НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ПОГОДЖЕНО
Декан факультету
Інформаційних технологій
Глазунова О.Г., д.пед.н, проф.
підпис ПІБ, вчене звання і ступінь
«___» 2021 р.

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри
Комп'ютерних систем і мереж
Лахно В.А., д.т.н., проф.
підпис ПІБ, вчене звання і ступінь
«___» 2021 р.

МАГІСТЕРСЬКА РОБОТА

На тему: «Дослідження комплексної системи контролю доступу до приміщення на базі МК ESP32 та STM32. Програмна частина»

Спеціальність 123 «Комп'ютерна інженерія»

Освітня програма «Комп'ютерні системи та мережі»

Орієнтація освітньої програми _____

Керівник магістерської роботи: _____ / Лахно В.А. /
підпис ПІБ

Виконав: _____ / Редько В.П. /
підпис ПІБ

КИЇВ-2021

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

«ЗАТВЕРДЖУЮ»
завідувач кафедри
комп'ютерних систем і мереж
_____ / Лахно В.А., д.т.н., проф. /
підпис ПІБ, вчене звання і ступінь
«__» _____ 20__ р.

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ РОБОТИ СТУДЕНТУ

Редько Владислав Петрович
(прізвище, ім'я, по батькові)
Спеціальність (напрям підготовки): комп'ютерна інженерія
Освітня програма: комп'ютерні системи та мережі
Орієнтація освітньої програми:

Тема магістерської роботи: «Дослідження комплексної системи контролю доступу до приміщення на базі МК ESP32 та STM32. Програмна частина»

затверджена наказом ректора НУБіП України від "23" жовтня 2020р. №
Термін подання завершеної роботи на кафедру 20 листопада 2021р.
Вихідні дані до магістерської роботи мови програмування PHP, JS, фреймворки Laravel, система контейнеризації Docker,

Перелік питань, що підлягають дослідженню:
1. Аналітичний огляд
2. Вимоги до системи
3. Проектування та реалізація компонентів системи

Перелік графічного матеріалу (за потреби) діаграма нефункціональних вимог, діаграма прецедентів, діаграми послідовності, діаграми класів сутностей системи

Дата видачі завдання "23" жовтня 2020 р.

Керівник магістерської роботи _____ Лахно В.А., д.т.н., проф.
(підпис) (прізвище та ініціали)
Завдання прийняв до виконання _____ Редько В.П.
(підпис) (прізвище та ініціали студента)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	23.10.2020	Виконано
2	Проектування системи	15.12.2020	Виконано
3	Реалізація системи	1.03.2021	Виконано
4	Тестування системи	16.06.2021	Виконано
5	Оформлення пояснювальної записки	01.07.2021	Виконано
6	Оформлення графічного матеріалу	10.08.2021	Виконано

Студент _____ **Редько В.И.**
(підпис) (ініціали та прізвище)

Керівник проекту (роботи) _____ **Лахно В.А.**

РЕФЕРАТ

Пояснювальна записка: 57 сторінок, 11 рисунків, 2 таблиці, 2 додатки, 11 джерел.

КОМП'ЮТЕРНА СИСТЕМА, КОНТРОЛЬ ДОСТУПУ, ПРОЕКТУВАННЯ, PHP, LARAVEL, DOCKER, BACK-END, FRONT-END, HTTPS, DOCKER, СЕРВІС, МОДЕЛЬ

Предметом дослідження є технології розроблення автоматизованих інформаційних систем для підприємств.

Об'єктом дослідження виступає процес автоматизації контролю доступу на підприємствах.

Мета магістерської роботи полягає в проектуванні та розробці автоматизованої системи контролю доступу до приміщень.

Завданнями магістерської роботи є прискорення процесу перевірки наявності студентів на заняттях за допомогою автоматизованої системи контролю відвідування студентів.

Апаратні та програмні засоби, що використовувались при проектуванні: draw.io, NinjaMock, Microsoft Visio, LucidChart, MacBook Pro 2019.

Результати досягнуті в процесі роботи – було розроблено автоматизовану систему контролю доступу до приміщень та проведено тестування створеної системи, яке показало, що дана система є повністю працездатною та виконує усі вказані функції.

Одержані результати можуть бути використані у усіх підприємствах, адже дана система є універсальною.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

АС – автоматизована система

xml – розширювана мова розмітки, Extensible Markup Language

QR – двовимірний штрих-код, quick response

front-end – інтерфейс для взаємодії між користувачем і back end

PHP – скриптова мова програмування, Hypertext Preprocessor

CSS – каскадні таблиці стилів, Cascading Style Sheets

HTML – мова розмітки гіпертексту, HyperText Markup Language

JS – JavaScript

MBSE – системна інженерія на основі моделей, model based systems engineering

СКБД – система керування базами даних

БД – база даних

IOS – мобільна операційна система, iPhone OS

IEEE – інститут інженерів з електротехніки та електроніки, Institute of Electrical and Electronics Engineers

UML – уніфікована мова моделювання, Unified Modeling Language

SysML – мова моделювання загального призначення для застосувань в системній інженерії, Systems Modeling Language

ORM – об'єктно-реляційна проекція, object-relational mapping

HTTP – протокол передачі гіпер-текстових документів, Hyper Text Transfer Protocol

API – програмний інтерфейс застосунку, Application Programing Interface

URL – уніфікований локатор ресурсів, Uniform Resource Locator

IDE – інтегроване середовище розробки, Integrated Development Environment

ВСТУП

Безпека будь-якого об’єкта має кілька етапів, кількість яких залежить від рівня режимності об’єкту. Але найважливішим з цих етапів завжди буде система управління контролю доступом.

Організована правильним чином СКУД дозволяє вирішувати низку завдань.

При реалізації конкретних СКУД використовують різні методи і реалізують власні пристрої для ідентифікації та аутентифікації. Слід зазначити, що СКУД є одним з найбільш розвинених сегментів ринку безпеки. За оцінками деяких експертів, щорічне зростання ринку СКУД становить понад 25%. Кількість спеціалістів, які працюють у сфері технічних систем безпеки, перевищила 500 тисяч осіб. Це пов’язано, по-перше, зі зростанням обізнаності про український ринок і, як наслідок, попитом на нові функції, функції та послуги, які були впроваджені в старіших класичних сегментах ринку систем безпеки. По-друге, на збільшення динаміки ринку істотно впливають такі фактори, як підвищення ризику терористичних загроз, підвищення загального рівня культури споживача (підвищення вимог до якості та можливостей систем, комплексні рішення привертають увагу тощо).

Часи ентузіастів, які хочуть виступити в якості полігону для тестування нових технологій, давно минули, і сьогодні СКУД сприймається більшістю користувачів як важливий компонент безпеки підприємства. Значну роль у досягненні цього результату відігравала і відіграє поінформованість кінцевих користувачів. З ростом усвідомлення, як правило, об'єкт підприємства вимагає рівня вимог до СКУД, наприклад, об'єктам необхідно підтримувати більш високий рівень безпеки (аеропорти, атомні об'єкти, промислові підприємства), почали використовувати біометричні системи ідентифікації (за відбитком пальця, формою долоні, райдужною оболонкою, рисами особи, у тому числі й

багатофакторний – за комбінацією біометричних ознак та пароля чи карти доступу.

1 Аналітичний огляд

1.1 Дослідження предметної області

Система контролю та управління доступом (СКУД) зазвичай складається з серверів СКУД, в залежності від навантаження та розгалуженості керованої мережі, що роль може грати як старий ноутбук, так і найсучасніший, найпотужніший кластер серверів - і керувати підключеними Контролери СКУД. панель) — це спеціалізований високонадійний комп'ютер, який зберігає інформацію про конфігурацію, режими системи, список людей, які мають доступ до ресурсу, та їхні привілеї для доступу до цього ресурсу. зчитувач, турнікет, замок або інший привід (рис. 1.1).

