.6 – Функціональна схема роботи ПЗ

Кафедра_КбПЗ_2021рік

Як можна побачити з Рисунку 3.6 портал складається з 2 блоків.

Портал має такий шлях авторизації та запровадження курсу:

1. Вчитель встановлює дату;
2. Учень на головній сторінці бачить коли буде наступний набір;
3. Учень заповнює форму на навчання;
4. На вказану учнем пошту надсилається лист на проходження вступного тесту;
5. За день до початку навчання вчитель змінює дату набору та зупиняє можливість проходження тесту;
6. Вчитель встановлює кількість учнів у групі та завантажує exel файл з результатами проходження тесту;
7. Імпортований документ сортується за результатами тесту та обирається перша кількість учасників з найвищими балами.
8. На пошту учні отримують запрошення на авторизацію;
9. Після заповнення особистих даних та даних авторизаціїї учні потрапляють на сторінку навчання;
10. Інформація з додатковими матеріалами виходить кожну субботу;
11. Учні мають тиждень на те щоб прослухати лекцію, ознайомитись з матеріалами та виконати дз.
12. Учні які виконали всі дз отримують сертифікат на пошту;
13. Учні які не завантажили дз вчасно отримують попередження (кожен учень має попередження).

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі

Кафедра_КОПЗ_2021рік

3.4 Розробка діаграми процесів

Відповідно до методичних рекомендацій розроблення графічної частини кваліфікаційної бакалаврської роботи розглянемо розроблену діаграму процесів яка зображена на Рисунку 3.7.

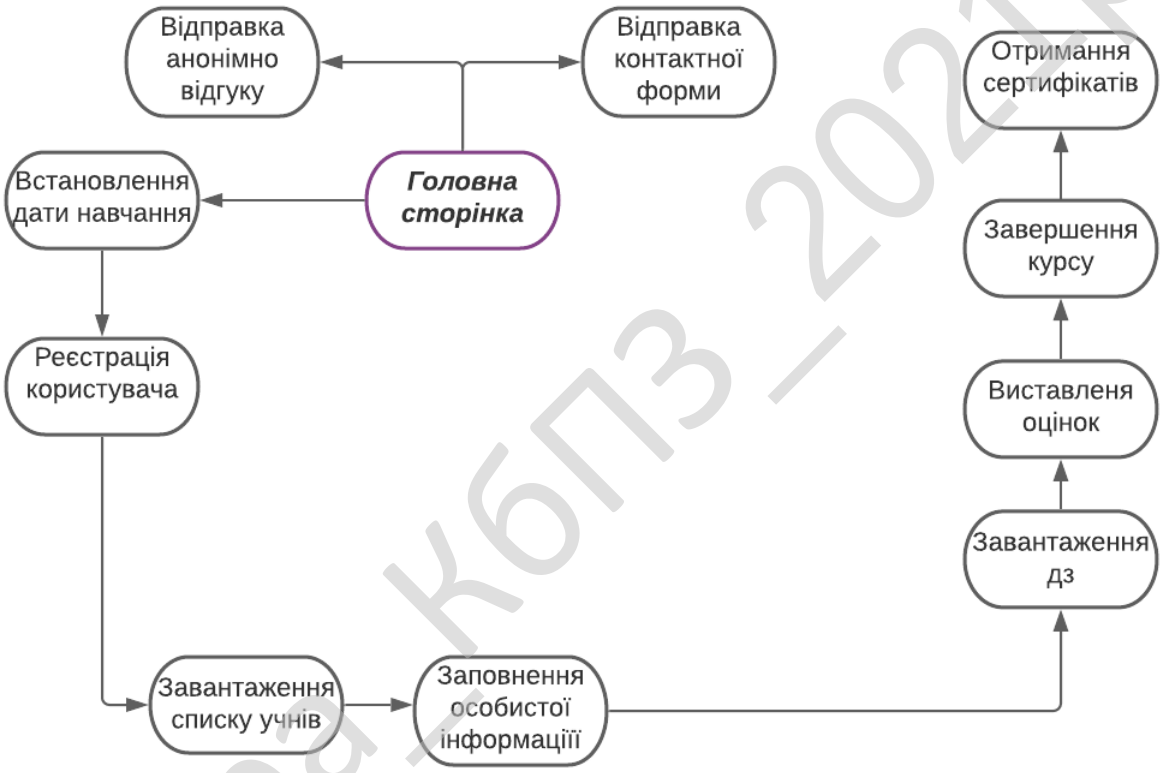


Рисунок 3.7 – Структурна схема роботи веб-застосунку

Загалом потрапивши на сторінку порталу у людини є 3 варіанти роботи з порталом:

- як звичайний користувач;
- як абітурієнт;
- як вчитель.

Кафедра_КбПЗ_2021рік

Звичайний користувач може оглядати лендінг, записуватись на курс але не проходити тестування. У не авторизованого користувача є можливість працювати з основним фукціоналом без реєстрації. Його дії не фіксуються в бд, та він ні як не взаємодіє з процесом навчання.

Абітурієнт, це користувач який потрапив на лендінг сторінку, але зареєструвався і пройшов тестування. У разі вдалого проходження тестування він отримує власний кабінет та доступ до матеріалів курсу. Після виконання усіх обов'язкових вимог він отримує сертифікат про проходження курсу та знання Junior QA.

Вимоги до студентів під час проходження курсу:

- Завантажувати домашнє завдання до вечора суботи;
- У разі винекнення питань звязатись з викладачем чи адміністратором курсу;
- Не розповсюджувати матеріали курсу.

Для студента в БД створена таблиця. Головний ключ приєднання це айдішнік учня. Також для більш зручної роботи з сутністю учень існує модель яка і працює з БД. Реалізація та структура таблиці також описані в міграції з БД.

Сутність вчитель, реалізація встановлення користувача вчителем реалізована через БД, такий функціонал на порталі не доступний. Що дає змогу ще раз акцентуватись на безпеці порталу і надійності в не розповсюджені інформації. Загалом у вчителя реалізовано зв'язок один до багатьох.

Бази даних можуть містити таблиці, які пов'язані між собою різними зв'язками. Зв'язок (relationship) представляє асоціацію між сутностями різних типів.

При виділенні з ним виділяють головну або батьківську таблицю (primary key table / master table) і залежну, дочірню таблицю (foreign key table / child table). Дочірня таблиця залежить від батьківської.

Кафедра_КбПЗ_2021рік

Для організації зв'язку використовуються зовнішні ключі. Зовнішній ключ являє один або декілька стовпців з однієї таблиці, який одночасно є потенційним ключем з іншої таблиці. Зовнішній ключ необов'язково повинен відповідати первинному ключу з головної таблиці. Хоча, як правило, зовнішній ключ з залежною таблиці вказує на первинний ключ з головної таблиці.

Зв'язки між таблицями бувають наступних типів:

- Один до одного (One to one);
- Один до багатьох (One to many);
- Багато до багатьох (Many to many).

Зв'язок один до одного

Даний тип зв'язків зустрічає не часто. В цьому випадку об'єкту однієї сутності можна порівняти тільки один об'єкт іншої сутності. Наприклад, на деяких сайтах користувач може мати тільки один блог. Тобто виникає ставлення один користувач - один блог.

Нерідко цей тип зв'язків передбачає розбиття однієї великої таблиці на кілька маленьких. Основна батьківська таблиця в цьому випадку продовжує утримувати часто використовувані дані, а дочірня залежна таблиця зазвичай зберігає дані, які використовуються рідше.

В цьому відношенні первинний ключ залежною таблиці в той же час є зовнішнім ключем, який посилається на первинний ключ з головної таблиці.

Наприклад, таблиця Users являє користувачів і має такі стовпці:

- UserId (ідентифікатор, первинний ключ);
- Name (ім'я користувача).

І таблиця Blogs представляє блоги користувачів і має такі стовпці:

- BlogId (ідентифікатор, первинний і зовнішній ключ);
- Name (назва блогу).

Кафедра_КбПЗ_2021рік

В цьому випадку стовпець BlogId буде зберігати значення з шпальти UserId з таблиці користувачів. Тобто стовпець BlogId буде виступати одночасно первинним і зовнішнім ключем.

Зв'язок один до багатьох

Це найбільш часто зустрічається тип зв'язків. У цьому типі зв'язків кілька рядків з дочірній таблиці залежать від одного рядка в батьківській таблиці. Наприклад, в одному блозі може бути кілька статей. В цьому випадку таблиця блогів є батьківської, а таблиця статей - дочірньої. Тобто один блог - багато статей. Або інший приклад, у футбольній команді може грати кілька футболістів. І в той же час один футболіст одночасно може грати тільки в одній команді. Тобто одна команда - багато футболістів.

Наприклад, є таблиця Articles, яка представляє статті блогу і яка має такі стовпці:

- ArticleId (ідентифікатор, первинний ключ);
- BlogId (зовнішній ключ);
- Title (назва статті);
- Text (текст статті).

В цьому випадку стовпець BlogId з таблиці статей буде зберігати значення з шпальти BlogId з таблиці блогів.

Зв'язок багато до багатьох

При цьому типі зв'язків один рядок з таблиці А може бути пов'язана з безліччю рядків з таблиці В. В свою чергу один рядок з таблиці В може бути пов'язана з безліччю рядків з таблиці А. Типовий приклад - студенти і курси: один студент може відвідувати кілька курсів , і відповідно на один курс можуть записатися кілька студентів.

Інший приклад - статті і теги: для однієї статті можна визначити кілька тегів, а один тег може бути визначений для декількох статей.

Але в SQL Server на рівні бази даних ми не можемо встановити пр ямий зв'язок багато до багатьох між двома таблицями. Це робиться за допомогою

Кафедра_КбПЗ_2021рік

допоміжної проміжної таблиці. Іноді дані з цієї проміжної таблиці представляють окрему сутність.

Наприклад, у випадку з статтями і тегами є таблиця Tags, яка має два стовпці:

- TagId (ідентифікатор, первинний ключ);
- Text (текст тега).

Також нехай буде проміжна таблиця ArticleTags з наступними полями:

- TagId (ідентифікатор, первинний і зовнішній ключ);
- ArticleIdId (ідентифікатор, первинний і зовнішній ключ).

Зв'язок багато до багатьох в SQL

Технічно ми отримаємо дві зв'язки один-ко-багатьох. Стовпець TagId з таблиці ArticleTags буде посилатися на стовпець TagId з таблиці Tags. А стовпець ArticleId з таблиці ArticleTags буде посилатися на стовпець ArticleId з таблиці Articles. Тобто стовпці TagId і ArticleId в таблиці ArticleTags представляють складовою первинний ключ і одночасно є зовнішніми ключами для зв'язку з таблицями Articles і Tags.

Посилальна цілісність даних

При зміні первинних і зовнішніх ключів слід дотримуватися такої аспект як довідкова цілісність даних (referential integrity). Її основна ідея полягає в тому, щоб дві таблиці в базі даних, які зберігають одні й ті ж дані, підтримували їх узгодженість. Цілісність даних являє правильно вибудувані відносини між таблицями з коректною інсталяцією посилань між ними. В яких випадках цілісність даних може порушуватися:

- Аномалія видалення (deletion anomaly). Виникає при видаленні рядка з головної таблиці. В цьому випадку зовнішній ключ з залежною таблиці продовжує посилатися на віддалену рядок з головної таблиці;
- Аномалія вставки (insertion anomaly). Виникає при вставці рядка в залежну таблицю. В цьому випадку зовнішній ключ з залежною таблиці не відповідає первинному ключу жодної з рядків з головної таблиці;

Кафедра_КПЗ_2021рік

- Аномалії оновлення (update anomaly). При подібній аномалії кілька рядків однієї таблиці можуть містити дані, які належать одному і тому ж об'єкту. При зміні даних в одному рядку вони можу прийти в протиріччя з даними з іншого рядка.

Для вирішення аномалії видалення для зовнішнього ключа слід встановлювати одне з двох обмежень.

Якщо рядок з залежною таблиці обов'язково вимагає наявності рядки з головної таблиці, то для зовнішнього ключа встановлюється каскадне видалення. Тобто при видаленні рядка з головної таблиці відбувається видалення пов'язаної рядки (рядків) з залежною таблиці.

Якщо рядок з залежною таблиці допускає відсутність зв'язку з рядком з головної таблиці (тобто такий зв'язок не обов'язкова), то для зовнішнього ключа при видаленні пов'язаної рядки з головної таблиці задається установка значення NULL. При цьому стовпець зовнішнього ключа повинен допускати значення NULL.

Для вирішення аномалії вставки при додаванні в залежну таблицю даних стовпець, який представляє зовнішній ключ, повинен допускати значення NULL. І таким чином, якщо додається об'єкт не має зв'язку з головною таблицею, то в стовпці зовнішнього ключа буде стояти значення NULL.

Для вирішення проблеми аномалії оновлення застосовується нормалізація.

Кафедра_КбПЗ_2021рік

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо алгоритм роботи основної програми. Його блок-схема зображена на рисунку 4.1. Робота підпрограми зображено на рисунку 4.2. З рисунку 4.1 видно, що після запуску програми відбуваються наступні дії:

Багато веб-додатки надають своїм користувачам можливість аутентифицироваться в додатку і «увійти в систему». Реалізація цієї функції в веб-додатках може бути складною і потенційно ризикованою завданням. З цієї причини Laravel прагне надати вам інструменти, необхідні для швидкої, безпечної і простої реалізації аутентифікації.

За своєю суттю засоби аутентифікації Laravel складаються з «охоронців» і «провайдерів». Охоронці визначають, як користувачі проходять аутентифікацію для кожного запиту. Наприклад, Laravel поставляється з session захистом, який підтримує стан за допомогою сховища сеансів і файлів cookie.

Провайдери визначають, як користувачі витягуються з постійного сховища. Laravel поставляється з підтримкою отримання користувачів за допомогою Eloquent і будівника запитів до бази даних. Однак ви можете визначати додаткових постачальників, якщо це необхідно для застосування.

Файл конфігурації аутентифікації програми знаходиться за адресою config / auth.php. Цей файл містить декілька добре задокументованих опцій для настройки поведінки служб аутентифікації Laravel.

Рекомендації по базі даних

Кафедра_КОПЗ_2021рік

За замовчуванням Laravel включає в каталог App \ Models \ User модель Eloquentapp / Models. Ця модель може використовуватися з драйвером аутентифікації Eloquent за замовчуванням. Якщо додаток не використовує Eloquent, ви можете використовувати databaseпоставщіка аутентифікації, який використовує будівник запитів Laravel.

При побудові схеми бази даних для App \ Models \ Usermоделі переконайтеся, що довжина стовпчика пароля становить не менше 60 символів. Звичайно, usersміграція таблиць, включена в нові додатки Laravel, вже створює стовпець, довжина якого перевищує цю довжину.

Крім того, ви повинні переконатися, що users (або еквівалентна) таблиця містить remember_tokenстолбец рядки з 100 символів, що допускає значення NULL. Цей стовпець буде використовуватися для зберігання токена для користувачів, які вибирають опцію «запам'ятати мене» при вході в програму. Знову ж, usersміграція таблиць за замовчуванням, включена в нові додатки Laravel, вже містить цей стовпець.

Laravel пропонує кілька пакетів, пов'язаних з аутентифікацією. Перш ніж продовжити, ми розглянемо загальну екосистему аутентифікації в Laravel і обговоримо призначення кожного пакету.

По-перше, розглянемо, як працює аутентифікація. При використанні веб-браузера користувач вводить своє ім'я користувача та пароль через форму входу. Якщо ці облікові дані вірні, додаток збереже інформацію про аутентифицироваться користувача в призначеному для користувача сеансі. Файл cookie, відправлений браузеру, містить ідентифікатор сеансу, щоб наступні запити до додатка могли зв'язати користувача з правильним сеансом. Після отримання файлу cookie сеансу додаток отримає дані сеансу на основі ідентифікатора сеансу, зверніть увагу, що інформація аутентифікації була збережена в сеансі, і буде вважати користувача «аутентифицироваться».

Кафедра_КОПЗ_2021рік

Коли віддаленої службі необхідно пройти перевірку справжності для доступу до API, файли cookie зазвичай не використовуються для перевірки справжності через відсутність веб-браузера. Замість цього віддалена служба відправляє API-токен API при кожному запиті. Додаток може перевірити вхідний токен по таблиці допустимих токенов API і «аутентифицировать» запит як виконується користувачем, пов'язаним з цим токеном API.