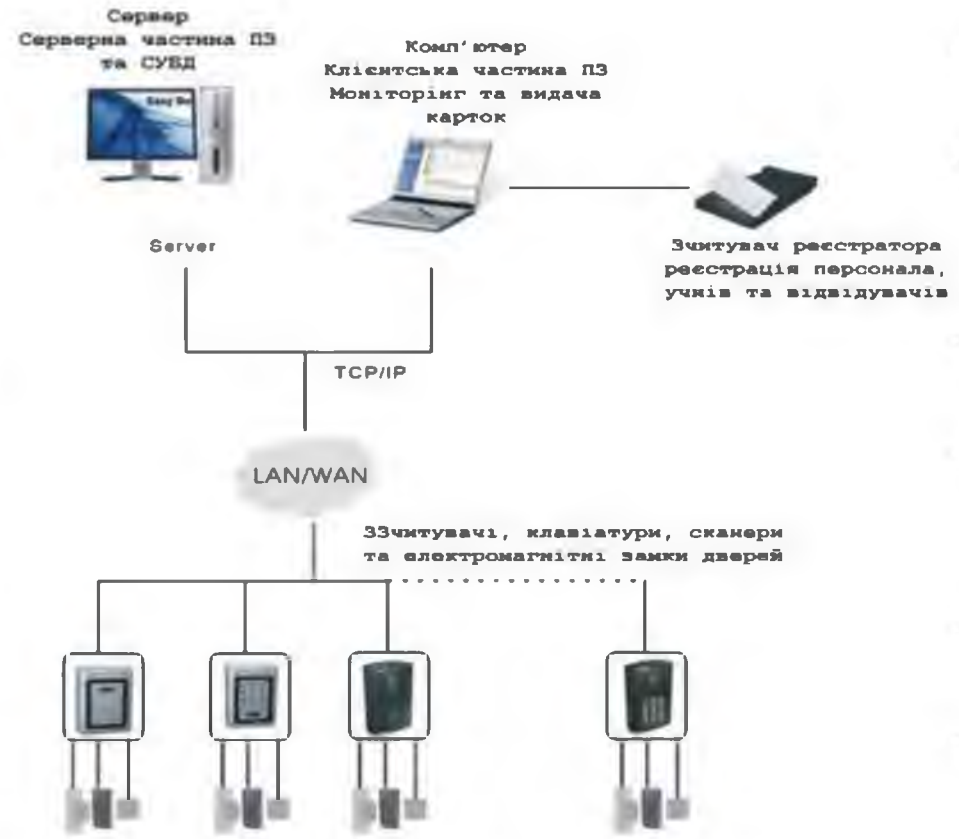

Рис. 1.1. Склад СКУД

Блокувальні пристрої (APD) – це пристрої, що створюють фізичні бар'єри доступу та оснащені приводами для контролю їх стану (турнікети, прохідні кабіни, двері та ворота, обладнані приводами САУ). Зчитувач, зчитувач – пристрій, призначений для зчитування (введення) ідентифікаційних ознак. Він передає цю інформацію контролеру, який приймає рішення про допуск особи на ресурс. Ви можете налаштувати контролер, щоб запитувати комп'ютер про підтвердження рішення. Для підвищення надійності ідентифікації до контролера, крім зчитувачів, можна підключити клавіатуру для набору персонального ідентифікаційного номера (PIN). Іншою важливою концепцією ACS є ідентифікатор користувача - унікальна ознака суб'єкта або об'єкта доступу. Як ідентифікатор можна використовувати код, біометричну функцію або фізичний код. Ідентифікатор, що використовує матеріальний код – це об'єкт, до якого (на який) за допомогою спеціальної технології вводиться ознака ідентифікації у вигляді кодової інформації (картки, електронні ключі, брелоки тощо). Іншим типом пристрою, який можна підключити до контролера, є панель безпеки. Це також спеціалізований контролер, який контролює стан датчиків безпеки (датчики на дверях, вікнах, датчики гучності та інші). Якщо стан будь-якого датчика змінюється, інформація про нього негайно надходить до головного контролера. Приводи - це пристрої або механізми, що забезпечують відкриття або закриття БЛП (електромеханічні, електромагнітні замки, електромагнітні засувки, механізми приводу замків, воріт, турнікетів та інших подібних пристроїв). В них може бути набір реле, за допомогою яких вони здійснюють управління виконавчими пристроями: електромеханічними замками, турнікетами, ліфтами, автоматичними воротами та іншими (рис. 1.2).

Рис. 1.2. Різноманітність засобів обмеження доступу

Нижче наведено основні особливості встановлення СКУД на захищений об'єкт. Контроль та керування доступом є основною функцією системи, вона використовується для поділу прав доступу у студентів, співробітників і відвідувачів до певних приміщень, а також для заборони доступу небажаним особам. Крім того, є можливість дистанційно керувати запірними пристроями (замками, турнікетами тощо). СКУД дозволяє заборонити прохід співробітників у святкові та вихідні дні, а також у позаурочний або робочий час. Збір та надання статистичних даних. СКУД збирає інформацію про осіб, які пройшли через певні точки контролю доступу. Для кожного працівника можна отримати таку інформацію: час входу та виходу, спроби проникнення в заборонені приміщення та зони, а також спроби проходу в несанкціонований час. Також є можливість відстежити переміщення працівника по території, вказавши місце і час. Таким чином, всі виявлені порушення трудової дисципліни можуть бути зафіксовані в особовій справі працівника, а керівництво правопорушника повідомляється в наказі про роботу. Крім того, на основі інформації про останній пункт проходження СКУД дозволяє в будь-який момент визначити місцезнаходження співробітника.

Доступ до контрольованих зон тільки за ідентифікатором. При проходженні через ідентифікаційну картку вся інформація про працівника та його фото може відображатися на екрані безпеки на екрані монітора, що виключає можливість передачі чужої картки. Також на рівні правил реагування СКУД можна забезпечити захист від передачі ідентифікатора іншій особі та заблокувати повторний вхід на територію об'єкта з тієї ж карти доступу. За допомогою розробленої системи обліку робочого часу реєструється час прибуття до школи робітників та учнів та час їх відбуття. В результаті надається можливість визначити сумарний час перебування співробітника на підприємстві, а на самому початку дня, наприклад, о 10:00 система обліку робочого часу, вбудована в СКУД, може формувати груповий звіт про працівників, які не пройшли через контрольну точку входу на територію. Це дозволяє в масовому порядку виявляти тих, хто запізнився або не з'явилися і може зняти навантаження про контроль. Подібний звіт можна отримати в кінці робочого дня на виході. Охоронні телевізійні (відео) системи та системи контролю доступу. СКУД оснащений системою безперебійного живлення, що дозволяє не переривати роботу у разі відключення електроенергії в будівлі. Також система контролю доступу завдяки функціональності контролера має можливість продовжити роботу, наприклад, у разі відмови керуючого комп'ютера. Захист об'єктів у режимі реального часу. СКУД дає змогу охороняти певні приміщення та знімати їх з-під охорони. Крім того, в режимі реального часу ви можете отримувати інформацію про всілякі надзвичайні ситуації та тривоги за допомогою спеціальних сповіщень відповідальних осіб. Крім того, в системній базі фіксуються всі тривожні події та події, що дозволяє отримати доступ до цієї інформації в майбутньому, якщо буде потрібно. Завдяки засобам, наявним в СКУД, охоронець зі свого робочого місця за допомогою комп'ютера має можливість не тільки керувати дверима та турнікетами, а й бити тривогу. У комп'ютер СКУД у співробітника охорони можуть бути занесені поверхові плани будівлі зі схемою розташування контролерів обмеження доступу. Віддалене управління системою через локальну мережу або мережу Інтернет або навіть з

мобільного телефону. Якщо при установці підключити СКУД до мережі Інтернет, то у адміністрації школи, або керівника служби охорони, з'являється можливість вести віддалене управління і контроль за роботою системи, також можна цю інформацію надавати диспетчеру служби охорони. Також можна сказати і про можливість управління СКД зі свого мобільного телефону, правда це більше відноситься до GSM систем контролю доступу. Інтеграція СКУД і СКД з іншими системами безпеки і охорони. Системи контролю і управління доступом можна поєднати і вбудувати з іншими системами безпеки, наприклад системою відеоспостереження, охоронною та пожежною сигналізацією. Контроль доступом разом з відеоспостереженням забезпечують майже повний контроль над охоронюваними приміщеннями. При виникненні позаштатної ситуації така система в найкоротші терміни дозволить виявити і заблокувати порушника. При інтеграції СКУД і охоронної сигналізації є можливість налаштувати спільну реакцію системи на несанкціоноване проникнення в те чи інше приміщення. Наприклад, можна включити сирену на пункті охорони, тривожну лампу або ж і зовсім заблокувати всі двері в необхідної частини будівлі. Інтеграція СКУД з системою пожежної сигналізації дозволяє автоматично розблокувати двері, турнікети і прохідні в разі пожежі, щоб не треба було бігати по задимлених коридорах та шукати когось, в кого є ключ від пожежного виходу. Всі ці заходи значно спрощують евакуацію персоналу в такий важкий період.

Функціонування систем контролю та управління доступом базується на принципі порівняння певних ідентифікаційних ознак конкретної особи чи об'єкта з даними, закладеними в систему. Кожен співробітник отримує картку доступу або брелок з індивідуальним кодом, який присвоюється при видачі карти доступу до КПП. Біометричні дані людини також можна використовувати як код. При проході на територію, що охороняється або в приміщення, що охороняється, проводиться зчитування даних з носія коду через зчитувачі, встановлені біля дверей. Інформація про відвідувача передається в систему, де проводиться аналіз та дається сигнал, адекватно реагує на ситуацію, що склалася. «Прохід

дозволено»,« Прохід заборонено», «Повторний прохід по одній карті», виведення сигналу «Тривога» на пульт охоронця при порушенні охоронюваної території без відповідних прав і т.і. При необхідності втручання охорони в ситуацію на екрані комп'ютера посту охорони виводиться сигнал тривоги та інструкції, які визначають дії особового складу в цій ситуації. Крім того, система може миттєво реагувати на тривожну ситуацію, замикаючи замки в приміщенні, що охороняється, і проходячи через точки доступу. Щоб проаналізувати події, ви можете переглянути та роздрукувати журнал подій за певний період часу. Для виключення зловживань в використанні карт і посилення прохідного режиму в особливо важливі зони є ряд функцій, що дозволяють: – виключити подвійний прохід в зону по одній карті (розрізняють можливості блокування повторного проходу на певний час – для систем, які не є обладнані зчитувачами на виході і заборона на вхід в несуміжні зони для повних систем контролю доступу); – дозволити доступ тільки по 2-м картками (увійти можуть тільки двоє людей, зустрівшись разом і що володіють відповідними повноваженнями); – обмежити кількість осіб в приміщенні і зоні (при перевищенні встановленого порогового значення контролер не пропустить в зону чергового людини); – встановити режим «вхід під примусом» (непомітно для оточуючих охороні подається сигнал тривоги); – охоронцеві дається право на самостійне прийняття рішення про дозвіл на прохід відвідувача (при зчитуванні карти на монітор охоронця виводиться фотографія власника, яка звіряє з зображенням, що видаються відеокамерою); – встановити режим лічильника на використання карти (кількість читань карти на конкретному зчитувачі обмежується); – встановити прихований контроль в приміщенні (подати сигнал тривоги на пульт охорони при проникненні в приміщення, що підлягає і відсутності відповідних прав, причому для зловмисника факт виявлення залишається невідомим).

1.2 Призначення програмного продукту

Система доступу до приміщень – інтернет додаток, що призначений для працівників будь-яких компаній. Основна конкурентна особливість – велика швидкодія, надійність та відмовостійкість, що для систем такого типу є основним. Додаток повинен ознайомити відвідувача з можливостями сайту та їх властивостями, надати корисну інформацію користувачеві. Організувати зручну та інтуїтивно зрозумілу взаємодію свіпробітників з додатком з метою контролю доступу до приміщень. Цільовою категорією є чоловіки та жінки від 14 до 45 років з активним способом життя, які проживають у містах України. Рівень матеріального стану не важливий.

Цілі створення системи:

– автоматизація контролю доступу;
– скорочення часу на налаштування системи та встановлення нових пунктів контролю.