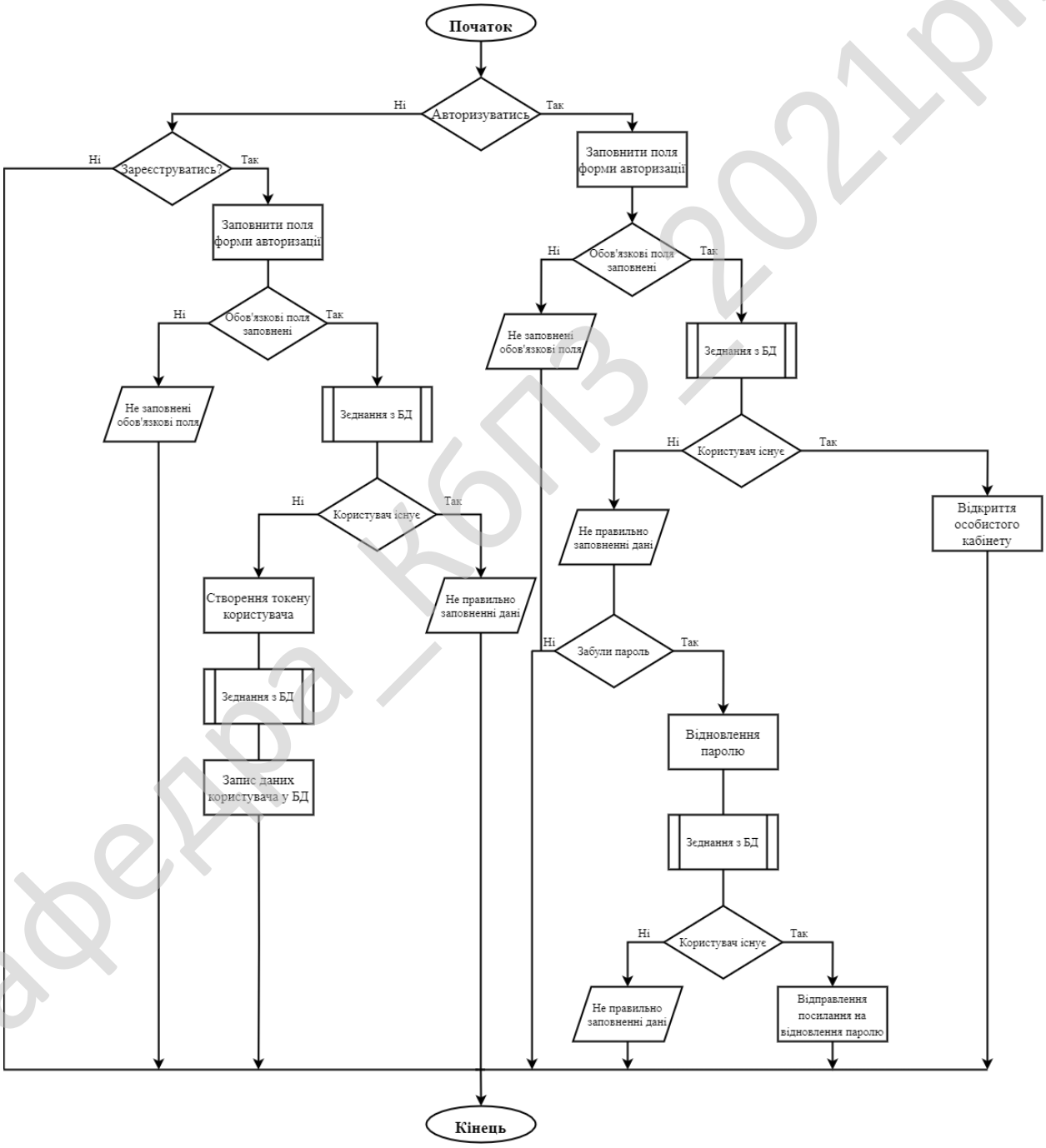


Рисунок 4.1 – Блок-схема роботи підпрограми авторизації

Кафедра_КБПЗ_2021рік

Для роботи з веб застосунком було створено базу даних для реєстрації користувачів. Також було створено веб сторінки для перегляду інформації, обробки даних, та надсилання інформації.

Так ПЗ має сторінки для взаємодії з загальною аудиторією, це головна, контакти, про нас а також сторінка авторизації. Користувач що потрапив на портал може відправити контактну форму зі сторінки Контакти, або ж прочитати основну інформацію про портал. Також якщо користувач має намір взаємодіяти більш поглиблено з порталом, він може пройти реєстрацію. Ознайомитись із роботою порталу можливо більш детально на Рисунку 4.2.

Так сторінка реєстрації створює в БД нового користувача, записує роль яку він обрав під час створення. Після авторизації користувач має змогу взаємодіяти з порталом у відповідності обраної ролі.

Кафедра_КбПЗ_2021рік

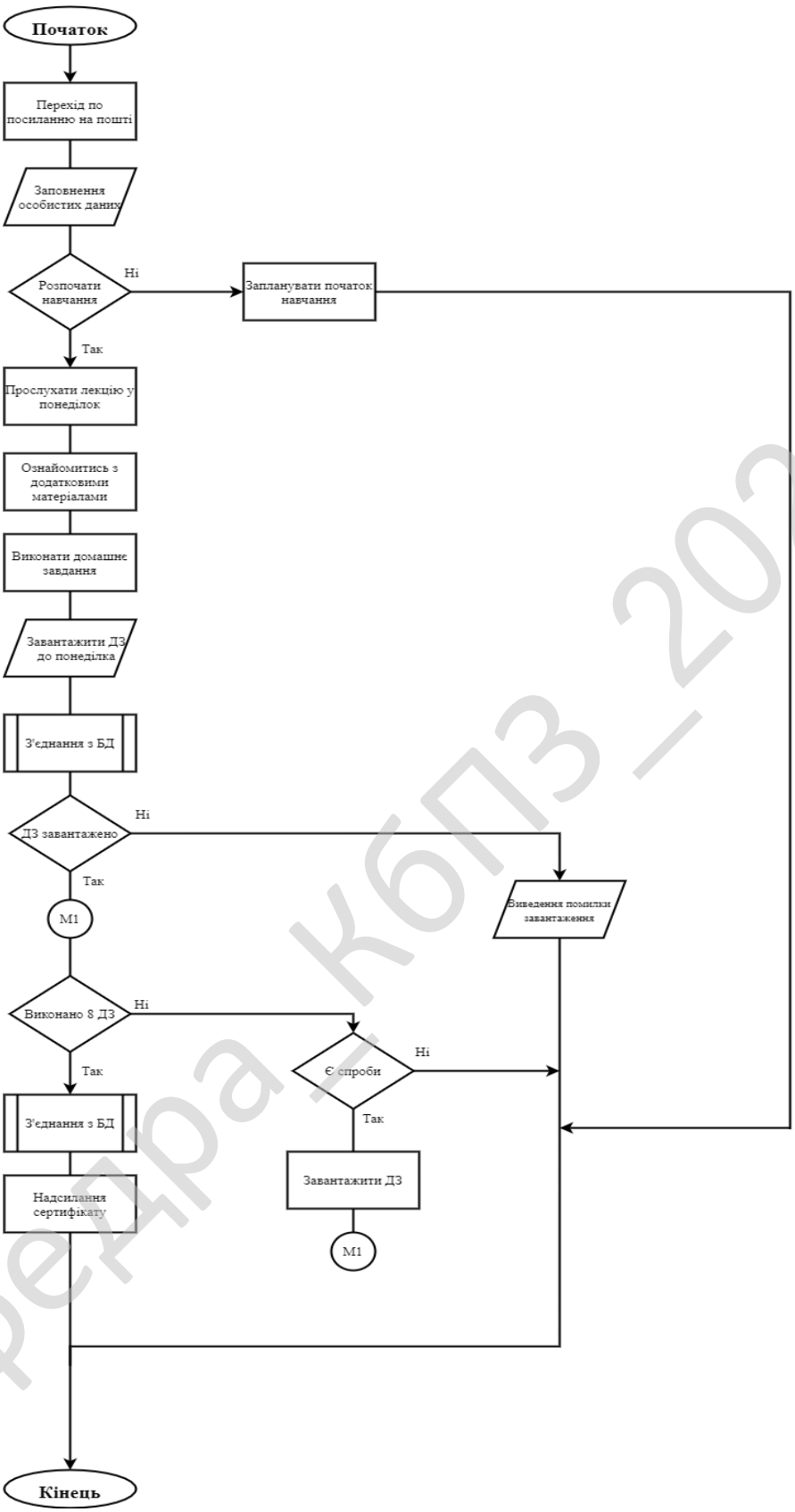


Рисунок 4.2 – Блок схема роботи порталу для учнів

Кафедра_КбПЗ_2021рік

4.2 Захист розробленого програмного забезпечення

Однією з найважливіших завдань в інформаційній безпеці є захист конфіденційних даних, яка є комплексною і досить складною. Навіть найдосвідченіші професіонали підходять до цієї справи з неабиякою часткою хвилювання. Організація ефективного захисту конфіденційних даних в рамках програм, баз даних, пристроїв зберігання і груп серверів сама по собі є непростим і ще більше ускладнюється при використанні різнорідних систем і суперечливих вимогах бізнесу.

Як правило, у випадку з конфіденційними даними ми розглядаємо різноманітні рішення з використанням шифрування, але в останні роки помітне зростання інтересу до іншої технології - токенізації.

Належним чином реалізоване шифрування є одним з найбільш ефективних засобів забезпечення безпеки конфіденційних даних. Воно дозволяє забезпечити доступ до даних тільки авторизованим користувачам і захищає дані, як при зберіганні, так і при передачі. Але шифрування не єдиний варіант захисту - є і альтернативні методи. Причому іноді найправильнішим рішенням буде не спроба захистити конфіденційні дані шифруванням, а взагалі відмовитися від їх передачі.

Токенізація якраз є технологією, яка надає цю можливість - її принцип полягає в підміні реальних конфіденційних даних якимись значеннями - токенами. Токенізація в чомусь схожа з шифруванням - обидві технології займаються маскированием реальних даних, але підхід до цього процесу в корені відрізняється. У разі шифрування, процес приховування даних звернемо при наявності правильного ключа. Будь-яка людина, отримавши ключ шифрування, зможе відновити вихідні дані.

Кафедра_КОПЗ_2021рік

У випадку з токенізацією, процес не є оборотним, адже замість даних при токенізації використовується токен, що не несе в собі ніякої конфіденційної інформації, і лише логічно пов'язаний з реальними даними, які зберігаються в добре захищеній базі даних. При цьому токен може мати той же формат (розмір і структура), що і оригінальні дані, що дозволяє мінімізувати необхідність внесення змін в працюючі з ним програми. Але при цьому викрадати токен безглуздо, так як він не допоможе отримати ніякі реальні конфіденційні дані.

Головна перевага токенізації в тому, що токени можуть повноцінно використовуватися всередині середовища їх застосування, але абсолютно даремні поза нею. Токенізація ідеально підходить для захисту конфіденційних даних, таких як номери кредитних карт, номери соціального страхування, а також будь-яких інших даних, які зловмисники регулярно викрадають з метою використання або продажу на "чорному ринку". Якщо зловмисникам не вдасться зламати сам сервер токенізації, щоб отримати пов'язані з токенами реальні дані, то викрадені токени не дадуть їм зовсім ніяких можливостей їх використовувати.