Завдання інтернет-сайту:

– викликати у користувача відчуття довіри до підприємства;
– можливість реєструвати час прибуття працівників до приміщення з максимальною автоматизацією процесу;
– надання інформації про приміщення в зручній формі;
– скорочення взаємодії користувачів з співробітниками;

1.3 Огляд існуючих засобів

Серед найпотужніших засобів автоматизації контролю доступу слід виділити ZkTeco, SIGUR, ControlGate та HID Global. Немає ніяких сумнівів, що всі ці системи досить функціональні і складні, але кожна з них має певні

особливості в експлуатації, тому далі в цьому розділі буде описано докладний опис їх можливостей, і на основі порівняння одна з них буде обрана в якості прикладу для роботи.

1.3.1 ZkTeco

Китайська компанія «ZkTeco Co., Ltd» широко відома своєю спрямованістю на біометричні методи верифікації і надає послуги для великого і середнього бізнесу. Підприємство володіє понад 90 філіями по всьому світу, в т. ч. Росії і країнах СНД[1].

Програмне забезпечення базується на хмарній платформі ZKTeco + Smart Office. Воно підтримує біометричну ідентифікацію, відеоспостереження, а також інтелектуальну систему входу і виходу «людина + автомобіль + об'єкт». За ZKTime.Net V3.0 призначена для роботи з системою обліку робочого часу компанії, включає в себе можливості розпізнавання осіб, відбитків і гібридну біометрію. На основі одержуваної інформації, програма здатна створювати звіти і платіжні відомості. Крім цього, враховуючи реалії часу, в компанії розробляють методи безконтактної ідентифікації COVID-19, що також інтегрується в програмне забезпечення через відповідні модулі.

1.3.2 SIGUR

Спеціальне рішення SIGUR для адміністративних будівель і бізнес-центрів дозволяє забезпечити контроль доступу в приміщення і на прилеглі до будівлі території (наприклад, парковки).

Необмежена кількість робочих місць для брендарів, а також готові АРМ служби охорони, бюро перепусток. Можливість організації автоматичної видачі та збору пропусків відвідувачів і гнучка масштабованість системи.

1.3.3 ControlGate

Російська компанія "КонтролГейт" спеціалізується на програмуванні і розробці систем безпеки. ПЗ ControlGate включає в себе повний набір модулів

для проведення спостереження, контролю обладнання, управління доступом, а також аналізу та збору інформації. Система безпеки об'єкта під управлінням ПЗ ControlGate здатна працювати з широким спектром ідентифікаторів, в т. ч. і біометричних. При необхідності, є можливості комбінувати між собою кілька типів ідентифікаторів. Наприклад, біометричний термінал і зчитувач ПІН-коду. Масштабування системи здійснюється за рахунок підключення додаткових модулів.

Основними плюсом софту є його кроссплатформенність, тобто можливість працювати на різних операційних системах (Windows, Linux, MacOs). Це дозволяє інтегруватися з обладнанням таких виробників як: Hikvision, Dahua, ZKTeco, IronLogik.

СКУД ControlGate побудована на принципах модульності. Існує основний базовий блок, до якого при необхідності підключаються Додаткові (інформатор, бюро перепусток, СУРВ, обмін даними). Яскравим плюсом служить те, що в базовому модулі встановлені наступні компоненти: безпосередньо контроль доступу, відеосистема, генератор звітів, середовище виконання призначених для користувача сценаріїв, планувальник завдань[2].

1.3.4 HID Global

Компанія "HID Global" являє світовим лідером з виробництва СКУД. Головний офіс компанії розташований в м. Остін, штат Техас (США).

Програмне забезпечення компанії має досить широку спрямованість і використовується в СКУД для фізичного і логічного доступу персоналу, а також у сфері випуску фінансових інструментів. Софт від HID Global використовується для реєстрації біометричних параметрів і аутентифікації, випуску ідентифікаційних карт для банківського і фінансового секторів. Також він широко застосовується в операційних системах для мікрочіпів, а також у сфері Internet of Things (IoT).

Разом з тим при всій різноспрямованості і плюсах розробок HID, в ПЗ є істотний мінус – його ціна. Втім, висока вартість програмного забезпечення характерна практично для всіх продуктів зарубіжного виробництва[3].

1.4 Функції програмного продукту

Система має працювати на змішаній системі, яка складається з пристрою зчитування карток та веб додатку. Користувач має приходити до приміщення та прикладати картку до спеціального пристрою. Після чого при введені персональних даних користувач отримує доступ до приміщення. Кожне приміщення має бути оснащена пристроєм для зчитування карток.

1.5 Вибір мови програмування

Перш ніж приступати до вивчення мов і написання коду, важливо розібратися в значенні цих двох термінів. Фронтенд фахівці займаються клієнтською стороною-тобто тим, що побачить Користувач.

Бекенд-це програмно-апаратна частина сервісу, те, що працює на сервері. Залежно від спеціалізації програміст задіє різні технології створення сайту. Фронтенд-розробники зазвичай не обходяться без HTML, CSS і JavaScript. Для Backend «must have» – PHP, Python, Ruby.

1.5.1 JavaScript

Одна з найпоширеніших мов. Часто початківці програмісти плутають Java і JavaScript. Незважаючи на співзвучну назву, це дві абсолютно різних мови. Область його застосування обширна і практично безмежна. На JavaScript пишуть серверні, мобільні та комп'ютерні програми. Будь-який браузер і будь-яка

операційна система добре знайома з JavaScript. Всі сценарії виконуються безпосередньо в браузері пристрою, користувачеві не потрібно робити будь-яких дій. У більшості випадків він використовується для створення простих анімацій, скриптів і об'єктів користувальницького інтерфейсу.

1.5.2 PHP

Головна перевага PHP-код мови не конфліктує з HTML верстком і може використовуватися одночасно для розмітки зовнішнього вигляду сторінки за допомогою HTML-тегів і функціоналу сторінки php-частиною. Він легкий в освоєнні практично на всіх етапах вивчення. Відрізняється розвиненою підтримкою даних, підходить під апаратні платформи і відомі ОС. Ця мова програмування призначена спеціально для роботи на стороні сервера. Бібліотека мови підходить для завдань, що виконуються багаторазово під час розробки сайту.

1.5.3 Python

Ця мова багато фахівців вважають ідеальним в Data Science (методика аналізу даних з використанням машинного навчання і штучного інтелекту). Одним з головних плюсів Python вважається його простота. При наявності бажання, з його особливостями і тонкощами програмування зможе розібратися кожен бажаючий. До того ж Python сприяє економії часу програміста, так як пропонує велике число спеціальних бібліотек з уже готовими програмними конструкціями.

1.5.4 Ruby

Основне призначення Ruby – формувати і програмувати сайти, а також мобільні додатки. Навколо мови Ruby склалася думка про його повільність і неможливість масштабувати великі проекти. На самому початку існування в плані продуктивності Ruby дійсно поступався PHP і Python. Однак численні оновлення мови докорінно виправили ситуацію, прийдешні апгрейди повинні

принести й інші зміни – можливість роботи з паралельними потоками. Повільність роботи сучасного додатка на Ruby цілком залежить від здібностей програміста і правильності побудови архітектури. З переваг можна відзначити легкість вивчення мови початківцям фахівцем, часто його використовують завдяки простим методам запису.

1.5.5 C#

Мова програмування C# перейняв багато від Java і C++. Більше половини його синтаксичних можливостей ідентичні з мовою Java. Спочатку використовувався як засіб розробки веб-сайтів. Відзначимо, що сьогодні C# активно розвивається, виходять оновлення і доповнення, з'явилися асинхронні методи, динамічні зв'язування. Якщо порівнювати його з іншими популярними мовами, то можна відзначити відносну молодість c#: його перша версія з'явилася в 2002 році.

1.5.6 Perl

Мови програмування для веб-розробки складно уявити без Perl. У самих витоках виникнення, Perl призначався для позбавлення від необхідності написання різних програм і сценаріїв на різних мовах, об'єднуючи можливості системного адміністрування та обробки документів в єдине мовне середовище. На поточний момент-Perl активно використовується при написанні інтерактивних додатків, адміністрування серверів і адаптований до всіх популярних платформ - Windows, Mac та інші.

1.5.7 Java

Мова, найчастіше використовується з метою створення мобільних додатків, мережевих програм. Вважається основною мовою розробки для Android. Мова йде в ногу з часом і сьогодні актуальна як ніколи. Він включає об'єктно-орієнтоване програмування (ООП) – методику спрощення складного коду, при якому ділянка коду з конфліктуючими один з одним функціями

ділиться на незалежні об'єкти, кожен з яких містить в собі ті ж функції і дані, які активуються при безпосередньому зверненні до них, а не одночасно, створюючи конфлікт (як при процедурному програмуванні). До інших переваг Java варто віднести безпеку, надійність і простий синтаксис.

2 Вимоги до системи

2.1 Бізнес-вимоги до системи

Оскільки, основною задачею є створення системи для контролю доступу до приміщень потрібно сформувати основні бізнес вимоги:

а) Система повинна сприяти та допомагати в створенні безпечного знаходження на території підприємства чи захисту певної території від несанкціонованого перебування.

б) За допомогою системи пройде етап цифровізації процесів в усіх підприємствах.

в) Система полегшить здійснення контролю та обліку знаходження працівників у певних частинах підприємства.

г) Система повинна допомагати в прийнятті рішень щодо попередження несанкціонованого знаходження на приватній території.

д) Створення єдиного цифрового середовища для створення, редагування доступу та перегляду інформації про отримання доступу.

2.2 Функціональні вимоги до системи

Система буде мати модульну структуру, що має такий набір модулів:

- список пристроїв;
- список користувачів;
- журнал отримання доступу;
- особистий кабінет;
- звіти.

Набір модулів.

Загальні вимоги:

1. АС повинна мати авторизацію по логіну та паролю.
2. Нагадування користувачу паролю по номеру мобільного телефону.
3. Відправлення повідомлень працівникам.

Модуль Список пристроїв:

1. Створення нового пристрою у системі.
2. Видалення пристроїв з системи.
3. Налаштування спеціального паролю до пристрою.
4. Налаштування доступу працівників до кожного пристрою.

Модуль Список користувачів:

1. Додавання користувача до системи.
2. Видалення користувачів з системи.
3. Редагування персональної інформації користувача.
4. Перегляд доступу кожного користувача до пристроїв.