Підвищення інтересу до токенізації пов'язано, перш за все, з тим, що вона дозволяє забезпечити захист даних при потенційно низької вартості. Додавання шифрування в системи - особливо в уже діючі - значно збільшує навантаження на них. Внесення змін до додатка для впровадження шифрування може значно підвищити витрати, знизити продуктивність і збільшити вимоги до програмістам і адміністраторам систем. Крім того, в разі використання різнорідних систем, виникає необхідність шифруванні, розшифрування і повторному шифруванні даних в різних місцях, що створює додаткові уразливості, які можуть бути використані зловмисниками. Причому, чим більше ключів використовується в системі, тим більше можливостей для її атаки. Робота з ключами особливо небезпечна, з огляду на поширеність шкідливого ПО для перехоплення даних безпосередньо в

Кафедра_КОПЗ_2021рік

оперативній пам'яті, яке дозволяє отримати доступ до ключів навіть не маючи адміністративних привілеїв на зараженій машині.

Крім мінімізації вимог до внесення змін до Програми, які використовуються, токенізація знижує ризики для конфіденційних даних. При правильній реалізації, додатки зможуть використовувати маркери у всій системі і звертатися до захищених конфіденційних даних тільки в разі крайньої необхідності. Додатки можуть зберігати, використовувати і здійснювати транзакції, оперуючи тільки токеном і не піддаючи реальні дані ризику. Хоча деякі операції, звичайно, вимагають звернення до реальних даних, токенізація дозволяє мінімізувати їх використання.

Наприклад, одним з найпоширеніших прикладів використання токенізації є її застосування для платежі з використанням платіжних карт. Більш детально цей сценарій застосування буде розглянуто пізніше, але використання токена замість значення номера платіжної картки дозволяє провести відстеження транзакції і її виконання без ризику компрометації реальних даних карти. Доступ до реального номеру знадобиться тільки в процесинговому центрі. Ну а в тому випадку, якщо процесинговий центр також використовує токенізацію, то можливе проведення транзакції взагалі без використання реальнОднією з найважливіших завдань в інформаційній безпеці є захист конфіденційних даних, яка є комплексною і досить складною. Навіть найдосвідченіші професіонали підходять до цієї справи з неабиякою часткою хвилювання. Організація ефективного захисту конфіденційних даних в рамках програм, баз даних, пристроїв зберігання і груп серверів сама по собі є непростим і ще більше ускладнюється при використанні різнорідних систем і суперечливих вимогах бізнесу.

Як правило, у випадку з конфіденційними даними ми розглядаємо різноманітні рішення з використанням шифрування, але в останні роки помітне зростання інтересу до іншої технології - токенізації.

Кафедра_КбПЗ_2021рік

Належним чином реалізоване шифрування є одним з найбільш ефективних засобів забезпечення безпеки конфіденційних даних. Воно дозволяє забезпечити доступ до даних тільки авторизованим користувачам і захищає дані, як при зберіганні, так і при передачі. Але шифрування не єдиний варіант захисту - є і альтернативні методи. Причому іноді найправильнішим рішенням буде не спроба захистити конфіденційні дані шифруванням, а взагалі відмовитися від їх передачі.

Токенізація якраз є технологією, яка надає цю можливість - її принцип полягає в підміні реальних конфіденційних даних якимись значеннями - токенами. Токенізація в чомусь схожа з шифруванням - обидві технології займаються маскированием реальних даних, але підхід до цього процесу в корені відрізняється. У разі шифрування, процес приховування даних звернемо при наявності правильного ключа. Будь-яка людина, отримавши ключ шифрування, зможе відновити вихідні дані.

У випадку з токенізацією, процес не є оборотним, адже замість даних при токенізації використовується токен, що не несе в собі ніякої конфіденційної інформації, і лише логічно пов'язаний з реальними даними, які зберігаються в добре захищеній базі даних. При цьому токен може мати той же формат (розмір і структура), що і оригінальні дані, що дозволяє мінімізувати необхідність внесення змін в працюючі з ним програми. Але при цьому викрадати токен безглуздо, так як він не допоможе отримати ніякі реальні конфіденційні дані.

Головна перевага токенізації в тому, що токени можуть повноцінно використовуватися всередині середовища їх застосування, але абсолютно даремні поза нею. Токенізація ідеально підходить для захисту конфіденційних даних, таких як номери кредитних карт, номери соціального страхування, а також будь-яких інших даних, які зловмисники регулярно викрадають з метою використання або продажу на "чорному ринку". Якщо зловмисникам не вдасться зламати сам сервер токенізації, щоб отримати

Кафедра_КбПЗ_2021рік

пов'язані з токенами реальні дані, то викрадені токени не дадуть їм зовсім ніяких можливостей їх використовувати.

Підвищення інтересу до токенізації пов'язано, перш за все, з тим, що вона дозволяє забезпечити захист даних при потенційно низької вартості. Додавання шифрування в системи - особливо в уже діючі - значно збільшує навантаження на них. Внесення змін до додатка для впровадження шифрування може значно підвищити витрати, знизити продуктивність і збільшити вимоги до програмістам і адміністраторам систем. Крім того, в разі використання різнорідних систем, виникає необхідність шифруванні, розшифрування і повторному шифруванні даних в різних місцях, що створює додаткові уразливості, які можуть бути використані зловмисниками. Причому, чим більше ключів використовується в системі, тим більше можливостей для її атаки. Робота з ключами особливо небезпечна, з огляду на поширеність шкідливого ПО для перехоплення даних безпосередньо в оперативній пам'яті, яке дозволяє отримати доступ до ключів навіть не маючи адміністративних привілеїв на зараженій машині.

Крім мінімізації вимог до внесення змін до Програми, які використовуються, токенізація знижує ризики для конфіденційних даних. При правильній реалізації, додатки зможуть використовувати маркери у всій системі і звертатися до захищених конфіденційних даних тільки в разі крайньої необхідності. Додатки можуть зберігати, використовувати і здійснювати транзакції, оперуючи тільки токеном і не піддаючи реальні дані ризику. Хоча деякі операції, звичайно, вимагають звернення до реальних даних, токенізація дозволяє мінімізувати їх використання.

Наприклад, одним з найпоширеніших прикладів використання токенізації є її застосування для платежі з використанням платіжних карт. Більш детально цей сценарій застосування буде розглянуто пізніше, але використання токена замість значення номера платіжної картки дозволяє

Кафедра_КОПЗ_2021рік

провести відстеження транзакції і її виконання без ризику компрометації реальних даних карти. Доступ до реального номеру знадобиться тільки в процесинговому центрі. Ну а в тому випадку, якщо процесинговий центр також використовує токенізацію, то можливе проведення транзакції взагалі без використання реальних даних.

Втім, це не означає, що токенізація завжди буде кращим вибором, ніж шифрування. Два цих рішення тісно пов'язані між собою і фокус в тому, щоб визначити, яке з рішень буде краще в даних конкретних обставинах. Наприклад, кожна система токенізації покладається на шифрування для захисту зберігаються конфіденційних даних. Але іноді, кращий спосіб забезпечення безпеки конфіденційних даних - це відмова від їх зберігання в більшій частині місць системи. Токенізація дозволяє домогтися цього, зберігши при цьому можливість працювати з конфіденційними даними, використовуючи маркери.

У цьому матеріалі ми максимально заглибимося в токенізацію і постараємося зрозуміти, як працює технологія, досліджувати різні варіанти використання і сценарії розгортання, а також звернемо увагу на критерії вибору, які дозволять підібрати правильне рішення. Крім того ми розглянемо використання сторонніх сервісів, що дозволяють виконати вимоги PCI, і можливість розробки власних рішень для внутрішніх додатків. Перш ніж ми заглибимося в специфіку, важливо уточнити значення деяких термінів, які ми будемо використовувати в даному матеріалі.

Токен. Випадкові дані, які використовуються в якості якогось сурогату для інших даних. У токені немає математичних залежностей між випадковими і оригінальними даними, тому оригінальні дані не можуть бути визначені за значенням токена. Асоціація між оригінальними і випадковими даними ведеться тільки в базі даних і зіставлення оригінальних даних і сурогату може бути проведено тільки в ній.

Кафедра_КОПЗ_2021рік

Шифрування. Оборотне перетворення даних з метою зробити їх нечитабельним для користувачів, у яких відсутній ключ, який дозволяє провести зворотне перетворення і повернути даними вихідний вид.

Хеш. Необоротне перетворення за певним алгоритмом масиву даних в бітову рядок фіксованої довжини з невипадковим значенням.

Зашифрований токен. Дані, які були зашифровані, використовуючи спеціальні методи шифрування зі збереженням формату і типу даних, а не замінені випадковим значенням. Зазвичай такі зашифровані дані використовуються у вигляді токенов, але можуть бути розшифровані з метою отримати оригінальні дані. Технічно це рішення відноситься до шифрування, а не токенізації.

Кафедра_КбПЗ_2021рік

5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програмне забезпечення зручне та ефективне в користуванні, має простий інтуїтивно зрозумілий інтерфейс, не потребує інсталювання в систему і внесення змін в реєстр. Керування програмою відбувається через головний інтерфейсний модуль.

5.1 Рівні впровадження ПЗ

Впровадження програмного забезпечення, покликаного автоматизувати значну частину бізнесу і виробничих процесів, відбувається на трьох рівнях.

• Технічний рівень включає послуги з установки й тестування програм, налаштування. На цьому етапі виконується настройка функцій програмного продукту під завдання проекту, перевіряються можливості вирішення і справність роботи.