Модуль Журнал отримання доступу:

1. Перегляд інформації про усі сеанси знаходження працівників на певній території.
2. Створення та видалення записів про отримання доступу.

Модуль Особистий кабінет:

1. Заповнення/редагування персональних даних користувача.
2. Додавання фотографії користувача.
3. Виведення журналу доступу користувача до певних пристроїв.

Модуль Звіти:

1. Формування звіту отримання доступу до певного пристрою.
2. Формування звіту отримання доступу певного користувача.

2.3 Нефункціональні вимоги до системи

Можливість повторного використання**:** вимоги до повторного використання реалізації або компонентів програми або системи (Reusability) відносяться до ключових аспектів проектування програми або системи (design time).

Найчастіше дана вимога має місце коли загальні компоненти використовуються декількома модулями системи, що розробляється.

Можливість повторного використання зустрічається в архітектурі, функціях, модулях, класах, дизайні, базах даних (коди доступів, користувачі), сценаріях тощо.

Компоненти повинні бути загальними, абстрактними і не дуже розумними. (важко замінити компоненти, які занадто багато знають про загальну систему.)

Вважається, що компоненти, які залежать від множини інших компонентів, обтяжені. Обтяжені компоненти також складно використовувати повторно, оскільки всі компоненти постачальника також повинні бути імпортовані в нову систему, зображено на рис .2.1.

Рис. 2.1 – Діаграма нефункціональних вимог до автоматизованої системи контролю відвідування

До нефункціональних вимог входять:

– система повинна адаптуватися під різний розмір екрану мобільного пристрою/планшету від 4 до 15 дюймів;

– інструментами розробки мають бути PHP, CSS, HTML, JS, в якості СКБД – MySQL;

– в системі повинно бути передбачено контроль введення даних щодо віку, групи та факультету;

– особисті дані користувача повинні шифруватися у відповідності зі стандартами інформаційної безпеки 268-с і 566-а;

– система може мати звукові ефекти, наприклад при формуванні звіту і досягненні пари вчасно видаватиметься звук «аплодисменти».

– всі права розробника захищені відповідно до Закону України від 23.12.1993 № 3792-XII «Про авторське право і суміжні права»;

– у разі виникнення відсутності з'єднання з сервером БД всі введені дані користувача повинні зберігатися на жорсткому диску пристрою і при наявності з'єднання передаватися в БД;

– система повинна займати обсяг оперативної пам'яті пристрою не більше 300 Мб;

– швидкість оброблення запиту повинна бути не більше 0,002 сек;

– середній час відновлюваності системи має складати не більше 180 секунд;

– щотижневий або щодневий звіт повинен бути у форматі HTML для відображення у веб-браузері та xls для перегляду у Exel та pdf;

– в системі повинні бути наступні модулі: особистий кабінет, звіти, опитування, показники;

– керівництво користувача по проєкті з системою повинно бути вбудоване в саму систему і викликатися відповідним пунктом меню;

– дані в системі повинні бути сумісними з різними версіями системи;

– всі дії користувача в системі повинні записуватися в журнал;

– для коректної роботи системи необхідно щоб на пристрої було не менше 100 мегабайт вільного простору;

– система може бути розширено до інтелектуальної системи, що має вбудовані методи штучного інтелекту та зчитування даних з різних пристроїв, наприклад з смартфону;

– система повинна працювати на різних версія ОС Windows, Linux, Arduino та IOS;

– система повинна розгортається на мобільному пристрою або планшеті без встановлення виконуваного фалу;

– в системі повинно бути передбачено зв'язок через імейл-листування шаблонами звернень;

– інтерфейс системи повинен бути спрощеним і інтуїтивно зрозумілий, оскільки користувачі системи люди різного віку та різного рівня комп'ютерної грамотності;

– алгоритм розрахунку особистих показників користувача може бути використаний в десктопній версії системи;

– система повинна працювати в усіх сучасних браузерах;

– всі дані з системи повині зберігатися в резервних копіях. Періодичність встановлює адміністратор системи;

– тестування системи повинно проводитись згідно стандарту IEEE 829 Standart «forSoftrwareTEstDocumantation».

Основні принципи проектування з повторним використанням компонентів, що зображено на рис. 2.2:

– Abstraction Principle;

– Modularity Principle;

– Open-Closed Principle.

Програмні компоненти часто можна класифікувати за рівнями повторного використання, що зображено на рис. 2.2:

1. Прикладами базових компонентів є такі класи, як Гроші, Дата, Список, Календар, Облікові дані (логін, пароль) і Число. Вони можуть бути використані практично в будь-якому додатку і мають дуже низьке навантаження.

2. Приклади специфічних для архітектури компонентів включають в себе механізми повідомлення про події, компоненти для користувача інтерфейсу і системи передачі повідомлень.

3. Приклади доменних компонентів включають в себе такі класи, як Customer, Account і Transaction.

4. Приклади специфічних для додатка компонентів включають обробники повідомлень, обробники винятків і представлень.

5. Панелі інструментів у додатках.

6. Фреймворки, що надають загальний користувальницький інтерфейс, обробку помилок, механізм збереження і архітектуру.

7. Використання бібліотек класів.

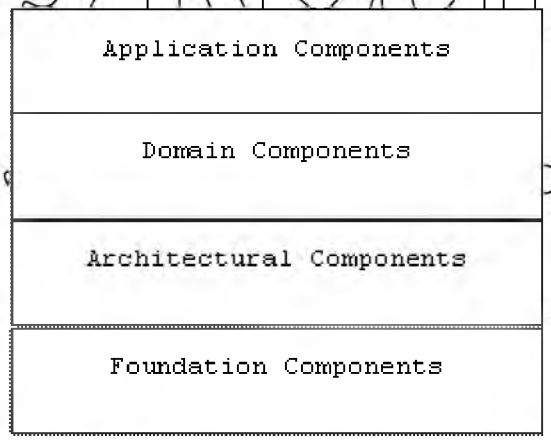

Рис. 2.2. Рівні повторного використання

Документація, пошук і простота використання є одними з важливих функцій для успішної стратегії повторного використання.

3 Проектування та реалізація компонентів системи

3.1 Моделювання поведінки системи

Розглянемо поведінку автоматизованої системи контролю доступу як набір функцій та акторів, що з ними взаємодіють. Модель поведінки системи для контролю доступу подано в нотації UML, діаграмою прецедентів, що описує доступні дії та випадки (рис. 3.1).

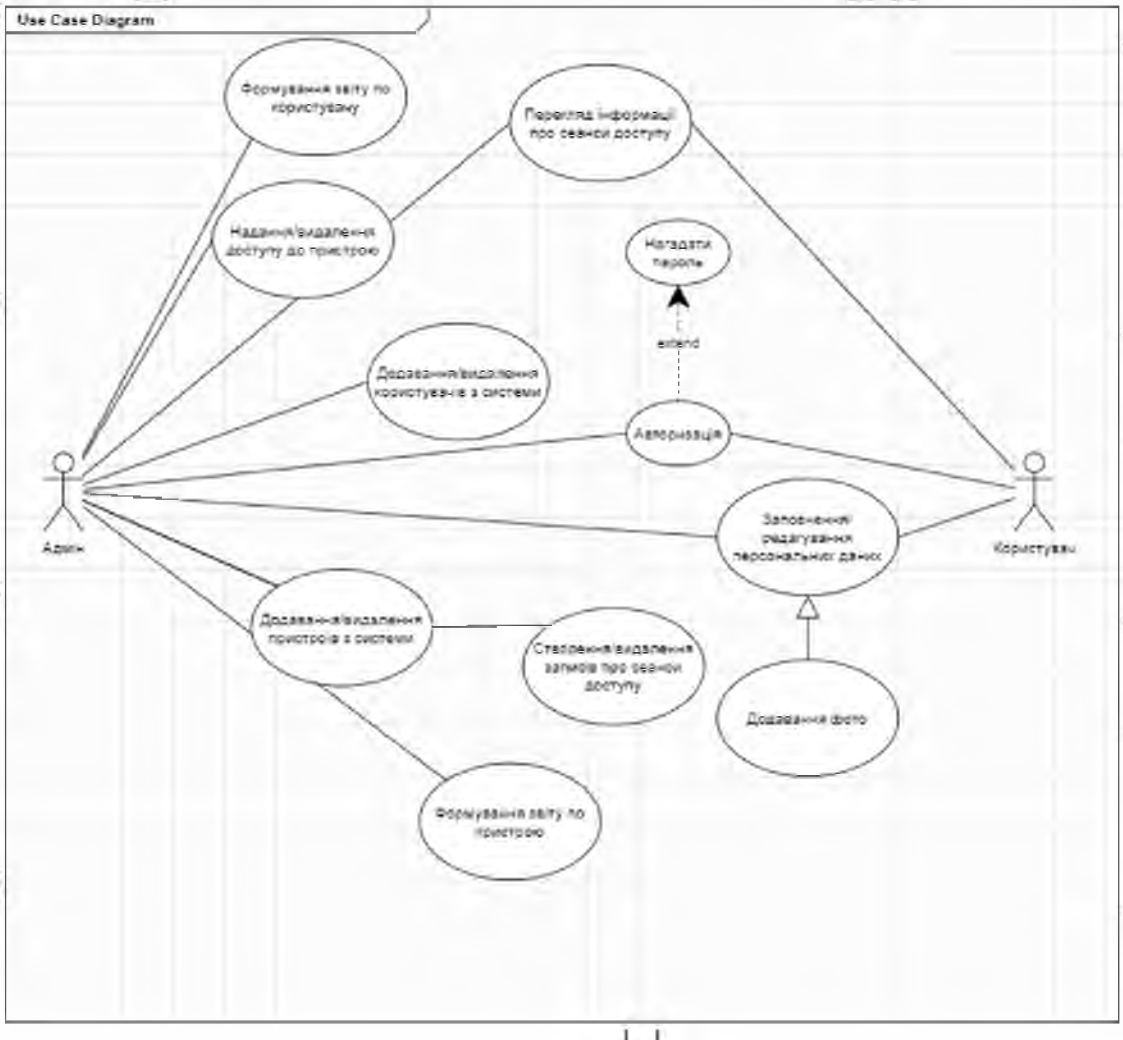

Рис. 3.1. Діаграма прецедентів автоматизованої системи контролю відвідувань

Для предметної області виділено наступних акторів, що зображено в табл. 3.1:

Таблиця 3.1

Визначені актори

Актор	Короткий опис
Адмін	Співробітник, який має можливість керувати усіма основними процесами системи, переглядати усі звіти
Користувач	Співробітник, який має можливість редагувати свої дані та переглядати інформацію про свої сеанси

Розглянемо тепер, які можливості має надавати наша система:

– актор Адмін використовує систему для редагування даних користувачів, створення та видалення користувачів та пристроїв, перегляду звітності про використання пристроїв;

– актор Користувач використовує систему для редагування своїх даних та перегляду звітності про використання пристроїв своєю картою.

На підставі вищевикладеного можна виділити наступні факти, що зображені в таблиці 3.2:

Таблиця 3.2

Опис прецедентів

Прецедент	Короткий опис
Формування звіту по користувачу	Запускається Адміном. Дозволяє отримати звіт про використання своєї картки певним користувачем.
Надання/видалення доступу до пристрою	Запускається Адміном. Дозволяє надати або видалити певному користувачу доступ до пристрою.
Перегляд інформації про сеанси доступу	Запускається Адміном та Користувачем. Дозволяю переглянути інформацію про отримання доступу користувачем.
Додавання/видалення користувачів з системи	Запускається Адміном. Дозволяю додати або видалити користувача з системи.
Авторизація	Запускається Адміном та Користувачем. Дозволяє провести авторизацію в системі

Продовження таблиці 3.2

Прецедент	Короткий опис
Нагадати пароль	Підсистема «Авторизація», яка запускається в разі необхідності відновити пароль користувача
Заповнення/Редагування персональних даних	Запускається Адміном, Користувачем. Дозволяє користувачам персоналізувати особистий кабінет.
Додавання/видалення пристроїв з системи	Запускається Адміном. Дозволяє додавати та видаляти пристрої з системи
Створення/видалення записів про сеанси доступу	Запускається Адміном. Дозволяє створювати та видаляти записи про отримання користувачами доступу до певних пристроїв.
Додавання фото	Підсистема «Заповнення/Редагування персональних даних», яка запускається в разі необхідності додати фото до особистого кабінету.
Формування звіту по пристрою	Запускається Адміном. Дозволяє отримати звіт про використання користувачами певного пристрою.

На діаграмах послідовності подано модель системних прецедентів акторів при взаємодії з системою для контролю відвідувань студентів у вищому навчальному закладі.

Діаграми послідовності є видом діаграм взаємодії мови UML, які описують відносини об'єктів в різних умовах. Умови взаємодії задаються сценарієм, отриманим на етапі розробки діаграм варіантів використання [4]. Існують різні погляди на застосування цього виду діаграм:

а) Фаулер пропонує будувати діаграми послідовності для візуалізації найбільш складних відносин на діаграмі КЛАСІВ [5];

б) Буч розглядає їх в якості альтернативи діаграм об'єктів і використовує для аналізу семантики сценаріїв на ранніх стадіях проектування (до створення протоколів окремих класів) [6];

в) Розенберг будує діаграми послідовності в рамках процесу ICONIX, тому вони будуються для кожного прецеденту (а не тільки для найцікавіших відносин, як у Фаулера). У процесі ICONIX розробці цього виду діаграм передує побудова діаграми робастності (придатності), тому вже виділені об'єкти, що беруть участь у прецеденті [7].

На рис. 3.2 зображено діаграму послідовності, яка описує процес авторизації користувача в системі.

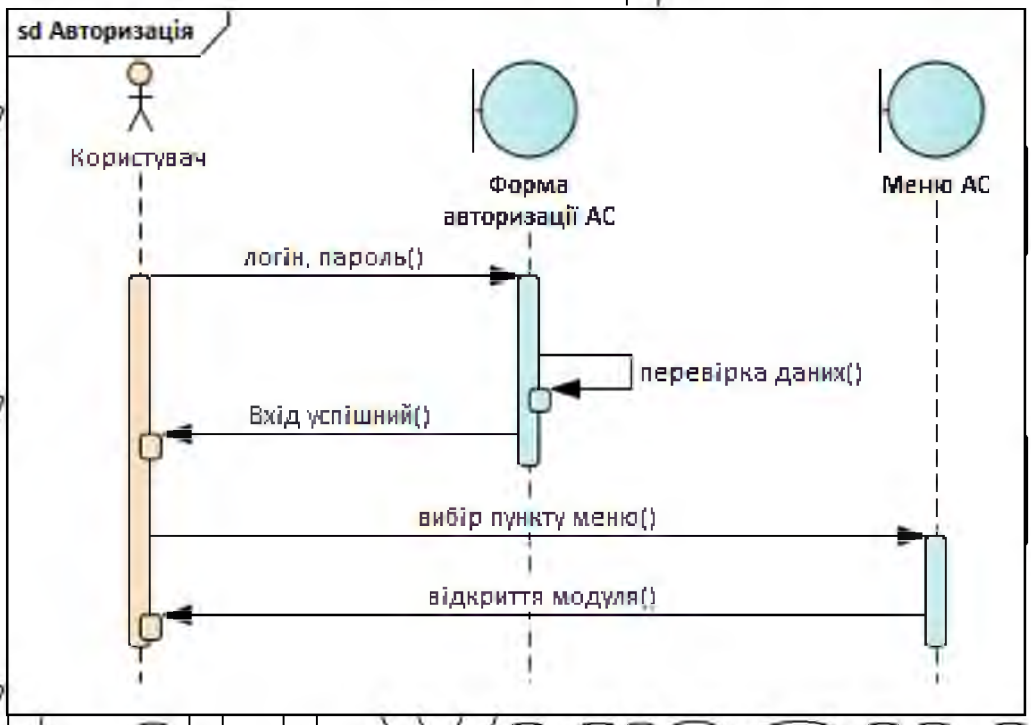

Рис. 3.2. Діаграма послідовності процесу «Авторизація»

На рис. 3.3 зображено діаграму послідовності, яка описує процес формування звітності по певним дисциплінам.

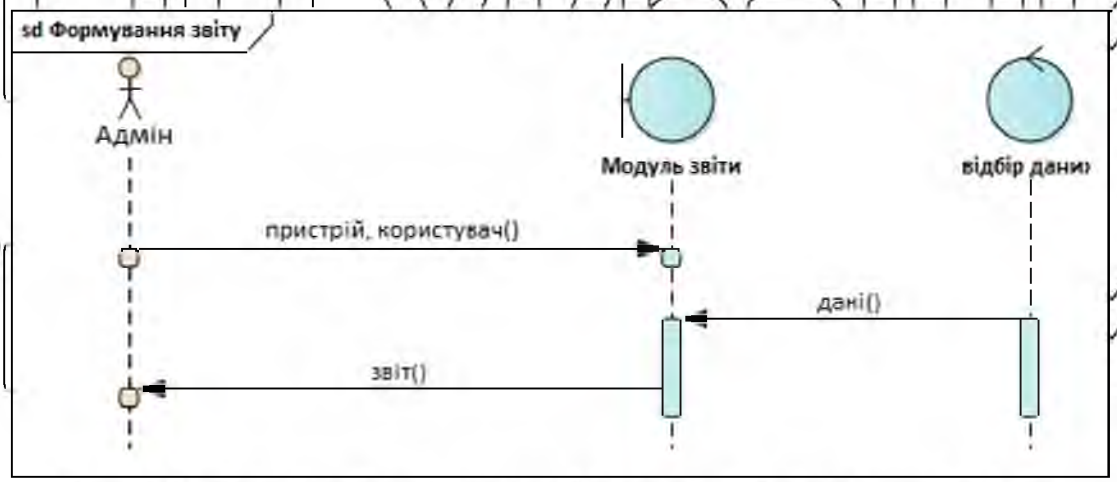

Рис. 3.3. Діаграма послідовності процесу «Формування звіту»

На рис. 3.4 зображено діаграму послідовності, яка описує процес заповнення особистого кабінету.

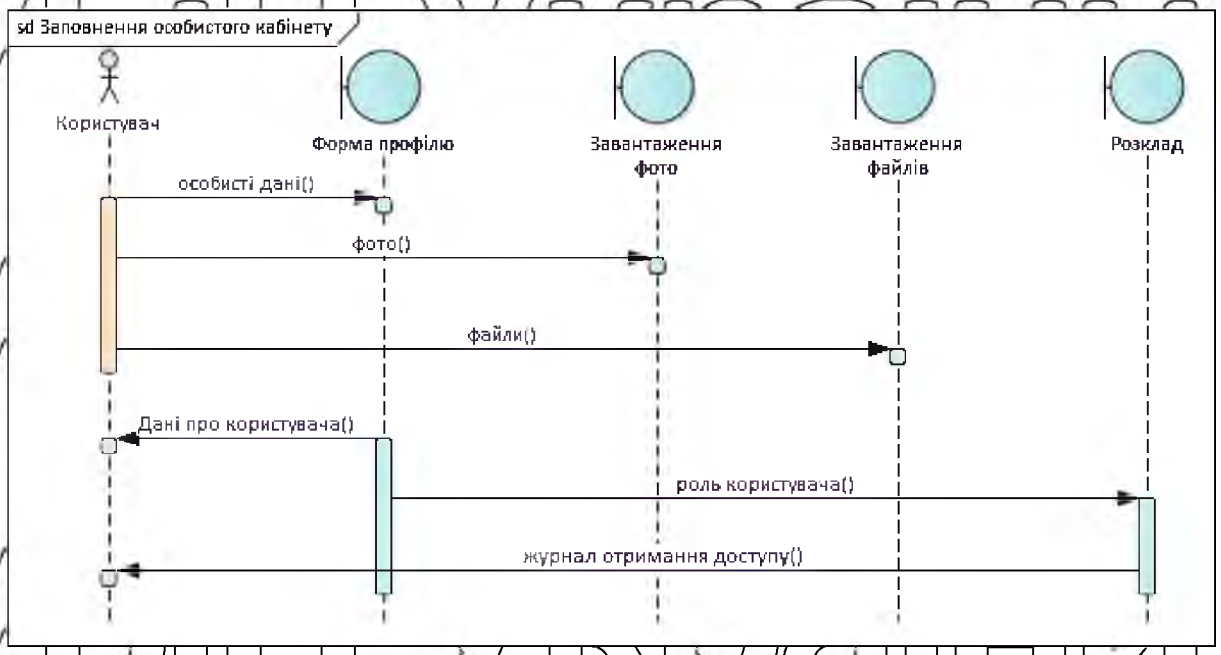

Рис 3.4. Діаграма послідовності процесу «Заповнення особистого кабінету»

3.2 Моделювання структури системи

SysML – це мова моделювання архітектури загального призначення для додатків System Engineering [8].