• Технологічний рівень - це інтеграція ПО в роботу підприємства, адаптація під інші програми. Автоматизація передбачає безперебійне взаємодія всіх процесів після установки програм.

• Організаційний рівень впровадження - це навчання персоналу роботі з новим програмним забезпеченням.

Після впровадження програми і виконання технічного завдання, проект по впровадженню ПО переходить в стадію технічної підтримки. Вона може тривати роками - в залежності від побажань замовника. Технічна підтримка включає виправлення помилок в роботі програми, адаптацію під нову техніку або інтеграцію з новими програмами, розширення можливостей і інші завдання.

Кафедра_КОПЗ_2021рік

5.2 Тестування

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

• відповідності пропонованим вимогам;

• правильну відповідь для всіх можливих вхідних даних;

• виконання функцій за прийнятний час;

• практичність;

• сумісність з ОС і стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченно, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу і ресурсів.

Проводилось тестування форматом білого ящика засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреної, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

В цьому випадку формуються тестові варіанти, в яких:

• гарантується перевірка всіх незалежних маршрутів програми;

• знаходяться гілки True, False для всіх логічних рішень;

• виконуються всі цикли (в межах їх кордонів і діапазонів);

• аналізується правильність внутрішніх структур даних.

Недоліки тестування "білого ящика":

• кількість незалежних маршрутів може бути дуже велика.

Кафедра_КбПЗ_2021рік

• повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

• у програмі можуть бути пропущені деякі маршрути.

• не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білого ящика" пов'язані з тим, що принцип «білого ящика» дозволяє врахувати особливості програмних помилок:

• кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

• попередні припущення про ймовірності потоку управління або даних в програмі часто бувають некоректними. В результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

• при записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних і семантичних).

• Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

5.3 Умови розповсюдження ПЗ

Під умовно-безкоштовним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційної ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому користувачеві пропонується обмежена за можливостями (НЕ повнофункціональна або демонстраційна версія), строками дії використання (тріал версія або версія з вбудованим набридливим нагадуванням про необхідність оплати) програми.

В угоді про використання (ліцензії кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або відповідного напряму підготовки (НЕ тестове) її використання.

Кафедра_КбПЗ_2021рік

Основний принцип умовно-безкоштовного програмного забезпечення - «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безкоштовний, надається користувачам безкоштовно. Звичайно

користувач платить тільки за час завантаження файлів через Інтернет або на носії (CD диск, флешку, ключ). Протягом певного терміну, який становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму.

Кафедра_КбПЗ_2021рік

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання кваліфікаційної бакалаврської роботи, призначено для системи дистанційного навчання контролю якості програмних застосунків..

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

Система що розробляється в дипломному проекті має простий, дружній і зручний інтерфейс, що забезпечує легкість в освоєнні роботи з програмним продуктом, зручність у використанні і не вимагає особливих спеціальних знань.

При створенні програмного забезпечення був використаний об'єктно-орієнтований підхід, який відповідає сучасним вимогам в області розробки комерційних програмних систем.

Система реалізована із застосуванням таких мов як PHP та JS, а також HTML, CSS. Це дозволило мінімізувати термін розробки програмного забезпечення, і, отже, зменшити витрати на його розробку. Запропонована система ділиться на Front-end та Back-end частини.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень і повністю відповідає вимогам технічного завдання. Створена система має потенційну можливість для подальшого вдосконалення і застосування в різних областях.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного

Кафедра_КоПЗ_2021рік

завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Кафедра_КбПЗ_2021рік

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Steven John Metsker Design Patterns in PHP Фортуна Эл - Москва, 2011. - 480 с.
2. Агуров, Павел PHP Сборник рецептов / Павел Агуров. - М.: "БХВ-Петербург", 2012. - 432 с.
3. Теория и практика обеспечения безопасного доступа к информационным ресурсам. - Москва: Наука, 2015. - 552 с.
4. Архитектура, программирование, веб приложения / В.Б. Бродин, М.И. Шагурин. - М.: ЭКОМ, 1999. - 400 с.
5. Дейтел Х.М., Дейтел П.Дж.. Как программировать на PHP М.: ЗАО БИНОМ, 1999, 1000 с.
6. Партыка, Т. Л. Информационная безопасность
7. Подбельский В.В., Фомин С.С. Программирование на языке PHP. М.: ФиС, 1999, 600 с.
8. A.B. Bagula, M. Botha, and A.E Krzesinski. Online Traffic Engineering: The Least Interference Optimization Algorithm // IEEE Communications Society – 2004, P. 1232-1236.
9. Anees Shaikh, Jennifer Rexford, and Kang G. Shin. Evaluating the Impact of Stale Link State on Quality-of-Service Routing // IEEE/ACM Transactions on Networking. – 2001. – №9(2), P. 162-176.
10. Basabi Chakraborty. Simultaneous Search for Multiple Routes using Genetic Algorithm / IEEE lntemational Conference on Computational oltelligencs for Measurement System and Applications Boston. MA, USA, 14-16, July 2004, P. 77-80/
11. Barakat, E. Altman, and W. Dabbous. On TCP performance in a heterogeneous network: a survey // IEEE Communications Magazine. – 2000. – №38(1). – P. 40 - 46.

Кафедра_КбПЗ_2021рік

Додаток А
(обов'язковий)

Технічне завдання

Зміст

Кафедра_КОПЗ_2021рік

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи дистанційного навчання контролю якості програмних застосунків.

2 Підстава для розробки

Підставою для розробки служить завдання на кваліфікаційну бакалаврську роботу, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 204-02 від 28.12.2020 року).

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення системи дистанційного навчання контролю якості програмних застосунків.

4 Джерела розробки

Джерелом цієї кваліфікаційної бакалаврської роботи є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

Кафедра_КБПЗ_2021рік

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– систему дистанційного навчання контролю якості програмних застосунків;

– цілісність даних у процесі роботи та при зберіганні;

– гнучкий дизайн;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

Кафедра_КбПЗ_2021рік

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows XP/Vista/7/8/10 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows XP/Vista/7/8/10.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Кафедра_КбПЗ_2021рік

PHP.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи. 2

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 68 аркушів.

Кафедра_КбПЗ_2021рік

8 Етапи розробки

8.1 Збір і обробка інформації по темі кваліфікаційної бакалаврської роботи. Постановка задачі на виконання кваліфікаційної бакалаврської роботи (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень кваліфікаційної бакалаврської роботи.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання кваліфікаційної бакалаврської роботи на попередній захист 22.05.2020 р.

11.2 Подання кваліфікаційної бакалаврської роботи на захист 3.06.2020 р.

Кафедра_КОПЗ_2021рік

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник кваліфікаційної бакалаврської роботи

_____ Бісюк В.А.

Програмне забезпечення системи дистанційного навчання контролю якості програмних застосунків

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 22

Літера: РП

Кропивницький – 2021 року

Кафедра_КбПЗ_2021рік

Env.php' – файл конфігурацій

```
APP_NAME=QA Mentoring
APP_ENV=local
APP_KEY=base64:Bcave/fIgoJRVouU9LpcanmQwJqHNrn99wMdxxEtvlk=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=localhost
DB_PORT=3307
DB_DATABASE=laravel
DB_USERNAME=root
DB_PASSWORD=root

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=mailhog
MAIL_PORT=1025
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS=null
MAIL_FROM_NAME="${APP_NAME}"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=

PUSHER_APP_ID=
PUSHER_APP_KEY=
```

Кафедра_КбПЗ_2021рік

```
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1

MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```

Кафедра_КбПЗ_2021рік

AuthenticatedSessionController.php' – файл авторизації

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Http\Requests\Auth\LoginRequest;
use App\Providers\RouteServiceProvider;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class AuthenticatedSessionController extends Controller
{
    /**
     * Display the login view.
     *
     * @return \Illuminate\View\View
     */
    public function create()
    {
        return view('auth.login');
    }

    /**
     * Handle an incoming authentication request.
     *
     * @param  \App\Http\Requests\Auth\LoginRequest  $request
     * @return \Illuminate\Http\RedirectResponse
     */
    public function store(LoginRequest $request)
    {
        $request->authenticate();

        $request->session()->regenerate();

        return redirect()->intended(RouteServiceProvider::HOME);
    }

    /**
     * Destroy an authenticated session.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\RedirectResponse
     */
    public function destroy(Request $request)
```

Кафедра_КбПЗ_2021рік

```
    {
        Auth::guard('web')->logout();

        $request->session()->invalidate();

        $request->session()->regenerateToken();

        return redirect('/');
    }
}
```

Кафедра_КбПЗ_2021рік

ConfirmablePasswordController.php' – файл підтвердження паролю

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Providers\RouteServiceProvider;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Validation\ValidationException;

class ConfirmablePasswordController extends Controller
{
    /**
     * Show the confirm password view.
     *
     * @return \Illuminate\View\View
     */
    public function show()
    {
        return view('auth.confirm-password');
    }

    /**
     * Confirm the user's password.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return mixed
     */
    public function store(Request $request)
    {
        if (! Auth::guard('web')->validate([
            'email' => $request->user()->email,
            'password' => $request->password,
        ])) {
            throw ValidationException::withMessages([
                'password' => __('auth.password'),
            ]);
        }

        $request->session()->put('auth.password_confirmed_at', time());

        return redirect()->intended(RouteServiceProvider::HOME);
    }
}
```

Кафедра_КбПЗ_2021рік

EmailVerificationNotificationController.php' – файл конфігурацій

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Providers\RouteServiceProvider;
use Illuminate\Http\Request;

class EmailVerificationNotificationController extends Controller
{
    /**
     * Send a new email verification notification.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\RedirectResponse
     */
    public function store(Request $request)
    {
        if ($request->user()->hasVerifiedEmail()) {
            return redirect()->intended(RouteServiceProvider::HOME);
        }

        $request->user()->sendEmailVerificationNotification();

        return back()->with('status', 'verification-link-sent');
    }
}
```

Кафедра_КбПЗ_2021рік

‘EmailVerificationPromptController.php’ – файл підтвердження емейлу

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Providers\RouteServiceProvider;
use Illuminate\Http\Request;

class EmailVerificationPromptController extends Controller
{
    /**
     * Display the email verification prompt.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return mixed
     */
    public function __invoke(Request $request)
    {
        return $request->user()->hasVerifiedEmail()
                    ? redirect()->intended(RouteServiceProvider::HOME)
                    : view('auth.verify-email');
    }
}
```

Кафедра_КбПЗ_2021рік

‘login.blade.php’ – файл сторінки логіну

```
<x-guest-layout>
    <x-auth-card>
        <x-slot name="logo">
            <a href="/">
                <x-application-logo class="w-20 h-20 fill-current text-gray-500"
/>
            </a>
        </x-slot>

        <!-- Session Status -->
        <x-auth-session-status class="mb-4" :status="session('status')" />

        <!-- Validation Errors -->
        <x-auth-validation-errors class="mb-4" :errors="$errors" />

        <form method="POST" action="{{ route('login') }}">
            @csrf

            <!-- Email Address -->
            <div>
                <x-label for="email" :value="__('Email')" />

                <x-input id="email" class="block mt-1 w-full" type="email"
name="email" :value="old('email')" required autofocus />
            </div>

            <!-- Password -->
            <div class="mt-4">
                <x-label for="password" :value="__('Password')" />

                <x-input id="password" class="block mt-1 w-full"
                                type="password"
                                name="password"
                                required autocomplete="current-password" />
            </div>

            <!-- Remember Me -->
            <div class="block mt-4">
                <label for="remember_me" class="inline-flex items-center">
                    <input id="remember_me" type="checkbox" class="rounded
border-gray-300 text-indigo-600 shadow-sm focus:border-indigo-300 focus:ring
focus:ring-indigo-200 focus:ring-opacity-50" name="remember">
                    <span class="ml-2 text-sm text-gray-600">{{ __('Remember
me') }}</span>
```

Кафедра_КбПЗ_2021рік

```
                </label>
            </div>

            <div class="flex items-center justify-end mt-4">
                @if (Route::has('password.request'))
                    <a class="underline text-sm text-gray-600 hover:text-gray-
900" href="{{ route('password.request') }}">
                        {{ __('Forgot your password?') }}
                    </a>
                @endif

                <x-button class="ml-3">
                    {{ __('Log in') }}
                </x-button>
            </div>
        </form>
    </x-auth-card>
</x-guest-layout>
```

Кафедра_КбПЗ_2021рік

‘dashboard.blade.php’ – файл сторінки навчання

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">

        <title>Laravel</title>

        <!-- Fonts -->
        <link
href="https://fonts.googleapis.com/css2?family=Nunito:wght@400;600;700&display=s
wap" rel="stylesheet">

        <!-- Styles -->
        <style>
            /*! normalize.css v8.0.1 | MIT License |
github.com/necolas/normalize.css */html{line-height:1.15;-webkit-text-size-
adjust:100%}body{margin:0}a{background-
color:transparent}[hidden]{display:none}html{font-family:system-ui,-apple-
system,BlinkMacSystemFont,Segoe UI,Roboto,Helvetica Neue,Arial,Noto Sans,sans-
serif,Apple Color Emoji,Segoe UI Emoji,Segoe UI Symbol,Noto Color Emoji;line-
height:1.5}*,:after,:before{box-sizing:border-box;border:0 solid
#e2e8f0}a{color:inherit;text-
decoration:inherit}svg,video{display:block;vertical-align:middle}video{max-
width:100%;height:auto}.bg-white{--bg-opacity:1;background-
color:#fff;background-color:rgba(255,255,255,var(--bg-opacity))}.bg-gray-100{--
bg-opacity:1;background-color:#f7fafc;background-color:rgba(247,250,252,var(--
bg-opacity))}.border-gray-200{--border-opacity:1;border-color:#edf2f7;border-
color:rgba(237,242,247,var(--border-opacity))}.border-t{border-top-
width:1px}.flex{display:flex}.grid{display:grid}.hidden{display:none}.items-
center{align-items:center}.justify-center{justify-content:center}.font-
semibold{font-weight:600}.h-5{height:1.25rem}.h-8{height:2rem}.h-
16{height:4rem}.text-sm{font-size:.875rem}.text-lg{font-size:1.125rem}.leading-
7{line-height:1.75rem}.mx-auto{margin-left:auto;margin-right:auto}.ml-1{margin-
left:.25rem}.mt-2{margin-top:.5rem}.mr-2{margin-right:.5rem}.ml-2{margin-
left:.5rem}.mt-4{margin-top:1rem}.ml-4{margin-left:1rem}.mt-8{margin-
top:2rem}.ml-12{margin-left:3rem}.-mt-px{margin-top:-1px}.max-w-6xl{max-
width:72rem}.min-h-screen{min-height:100vh}.overflow-hidden{overflow:hidden}.p-
6{padding:1.5rem}.py-4{padding-top:1rem;padding-bottom:1rem}.px-6{padding-
left:1.5rem;padding-right:1.5rem}.pt-8{padding-
top:2rem}.fixed{position:fixed}.relative{position:relative}.top-0{top:0}.right-
0{right:0}.shadow{box-shadow:0 1px 3px 0 rgba(0,0,0,.1),0 1px 2px 0
rgba(0,0,0,.06)}.text-center{text-align:center}.text-gray-200{--text-
opacity:1;color:#edf2f7;color:rgba(237,242,247,var(--text-opacity))}.text-gray-
```

Кафедра КБПЗ 2021 рік

```
300{--text-opacity:1;color:#e2e8f0;color:rgba(226,232,240,var(--text-
opacity))}.text-gray-400{--text-
opacity:1;color:#cbd5e0;color:rgba(203,213,224,var(--text-opacity))}.text-gray-
500{--text-opacity:1;color:#a0aec0;color:rgba(160,174,192,var(--text-
opacity))}.text-gray-600{--text-
opacity:1;color:#718096;color:rgba(113,128,150,var(--text-opacity))}.text-gray-
700{--text-opacity:1;color:#4a5568;color:rgba(74,85,104,var(--text-
opacity))}.text-gray-900{--text-
opacity:1;color:#1a202c;color:rgba(26,32,44,var(--text-
opacity))}.underline{text-decoration:underline}.antialiased{-webkit-font-
smoothing:antialiased;-moz-osx-font-smoothing:grayscale}.w-5{width:1.25rem}.w-
8{width:2rem}.w-auto{width:auto}.grid-cols-1{grid-template-
columns:repeat(1,minmax(0,1fr))}@media (min-width:640px){.sm\:rounded-lg{border-
radius:.5rem}.sm\:block{display:block}.sm\:items-center{align-
items:center}.sm\:justify-start{justify-content:flex-start}.sm\:justify-
between{justify-content:space-between}.sm\:h-20{height:5rem}.sm\:ml-0{margin-
left:0}.sm\:px-6{padding-left:1.5rem;padding-right:1.5rem}.sm\:pt-0{padding-
top:0}.sm\:text-left{text-align:left}.sm\:text-right{text-align:right}}@media
(min-width:768px){.md\:border-t-0{border-top-width:0}.md\:border-l{border-left-
width:1px}.md\:grid-cols-2{grid-template-columns:repeat(2,minmax(0,1fr))}}@media
(min-width:1024px){.lg\:px-8{padding-left:2rem;padding-right:2rem}}@media
(prefers-color-scheme:dark){.dark\:bg-gray-800{--bg-opacity:1;background-
color:#2d3748;background-color:rgba(45,55,72,var(--bg-opacity))}.dark\:bg-gray-
900{--bg-opacity:1;background-color:#1a202c;background-color:rgba(26,32,44,var(-
-bg-opacity))}.dark\:border-gray-700{--border-opacity:1;border-
color:#4a5568;border-color:rgba(74,85,104,var(--border-opacity))}.dark\:text-
white{--text-opacity:1;color:#fff;color:rgba(255,255,255,var(--text-
opacity))}.dark\:text-gray-400{--text-
opacity:1;color:#cbd5e0;color:rgba(203,213,224,var(--text-opacity))}}
        </style>

        <style>
            body {
                font-family: 'Nunito', sans-serif;
            }
        </style>
    </head>
    <body class="antialiased">
        <div class="relative flex items-top justify-center min-h-screen bg-gray-
100 dark:bg-gray-900 sm:items-center py-4 sm:pt-0">
            @if (Route::has('login'))
                <div class="hidden fixed top-0 right-0 px-6 py-4 sm:block">
                    @auth
                        <a href="{{ url('/dashboard') }}" class="text-sm text-
gray-700 underline">Dashboard</a>
                    @else
```