SysML підтримує специфікацію, аналіз, проектування, перевірку та перевірку широкого спектру систем та системних систем. Ці системи можуть включати апаратне, програмне забезпечення, інформацію, процеси, персонал та засоби.

SysML свого роду, це діалект UML 2.1 визначається як профіль UML 2. (Профіль UML – це діалект UML, який налаштовує мову за допомогою трьох механізмів: стереотипи, позначені значення та обмеження.)

SysML – це сприятлива технологія для моделювання системних інженерій (MBSE).

За допомогою діаграми внутрішніх блоків, в нотації SysML зображено складові автоматизованої контролю відвідувань студентів, та їх зв'язки, що між ними існують (рис. 3.5).

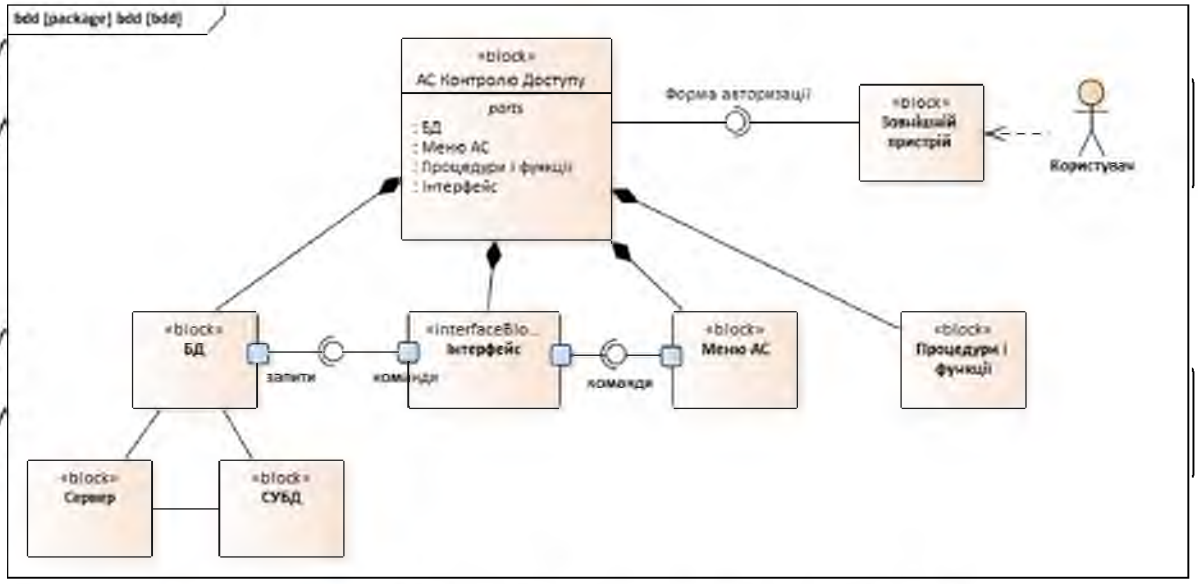

Рис. 3.5. Діаграма внутрішніх блоків системи

На діаграмі класів-сутностей (рис. 3.6) відображено основні класи-сутності, що відображають структуру системи контролю відвідувань і використовуються для моделювання даних і поведінки.

Схема даних системи контролю відвідування складається з трьох класів сутностей.

Користувачі:

- код користувача; тип даних – integer;
- ім'я користувача; тип даних – character;
- номер карти користувача,тип даних – character;
- пароль користувача, тип даних – character;
- дата створення; тип даних – timestamp;
- дата останнього редагування; тип даних – timestamp;
- створити користувача(), тип даних для повернення – користувач;
- редагувати користувача(), тип даних для повернення – void;
- видалити користувача, тип даних для повернення – void.

Пристрої:

- код пристрою; тип даних – integer;
- назва пристрою; тип даних – character;
- пароль від пристрою, тип даних – character;
- ключ доступу пристрою, тип даних – character;
- дата створення; тип даних – timestamp;
- дата останнього редагування; тип даних – timestamp;
- створити пристрій(), тип даних для повернення – пристрій;
- видалити пристрій(), тип даних для повернення – void;

Логи отримання доступу:

- код користувача; тип даних – integer;
- код пристрою, тип даних – integer;
- дата створення; тип даних – timestamp;
- дата останнього редагування; тип даних – timestamp;
- створити запис(), тип даних для повернення – запис;
- видалити запис(), тип даних для повернення – void;

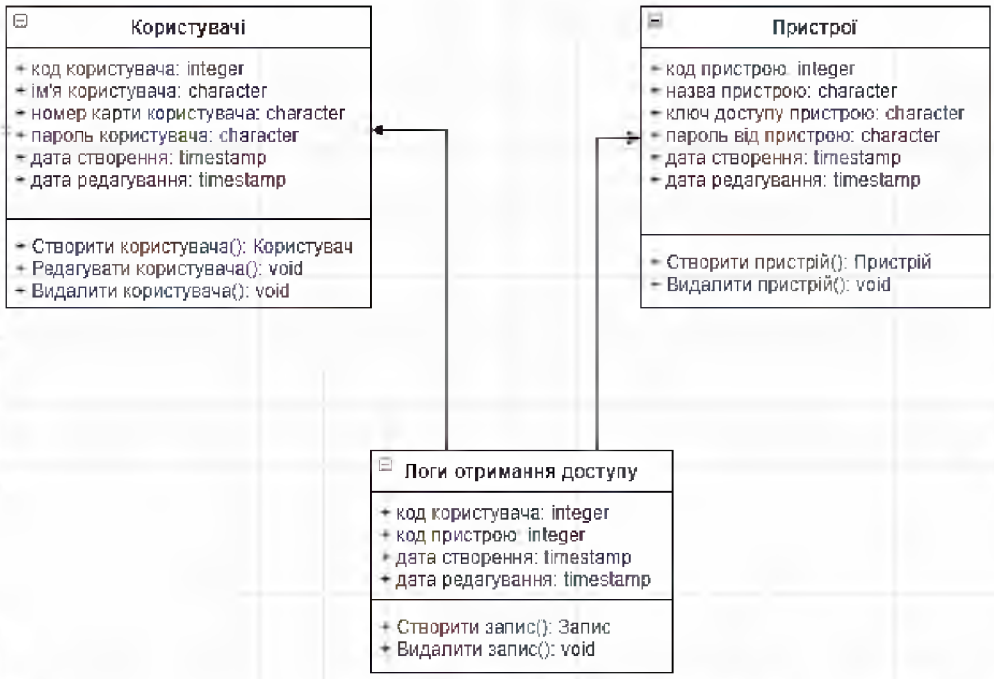

Рис. 3.6. Діаграма класів-сутностей системи контролю доступу до приміщень

3.3 Розробка структури програмних модулів

Для створення програмного додатку буде використовуватися:

- PHP середовище;
- Docker;
- База даних (MySQL);
- PHPUnit;
- NodeJS.

При створенні нового проекту Laravel процес установки автоматично створює файл .env (скопійований з файлу .env.приклад) для конфігурації та облікових даних. Після створення потрібно налаштувати наступні змінні.

Лістинг 3.1 – Налаштування середовища

```
DB_CONNECTION=mysql
2DB_HOST=127.0.0.1
3DB_PORT=3306
4DB_DATABASE=laravel
5DB_USERNAME=root
6DB_PASSWORD=
```

Щоб створити уявлення, що відображає список посилань, нам необхідно оновити основний маршрут проекту, а також визначити новий маршрут, на якому буде відображатися наша форма відправки. Ми можемо додати нові маршрути в наш додаток в web.php файл.

Лістинг 3.2 – Налаштування посилань

```
Route::get('/', function () {
    return view('welcome');
});
```

Для створення нового маршруту ми можемо використовувати або замикання маршруту, або виділений клас контролера. Ми будемо використовувати замикання для наших маршрутів відправки та індексування.

Контролер створює зв'язок між вашими моделями і вашими уявленнями. Коли користувач надсилає нову форму запису в блозі, дані надходять до контролера, де вони обробляються, а потім передаються в модель для зберігання в базі даних, потім контролер надсилає зворотний зв'язок назад до подання, повідомляючи, що запис блогу був створений.

Контролери будуть згенеровані за допомогою спеціальної команди і будуть слідувати Угоді про формулювання єдиного регістра заголовка зі словом Controller в кінці. Наш контролер формування звітів ми будемо називати ReportController.php

Лістинг 3.3 – Команда створення контролеру

```
php artisan make:controller ReportController
```

Контролер має 7 методів, які дозволяють виконувати операції crud:

- index() – щоб отримати всі ресурси, наприклад, всі доступні записи.
- show() – щоб отримати один ресурс, наприклад, один звіт.
- create() – показує форму, використовувану для створення ресурсу (недоступна для контролерів API).
- store() – щоб зафіксувати ресурс у базі даних.
- update() – щоб зафіксувати відредагований ресурс у базі даних.
- destroy() – щоб видалити ресурс з бази даних.

Для роботи з базою даних потрібно створити таблиці. Таблиці створюються за допомогою файлів міграції. Для запуску файлу міграції та створення таблиці потрібно виконати спеціальну команду.

Лістинг 3.4 – Запуск міграції

```
php artisan migrate
```

Нова міграція буде поміщена в папці database/migrations. Кожне ім'я файлу міграції містить тимчасову мітку, яка дозволяє Laravel визначати порядок міграції.

Клас міграції містить два методи: " up "і " down ". Метод up використовується для додавання нових таблиць, стовпців або індексів у вашу базу даних, тоді як метод "down" повинен реверсувати операції, що виконуються методом "up".