Кафедра КБТЗ 2021рік

```
                        <a href="{{ route('login') }}" class="text-sm text-gray-
700 underline">Log in</a>

                        @if (Route::has('register'))
                            <a href="{{ route('register') }}" class="ml-4 text-
sm text-gray-700 underline">Register</a>
                        @endif
                    @endauth
                </div>
            @endif

            <div class="max-w-6xl mx-auto sm:px-6 lg:px-8">
                <div class="flex justify-center pt-8 sm:justify-start sm:pt-0">
                    <svg viewBox="0 0 651 192" fill="none"
xmlns="http://www.w3.org/2000/svg" class="h-16 w-auto text-gray-700 sm:h-20">
                        <g clip-path="url(#clip0)" fill="#EF3B2D">
                            <path d="M248.032 44.676h-16.466v100.23h47.394v-
14.748h-30.928V44.676zM337.091 87.202c-2.101-3.341-5.083-5.965-8.949-7.875-
3.865-1.909-7.756-2.864-11.669-2.864-5.062 0-9.69.931-13.89 2.792-4.201 1.861-
7.804 4.417-10.811 7.661-3.007 3.246-5.347 6.993-7.016 11.239-1.672 4.249-2.506
8.713-2.506 13.389 0 4.774.834 9.26 2.506 13.459 1.669 4.202 4.009 7.925 7.016
11.169 3.007 3.246 6.609 5.799 10.811 7.66 4.199 1.861 8.828 2.792 13.89 2.792
3.913 0 7.804-.955 11.669-2.863 3.866-1.908 6.849-4.533 8.949-
7.875v9.021h15.607V78.182h-15.607v9.02zm-1.431 32.503c-.955 2.578-2.291 4.821-
4.009 6.73-1.719 1.91-3.795 3.437-6.229 4.582-2.435 1.146-5.133 1.718-8.091
1.718-2.96 0-5.633-.572-8.019-1.718-2.387-1.146-4.438-2.672-6.156-4.582-1.719-
1.909-3.032-4.152-3.938-6.73-.909-2.577-1.36-5.298-1.36-8.161 0-2.864.451-5.585
1.36-8.162.905-2.577 2.219-4.819 3.938-6.729 1.718-1.908 3.77-3.437 6.156-4.582
2.386-1.146 5.059-1.718 8.019-1.718 2.958 0 5.656.572 8.091 1.718 2.434 1.146
4.51 2.674 6.229 4.582 1.718 1.91 3.054 4.152 4.009 6.729.953 2.577 1.432 5.298
1.432 8.162-.001 2.863-.479 5.584-1.432 8.161zM463.954 87.202c-2.101-3.341-
5.083-5.965-8.949-7.875-3.865-1.909-7.756-2.864-11.669-2.864-5.062 0-9.69.931-
13.89 2.792-4.201 1.861-7.804 4.417-10.811 7.661-3.007 3.246-5.347 6.993-7.016
11.239-1.672 4.249-2.506 8.713-2.506 13.389 0 4.774.834 9.26 2.506 13.459 1.669
4.202 4.009 7.925 7.016 11.169 3.007 3.246 6.609 5.799 10.811 7.66 4.199 1.861
8.828 2.792 13.89 2.792 3.913 0 7.804-.955 11.669-2.863 3.866-1.908 6.849-4.533
8.949-7.875v9.021h15.607V78.182h-15.607v9.02zm-1.432 32.503c-.955 2.578-2.291
4.821-4.009 6.73-1.719 1.91-3.795 3.437-6.229 4.582-2.435 1.146-5.133 1.718-
8.091 1.718-2.96 0-5.633-.572-8.019-1.718-2.387-1.146-4.438-2.672-6.156-4.582-
1.719-1.909-3.032-4.152-3.938-6.73-.909-2.577-1.36-5.298-1.36-8.161 0-2.864.451-
5.585 1.36-8.162.905-2.577 2.219-4.819 3.938-6.729 1.718-1.908 3.77-3.437 6.156-
4.582 2.386-1.146 5.059-1.718 8.019-1.718 2.958 0 5.656.572 8.091 1.718 2.434
1.146 4.51 2.674 6.229 4.582 1.718 1.91 3.054 4.152 4.009 6.729.953 2.577 1.432
5.298 1.432 8.162 0 2.863-.479 5.584-1.432 8.161zM650.772 44.676h-
15.606v100.23h15.606V44.676zM365.013 144.906h15.607V93.538h26.776V78.182h-
42.383v66.724zM542.133 78.182l1-19.616 51.096-19.616-51.096h-15.808l25.617
```

Кафедра КБПЗ 2021рік

```
66.724h19.614l25.617-66.724h-15.808zM591.98 76.466c-19.112 0-34.239 15.706-
34.239 35.079 0 21.416 14.641 35.079 36.239 35.079 12.088 0 19.806-4.622 29.234-
14.688l-10.544-8.158c-.006.008-7.958 10.449-19.832 10.449-13.802 0-19.612-
11.127-19.612-16.884h51.777c2.72-22.043-11.772-40.877-33.023-40.877zm-18.713
29.28c.12-1.284 1.917-16.884 18.589-16.884 16.671 0 18.697 15.598 18.813
16.884h-37.402zM184.068 43.892c-.024-.088-.073-.165-.104-.25-.058-.157-.108-
.316-.191-.46-.056-.097-.137-.176-.203-.265-.087-.117-.161-.242-.265-.345-.085-
.086-.194-.148-.29-.223-.109-.085-.206-.182-.327-.252l-.002-.001-.002-.002-
35.648-20.524a2.971 2.971 0 00-2.964 0l-35.647 20.522-.002.002-.002.001c-
.121.07-.219.167-.327.252-.096.075-.205.138-.29.223-.103.103-.178.228-.265.345-
.066.089-.147.169-.203.265-.083.144-.133.304-.191.46-.031.085-.08.162-.104.25-
.067.249-.103.51-.103.776v38.979l-29.706 17.103V24.493a3 3 0 00-.103-.776c-.024-
.088-.073-.165-.104-.25-.058-.157-.108-.316-.191-.46-.056-.097-.137-.176-.203-
.265-.087-.117-.161-.242-.265-.345-.085-.086-.194-.148-.29-.223-.109-.085-.206-
.182-.327-.252l-.002-.001-.002-.002L40.098 1.396a2.971 2.971 0 00-2.964 0L1.487
21.919l-.002.002-.002.001c-.121.07-.219.167-.327.252-.096.075-.205.138-.29.223-
.103.103-.178.228-.265.345-.066.089-.147.169-.203.265-.083.144-.133.304-.191.46-
.031.085-.08.162-.104.25-.067.249-.103.51-.103.776v122.09c0 1.063.568 2.044
1.489 2.575l171.293 41.045c.156.089.324.143.49.202.078.028.15.074.23.095a2.98
2.98 0 001.524 0c.069-.018.132-.059.2-.083.176-.061.354-.119.519-.214l171.293-
41.045a2.971 2.971 0 001.489-2.575v-38.979l134.158-19.666a2.971 2.971 0 001.489-
2.575V44.666a3.075 3.075 0 00-.106-.774zM74.255 143.167l-29.648-16.779 31.136-
17.926.001-.001 34.164-19.669 29.674 17.084-21.772 12.428-43.555 24.863zm68.329-
76.259v33.841l-12.475-7.182-17.231-9.92V49.806l12.475 7.182 17.231 9.92zm2.97-
39.335l29.693 17.095-29.693 17.095-29.693-17.095 29.693-17.095zM54.06 114.089l-
12.475 7.182V46.733l17.231-9.92 12.475-7.182v74.537l-17.231 9.921zM38.614
7.398l29.693 17.095-29.693 17.095L8.921 24.493 38.614 7.398zM5.938 29.632l12.475
7.182 17.231 9.92v79.676l.001.005-.001.006c0
.114.032.221.045.333.017.146.021.294.059.434l.002.007c.032.117.094.222.14.334.05
1.124.088.255.156.371a.036.036 0
00.004.009c.061.105.149.191.222.288.081.105.149.22.244.314l.008.01c.084.083.19.1
42.284.215.106.083.202.178.32.247l.013.005.011.008 34.139 19.321v34.175L5.939
144.867V29.632h-.001zm136.646 115.235l-65.352 37.625V148.311l48.399-27.628
16.953-9.677v33.862zm35.646-61.221-29.706 17.102V66.908l17.231-9.92 12.475-
7.182v33.841z"/>
                        </g>
                    </svg>
                </div>

                <div class="mt-8 bg-white dark:bg-gray-800 overflow-hidden
shadow sm:rounded-lg">
                    <div class="grid grid-cols-1 md:grid-cols-2">
                        <div class="p-6">
                            <div class="flex items-center">
                                <svg fill="none" stroke="currentColor" stroke-
linecap="round" stroke-linejoin="round" stroke-width="2" viewBox="0 0 24 24"
```