В обох цих методах ви можете використовувати конструктор схем Laravel для виразного створення і зміни таблиць. Щоб дізнатися про всі методи, доступних в конструкторі " Schema ", ознайомтеся з його документацією. Наприклад, наступна міграція створює таблицю " Users ":

Лістинг 3.5 – Створення таблиці Users

```
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('card_number')->nullable()->
>default(null);
            $table->string('password');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
```

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

Після того, як ми налаштували контролери та базу даних, потрібно створити веб сторінки, через які буде відбуватися взаємодія користувача з нашими методами. Всередині папки ресурсів створімо файл, який буде відповідати за нашу веб сторінку.

Нижче наведено остаточний код того, як виглядатиме файл:

Лістинг 3.6 – Веб-сторінка

```
@extends('index')

@section('content')
<div class="container">
<div class="row">
<div class="offset-col-3 col-6">
<form action="/users/store" method="POST" class="form">
@csrf

<div class="form-group">
<label for="name">User name</label>
<input type="text" id="name" name="name">
</div>

<div class="form-group">
<label for="card_number">Card number</label>
<input type="text" id="card_number" name="card_number">
</div>

<div class="form-group">
<label for="password">Password</label>
<input type="text" id="password" maxlength="4" minlength="4"
name="password">
</div>

<div class="form-group">
<label for="device">Device</label>
<select id="device_id" name="device_id">
@foreach($devices as $device)
<option value="{{$device->id}}">{{$device->name}}</option>
@endforeach
</select>
</div>
```

```
<button type="submit">Create</button>
</form>
</div>
</div>
</div>
@endsection
```

3.4 Розгортання та налаштування системи

Платформа Laravel має кілька системних вимог. Ви повинні переконатися, що ваш веб-сервер має наступну мінімальну версію PHP і розширення:

- PHP >= 7.3;
- BCMath PHP Extension;
- Ctype PHP Extension;
- Fileinfo PHP Extension;
- JSON PHP Extension;
- Mbstring PHP Extension;
- OpenSSL PHP Extension;
- PDO PHP Extension;
- Tokenizer PHP Extension;
- XML PHP Extension.

Я розгортаю свою програму на сервері під керуванням Nginx, тому використовую наступний файл конфігурації як відправну точку для налаштування веб-сервера. Якщо вам потрібна допомога в управлінні вашим сервером, розгляньте можливість використання служби управління та розгортання Laravel server від першої особи, такої як Laravel Forge [9].

Для коректної роботи серверу потрібно правильно налаштувати файл.

Лістинг 3.7 – Файл налаштувань серверу

```
server {
    listen 80;
    server_name example.com;
    root /srv/example.com/public;
```

```
        add_header X-Frame-Options "SAMEORIGIN";
        add_header X-Content-Type-Options "nosniff";

        index index.php;

        charset utf-8;

        location / {
            try_files $uri $uri/ /index.php?$query_string;
        }
        location = /favicon.ico { access_log off; log_not_found
off; }
        location = /robots.txt  { access_log off; log_not_found
off; }

        error_page 404 /index.php;

        location ~ \.php$ {
            fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
            fastcgi_param                       SCRIPT_FILENAME
$realpath_root$fastcgi_script_name;
            include fastcgi_params;
        }

        location ~ /\.(?!well-known).* {
            deny all;
        }
    }
```

Є кілька провайдерів VPS, таких як DigitalOcean, Vultr, Linode та Hetzner. Хоча робота з некерованим VPS більш-менш однакова у різних провайдерів, вони не надають однакові послуги.

Наприклад, DigitalOcean надає послуги з управління базами даних. З іншого боку, у Linode і Vultr таких сервісів немає.

Перед підготовкою нового сервера потрібно згенерувати SSH-ключі.

Після завершення базових налаштувань наступне – це розгортання коду на сервері. Я бачив, як люди клонували репозиторій десь на робочому сервері і входили на сервер, щоб виконувати вилучення кожного разу, коли в коді з'являються якісь нові зміни.

Є набагато кращий спосіб зробити це. Замість входу на сервер для виконання вилучення, ви можете використовувати сам сервер в якості сховища і відправляти код безпосередньо на сервер. Ви також можете автоматизувати етапи після розгортання, такі як встановлення залежностей, запуск міграції тощо, що зробить розгортання Laravel на сервері простою дією. Але перш ніж робити все це, вам спочатку потрібно встановити PHP і Composer на сервер [10].

Додаток тепер майже готове приймати запити з усього світу. Останній крок, який я хотів би зробити – це кешування конфігурації, переглядів і маршрутів Laravel для підвищення продуктивності.

За допомогою класу валідації відбувається перевірка персонального ключу користувача та правильність введених даних за допомогою спеціальних правил, що вказуються у відповідній функції.

Laravel Sail – це інструмент командного рядка для взаємодії з середовищем розробки Docker. Sail забезпечує відмінну відправну точку для створення Програми Laravel з використанням PHP, MySQL і Redis. Досвід роботи з Docker не потрібно. По суті, Sail – це файл docker-compose.yml, який зберігається в корені вашого проекту і набір скриптів sail, за допомогою яких можна управляти Docker-контейнерами, визначеними в docker-compose.yml. Laravel Sail підтримується в macOS, Linux і Windows (через WSL2). Laravel Sail автоматично встановлюється з усіма новими додатками Laravel, тому є можливість відразу ж почати його використовувати.

Файл docker-compose.yml Laravel Sail визначає різні контейнери Docker, які працюють разом, щоб допомогти Вам створювати додатки Laravel. Щоб дізнатися, що це за контейнери – зверніться до запису вашого файлу docker-compose.yml. Контейнер laravel.test – це основний контейнер, який буде обслуговувати ваш додаток.

Перед запуском Sail переконайтеся, що на вашому локальному комп'ютері не працюють інші веб-сервери або бази даних. Щоб запустити всі контейнери Docker, визначені у файлі docker-compose.yml вашої програми, ви повинні виконати команду up:

Лістинг 3.11 – Команда для запуску контейнеру з додатком

```
sail up
```

Щоб зупинити всі контейнери, ви можете просто натиснути Control + C, щоб зупинити виконання контейнера. Якщо контейнери працюють у фоновому режимі, ви можете використовувати команду stop:

Лістинг 3.12 – Команда для зупинки контейнеру з додатком

```
sail stop
```

3.5 Тестування працездатності системи

З коробки Laravel підтримує 3 типи тестів:

- Browser;
- Feature;
- Unit.

У більшості випадків треба писати окремий модульний тест для кожного випадку, визначеного у запиті форми тестування. Це призводить до великої кількості тестів, таких як test_request_without_title і test_request_without_content. Всі ці методи реалізуються точно так же, тільки викликаючи вашу кінцеву точку з деякими іншими даними. Це призведе до великої кількості дублікатів коду.

Тестування системи повинно включати контроль повноти, коректності і несуперечливості технічної документації.

3.5.1 Об'єкти експертного аналізу коректності документації

Об'єкти експертного аналізу коректності документації:

- технічне завдання;
- документація технічного проекту;

– робоча документація: керівництво користувача, керівництво адміністратора.

3.5.2 Порядок проведення аналізу коректності

Порядок проведення експертного аналізу коректності документації:

– для проведення експертного аналізу коректності потрібно отримати технічну документацію, а також інші дані, необхідні для проведення експертного аналізу коректності документації (функціональні вимоги користувачів, документацію аналогічних реалізованих проектів і т. д.);

– виконавець організовує проведення експертного аналізу коректності документації об'єкта експертного аналізу:

– перевірка документації на відповідність назв діалогових вікон, команд меню, кнопок панелей інструментів;

– перевірка коректності описуваних в документації дій користувача, тобто того, що описувані дії приведуть до бажаного результату;

– несуперечливість окремих вимог один одному і положенням інших документів, наданих на експертизу, відповідність необхідних послуг поставленим цілям і завданням;

– повнота і необхідний рівень деталізації;

– виконавець оформляє експертний висновок за результатами експертного аналізу коректності документації;

– форма експертного висновку розробляється виконавцем і повинна бути узгоджена з Замовником до моменту передачі першого висновку.

Laravel побудований з урахуванням тестування. Фактично, підтримка PHPUnit доступна за замовчуванням, а файл phpunit.xml вже налаштований для вашої програми. Також фреймворк містить зручні методи для повноцінного тестування ваших додатків. Папка tests містить файл з прикладом тесту ExampleTest.php. Після установки нової програми Laravel просто виконайте команду phpunit для запуску ваших тестів.

Для створення тесту використовується Artisan-команда:

Лістинг 3.13 – Команда створення тесту

```
php artisan make:test DeviceTest
```

Згенерований файл класу буде розміщена в App/HTTP і запити. Після чого треба оголосити набір правил перевірки для цієї форми запиту.

Функціональне тестування повинно включати в себе:

- розробку методики тестування;
- побудова інфраструктури тестування;
- розробку тестових сценаріїв;
- проведення тестів згідно затвердженої методики;
- аналіз результатів тестів та оформлення фінального звіту;
- презентація результатів проекту.

3.5.3 Розробка методики тестування

Методика повинна описувати підхід до проведення тестування, а також інструментарій, який буде використовуватися для супроводу процесу тестування.

У процесі розробки методики на підставі наданої замовником технічної документації (керівництво користувача, Керівництво Адміністратора) повинен бути деталізований перелік перевіряються бізнес процесів, що реалізуються ІС, проведена оцінка ризиків, і встановлені пріоритети тестових сценаріїв.

3.5.4 Побудова інфраструктури тестування

Повинна бути проведена установка і настройка системи підтримки тестування під процеси замовника.

3.5.5 Розробка тестових сценаріїв

Повинна бути виконана розробка тестових вимог, тестових сценаріїв і створення впорядкованих особливим чином для найбільш ефективного тестування наборів сценаріїв.

Вимоги до тестових сценаріїв:

– тестовий сценарій повинен бути оформлений в структурованому вигляді з описом полів і структури і внесений в систему підтримки тестування;

– кожен тестовий сценарій повинен містити послідовність кроків, вхідні дані, очікувані результати, критерії для перевірки успішності;

– тестові дані повинні бути оформлені в структурованому вигляді з описом полів і структури.