Кафедра_КСПЗ_2021рік

```
class="w-8 h-8 text-gray-500"><path d="M12 6.253v13m0-13C10.832 5.477 9.246 5 7.5 5S4.168 5.477 3 6.253v13C4.168 18.477 5.754 18 7.5 18s3.332.477 4.5 1.253m0-13C13.168 5.477 14.754 5 16.5 5c1.747 0 3.332.477 4.5 1.253v13C19.832 18.477 18.247 18 16.5 18c-1.746 0-3.332.477-4.5 1.253"></path></svg>
                                <div class="ml-4 text-lg leading-7 font-semibold"><a href="https://laravel.com/docs" class="underline text-gray-900 dark:text-white">Documentation</a></div>
                            </div>

                            <div class="ml-12">
                                <div class="mt-2 text-gray-600 dark:text-gray-400 text-sm">
                                    Laravel has wonderful, thorough documentation covering every aspect of the framework. Whether you are new to the framework or have previous experience with Laravel, we recommend reading all of the documentation from beginning to end.
                                </div>
                            </div>
                        </div>

                        <div class="p-6 border-t border-gray-200 dark:border-gray-700 md:border-t-0 md:border-l">
                            <div class="flex items-center">
                                <svg fill="none" stroke="currentColor" stroke-linecap="round" stroke-linejoin="round" stroke-width="2" viewBox="0 0 24 24" class="w-8 h-8 text-gray-500"><path d="M3 9a2 2 0 012-2h.93a2 2 0 001.664-.89l.812-1.22A2 2 0 0110.07 4h3.86a2 2 0 011.664.89l.812 1.22A2 2 0 0018.07 7H19a2 2 0 012 2v9a2 2 0 01-2 2H5a2 2 0 01-2-2V9z"></path><path d="M15 13a3 3 0 11-6 0 3 3 0 016 0z"></path></svg>
                                <div class="ml-4 text-lg leading-7 font-semibold"><a href="https://laracasts.com" class="underline text-gray-900 dark:text-white">Laracasts</a></div>
                            </div>

                            <div class="ml-12">
                                <div class="mt-2 text-gray-600 dark:text-gray-400 text-sm">
                                    Laracasts offers thousands of video tutorials on Laravel, PHP, and JavaScript development. Check them out, see for yourself, and massively level up your development skills in the process.
                                </div>
                            </div>
                        </div>

                        <div class="p-6 border-t border-gray-200 dark:border-gray-700">
```

КафедраКБПЗ_2021рік

```
                        <div class="flex items-center">
                            <svg fill="none" stroke="currentColor" stroke-
linecap="round" stroke-linejoin="round" stroke-width="2" viewBox="0 0 24 24"
class="w-8 h-8 text-gray-500"><path d="M7 8h10M7 12h4m1 8l-4-4H5a2 2 0 01-2-
2V6a2 2 0 012-2h14a2 2 0 012 2v8a2 2 0 01-2 2h-3l-4 4z"></path></svg>
                            <div class="ml-4 text-lg leading-7 font-
semibold"><a href="https://laravel-news.com/" class="underline text-gray-900
dark:text-white">Laravel News</a></div>
                        </div>

                        <div class="ml-12">
                            <div class="mt-2 text-gray-600 dark:text-gray-
400 text-sm">
                                Laravel News is a community driven portal
and newsletter aggregating all of the latest and most important news in the
Laravel ecosystem, including new package releases and tutorials.
                            </div>
                        </div>
                    </div>

                    <div class="p-6 border-t border-gray-200 dark:border-
gray-700 md:border-l">
                        <div class="flex items-center">
                            <svg fill="none" stroke="currentColor" stroke-
linecap="round" stroke-linejoin="round" stroke-width="2" viewBox="0 0 24 24"
class="w-8 h-8 text-gray-500"><path d="M3.055 11H5a2 2 0 012 2v1a2 2 0 002 2 2 2
0 012 2v2.945M8 3.935V5.5A2.5 2.5 0 0010.5 8h.5a2 2 0 012 2 2 2 0 104 0 2 2 0
012-2h1.064M15 20.488V18a2 2 0 012-2h3.064M21 12a9 9 0 11-18 0 9 9 0 0118
0z"></path></svg>
                            <div class="ml-4 text-lg leading-7 font-semibold
text-gray-900 dark:text-white">Vibrant Ecosystem</div>
                        </div>

                        <div class="ml-12">
                            <div class="mt-2 text-gray-600 dark:text-gray-
400 text-sm">
                                Laravel's robust library of first-party
tools and libraries, such as <a href="https://forge.laravel.com"
class="underline">Forge</a>, <a href="https://vapor.laravel.com"
class="underline">Vapor</a>, <a href="https://nova.laravel.com"
class="underline">Nova</a>, and <a href="https://envoyer.io"
class="underline">Envoyer</a> help you take your projects to the next level.
Pair them with powerful open source libraries like <a
href="https://laravel.com/docs/billing" class="underline">Cashier</a>, <a
href="https://laravel.com/docs/dusk" class="underline">Dusk</a>, <a
href="https://laravel.com/docs/broadcasting" class="underline">Echo</a>, <a
```

Кафедра_КБПЗ_2021рік

```
href="https://laravel.com/docs/horizon" class="underline">Horizon</a>, <a
href="https://laravel.com/docs/sanctum" class="underline">Sanctum</a>, <a
href="https://laravel.com/docs/telescope" class="underline">Telescope</a>, and
more.
                                </div>
                            </div>
                        </div>
                    </div>

                    <div class="flex justify-center mt-4 sm:items-center sm:justify-
between">
                        <div class="text-center text-sm text-gray-500 sm:text-left">
                            <div class="flex items-center">
                                <svg fill="none" stroke-linecap="round" stroke-
linejoin="round" stroke-width="2" viewBox="0 0 24 24" stroke="currentColor"
class="-mt-px w-5 h-5 text-gray-400">
                                    <path d="M3 3h2l.4 2M7 13h10l4-8H5.4M7 13L5.4
5M7 13l-2.293 2.293c-.63.63-.184 1.707.707 1.707H17m0 0a2 2 0 100 4 2 2 0 000-
4zm-8 2a2 2 0 11-4 0 2 2 0 014 0z"></path>
                                </svg>

                                <a href="https://laravel.bigcartel.com" class="ml-1
underline">
                                    Shop
                                </a>

                                <svg fill="none" stroke="currentColor" stroke-
linecap="round" stroke-linejoin="round" stroke-width="2" viewBox="0 0 24 24"
class="ml-4 -mt-px w-5 h-5 text-gray-400">
                                    <path d="M4.318 6.318a4.5 4.5 0 000 6.364L12
20.364l7.682-7.682a4.5 4.5 0 00-6.364-6.364L12 7.636l-1.318-1.318a4.5 4.5 0 00-
6.364 0z"></path>
                                </svg>

                                <a href="https://github.com/sponsors/taylorotwell"
class="ml-1 underline">
                                    Sponsor
                                </a>
                            </div>
                        </div>

                        <div class="ml-4 text-center text-sm text-gray-500 sm:text-
right sm:ml-0">
                            Laravel v{{ Illuminate\Foundation\Application::VERSION
}} (PHP v{{ PHP_VERSION }})
```

Кафедра_КбПЗ_2021рік

```
                    </div>
                </div>
            </div>
        </div>
    </body>
</html>
```

Кафедра_КбПЗ_2021рік

'VerifyEmailController.php – емейл верифікація'

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Providers\RouteServiceProvider;
use Illuminate\Auth\Events\Verified;
use Illuminate\Foundation\Auth\EmailVerificationRequest;

class VerifyEmailController extends Controller
{
    /**
     * Mark the authenticated user's email address as verified.
     *
     * @param  \Illuminate\Foundation\Auth\EmailVerificationRequest  $request
     * @return \Illuminate\Http\RedirectResponse
     */
    public function __invoke(EmailVerificationRequest $request)
    {
        if ($request->user()->hasVerifiedEmail()) {
            return redirect()-
>intended(RouteServiceProvider::HOME.'?verified=1');
        }

        if ($request->user()->markEmailAsVerified()) {
            event(new Verified($request->user()));
        }

        return redirect()->intended(RouteServiceProvider::HOME.'?verified=1');
    }
}
```

Кафедра_КбПЗ_2021рік

'RegisteredUserController.php – реєстрація нового учня'

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Models\User;
use App\Providers\RouteServiceProvider;
use Illuminate\Auth\Events\Registered;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;

class RegisteredUserController extends Controller
{
    /**
     * Display the registration view.
     *
     * @return \Illuminate\View\View
     */
    public function create()
    {
        return view('auth.register');
    }

    /**
     * Handle an incoming registration request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\RedirectResponse
     *
     * @throws \Illuminate\Validation\ValidationException
     */
    public function store(Request $request)
    {
        $request->validate([
            'name' => 'required|string|max:255',
            'email' => 'required|string|email|max:255|unique:users',
            'password' => 'required|string|confirmed|min:8',
        ]);

        $user = User::create([
            'name' => $request->name,
            'email' => $request->email,
            'password' => Hash::make($request->password),
        ]);
```

Кафедра_КбПЗ_2021рік

```
            event(new Registered($user));

            Auth::login($user);

            return redirect(RouteServiceProvider::HOME);
        }
    }
```

Кафедра_КбПЗ_2021рік

‘ConfirmablePasswordController.php – підтвердження паролю’

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Providers\RouteServiceProvider;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Validation\ValidationException;

class ConfirmablePasswordController extends Controller
{
    /**
     * Show the confirm password view.
     *
     * @return \Illuminate\View\View
     */
    public function show()
    {
        return view('auth.confirm-password');
    }

    /**
     * Confirm the user's password.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return mixed
     */
    public function store(Request $request)
    {
        if (! Auth::guard('web')->validate([
            'email' => $request->user()->email,
            'password' => $request->password,
        ])) {
            throw ValidationException::withMessages([
                'password' => __('auth.password'),
            ]);
        }

        $request->session()->put('auth.password_confirmed_at', time());

        return redirect()->intended(RouteServiceProvider::HOME);
    }
}
```

Кафедра_КбПЗ_2021рік