3.5.6 Проведення тестів згідно затвердженої методики

Повинні бути виконані наступні операції:

– підготовка тестових даних;

– виконання тестових сценаріїв

– контроль коректності виконання тестів;

– протоколювання виявлених дефектів і проблем;

– збір і попередня обробка отриманих даних.

Безпосередньо після завершення кожного з тестових сценаріїв, інформація про всі виявлені помилки і проблеми повинна вноситься в БД ведення дефектів. У тому випадку, якщо виявлені дефекти не носять критичний характер і не перешкоджають проведенню тестових робіт (не спотворюють результатів роботи системи), тестування триває. В іншому випадку, при виявленні блокуючих дефектів, складається і підписується представниками Виконавця і Замовника протокол про припинення випробувань, в якому відображений список критичних помилок і час відновлення випробування.

По завершенні кожної ітерації функціонального тестування повинен бути сформований звіт по кожній ітерації.

3.5.7 Аналіз результатів тестів та оформлення фінального звіту

Після закінчення тестування повинен бути проведений комплексний аналіз результатів досліджень. Замовнику повинен бути наданий звіт про проведення

функціонального тестування. Звіт про проведення функціонального тестування повинен містити:

– кількість виконаних тестових сценаріїв;
– кількість знайдених дефектів і їх критичність;
– опис критичних дефектів;
– термін проведення тестування;
– підсумковий статус про готовність системи.

Тестування безпеки – окремий напрямок тестування, який вимагає від спеціаліста фундаментальних знань технічного характеру та хорошої профільної кваліфікації. Я відзначу низку загальних моментів, які можуть допомогти будь-якому тестувальнику знаходити класичні вразливості, не допускаючи їх вихід на продакшен. Питання безпеки програм регламентуються OWASP Guide, CHECK, ISACA, NIST Guideline, OSSTMM.

Існує ряд принципів безпеки, до яких належать конфіденційність, цілісність та доступність:

1. Конфіденційність – обмеження доступу до тієї чи іншої інформації для певної категорії користувачів (або, навпаки, надання доступу лише обмеженій категорії).
2. Цілісність включає:
   a. можливість відновити дані в повному обсязі при їх пошкодженні;
   b. доступ на зміну інформації лише певної категорії користувачів.
3. Доступність – ієрархія рівнів доступу та чітке їх дотримання.

Перерахуємо класичні вразливості сучасних веб-додатків:

– XSS - генерація на сторінці продукту скриптів, що становлять небезпеку для користувачів продукту;
– XSRF - вразливість, при якій користувач переходить з довіреної сторінки на шкідливу, де крадуть які представляють цінність дані користувача;
– code injection (PHP, SQL) – ін'єкція частини виконавчого коду, яка уможливлює отримання несанкціонованого доступу до програмного коду або бази даних та вносити до них зміни;

– authorization bypass – це вид уразливості, за якого можна отримати несанкціонований доступ до облікового запису або документів іншого користувача;

– переповнення буфера – явище, якого можна досягти у шкідливих цілях, за своєю суттю є використання місця для запису даних далеко за межами виділеного буфера пам'яті.

ВИСНОВКИ

У результаті виконання магістерської роботи було спроектовано та розроблено програмну частину автоматизованої системи контролю доступу.

У проекті вирішено наступні задачі:

1. Проведено аналіз та опис предметної області програмного продукту. Проаналізовано усі існуючі подібні системи та виявлено сильні та слабкі сторони. Описано основні функції програмного продукту.

2. Виділено основні бізнес вимоги, функціональні вимоги та нефункціональні вимоги до системи. Підібрано мову програмування, яка забезпечить високу якість системи.

3. Спроектовано систему, розроблено ієрархію проекту та проведено опис діаграм. Для проектування систему було використано діаграми послідовності, прецедентів, внутрішніх блоків, класів-сутностей, діаграми класів керування системи та діаграми граничних класів системи.

4. Описано кроки реалізації, наведено код системи з коментарями до нього, описано основні архітектурні рішення при виростанні фреймворку Laravel для beck-end розробки.

5. Проведено Unit тестування системи. Тестування системи показує, що вона відповідає вимогам, які були створені для неї.

Аналіз системи показав, що створений проект є конкурентоспроможним та затьмарює усіх конкурентів на ринку.

Спроектована система автоматизує або взагалі видаляє потрібність керівництва витрачати додаткові кошти на використання непотрібних способів слідкування за персоналом. Крім цього, у розробку покладено модель, за якою власники системи зможуть її покращити та розширити функціонал.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ZkTeco. URL: https://zkteco.com.ua (дата звернення: 20.05.2021).
2. ControlGate. URL: https://controlgate.ru (дата звернення: 20.05.2021).
3. HID Global. URL: https://www.hidglobal.com/sites/default/files/dtk/plt-04900-b.1-hid-signo-biometric-reader-25b-user-guide.pdf (дата звернення: 20.05.2021).
4. Основы UML – диаграммы использования (use-case). URL: https://pro-prof.com/archives/2594 (дата звернення: 20.05.2021).
5. Фаулер М., Скотт К UML. Основы СПб.: Символ, 2006, 184 с.
6. Буч Гради Объектно-ориентированный анализ и проектирование с примерами приложений, 3-е изд. / Буч Гради, Максимчук Роберт А., Энгл Майкл У., Янг Бобби Дж., Коналлен Джим, Хьюстон Келли А.: Пер с англ. – М.: ООО “И.Д. Вильямс”, 2010. – 720 с.
7. Розенберг Д., Скотт К. Применение объектного моделирования с использованием UML и анализ прецедентов.: Пер. с англ. М.: ДМК Пресс, 2002
8. SysML URL: URL: https://ru.wikipedia.org/wiki/XAMPP. (дата звернення: 21.05.2021). (дата звернення: 21.05.2021).
9. NGINX. URL: https://ru.wikipedia.org/wiki/NGINX. (дата звернення: 21.05.2021).
10. Composer. URL: https://ru.wikipedia.org/wiki/Composer. (дата звернення: 19.05.2021).

Компонент «ApiController»

```
<?php

namespace App\Http\Controllers;

use App\Http\Requests\OpenRequest;
use App\Http\Requests\RegisterRequest;
use App\Models\DeviceUseLog;
use App\Models\User;
use Carbon\Carbon;

class ApiController extends Controller
{
    public function getUsers()
    {
        return User::all();
    }

    public function openDevice(OpenRequest $request)
    {
        $user = User::where('card_number', $request->card_number)->orWhere('password', $request->password)->firstOr(function () {
            return response()->json([
                'success' => false,
                'message' => 'No such user'
            ]);
        });

        if ($user->isAbleToOpen()){
            DeviceUseLog::create([
                'user_id' => $user->id,
                'device_id' => $user->device_id
            ]);

            return response()->json([
                'success' => true
            ]);
        }

        return response()->json([
            'success' => false,
            'message' => 'You have already used your card'
        ]);
    }

    public function registerCardForNewUser(RegisterRequest $request)
    {
        $user = User::where('created_at', '>', Carbon::now()->subMinutes(10))
```

```
            ->whereNull('card_number')
            ->latest()
            ->firstOr(function () {
                return response()->json([
                    'success' => false,
                    'message' => 'No suitable users'
                ]);
            });

        $user->update([
            'card_number' => $request->card_number
        ]);

        return response()->json([
            'success' => true,
            'message' => 'Card saved to user'
        ]);
    }
}
```

```
<?php

namespace App\Exports;

use App\Models\DeviceUseLog;
use Illuminate\Support\Collection;
use Maatwebsite\Excel\Concerns\FromCollection;
use Maatwebsite\Excel\Concerns\WithColumnFormatting;
use Maatwebsite\Excel\Concerns\WithHeadings;
use Maatwebsite\Excel\Concerns\WithMapping;

class HistoryExport implements FromCollection, WithMapping,
WithHeadings
{
    protected $user_id;

    public function __construct($user_id = null)
    {
    $this->user_id = $user_id;
    }

    /**
     * @return Collection
     */
    public function collection(): Collection
    {
        if ($this->user_id) {
            return DeviceUseLog::where('user_id', $this-
>user_id)->get();
        }
```

```
        return DeviceUseLog::all();
    }

    public function map($tick): array
    {
        return [
            $tick->user->name,
            $tick->device->name,
            $tick->created_at
        ];
    }

    public function headings(): array
    {
        return [
            'User name',
            'Device name',
            'Date'
        ];
    }
}
```

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Device extends Model
{
    use HasFactory;

    protected $fillable = [
        'name',
        'password',
        'token'
    ];

    public function users()
    {
        return $this->hasMany(User::class);
    }

    public function logs()
    {
        return $this->hasMany(DeviceUseLog::class);
    }
}
```

```
<?php
```

```
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class DeviceUseLog extends Model
{
    use HasFactory;

    protected $fillable = [
        'user_id',
        'device_id'
    ];

    public function user()
    {
        return $this->hasOne(User::class, 'id', 'user_id');
    }

    public function device()
    {
        return $this->hasOne(Device::class, 'id',
'device_id');
    }
}
<?php

namespace App\Models;

use Carbon\Carbon;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
    use HasFactory, Notifiable;

    protected $fillable = [
        'name',
        'card_number',
        'password',
        'role',
        'device_id'
    ];

    protected $hidden = [
        'password'
    ];

    public function device():
\Illuminate\Database\Eloquent\Relations\HasOne
```

```
    {
        return $this->hasOne(Device::class, 'id',
'device_id');
    }

    public function logs():
\Illuminate\Database\Eloquent\Relations\HasMany
    {
        return $this->hasMany(DeviceUseLog::class, 'user_id',
'id');
    }

    public function user_devices():
\Illuminate\Database\Eloquent\Relations\HasMany
    {
        return $this->hasMany(UserDevices::class, 'user_id',
'id');
    }

    public function isAdmin(): bool
    {
        return $this->role == 'admin';
    }
}
```

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class UserDevices extends Model
{
    use HasFactory;

    public $timestamps = false;

    protected $fillable = [
        'user_id',
        'device_id'
    ];

    public function user()
    {
        return $this->hasOne(User::class, 'id', 'user_id');
    }

    public function device()
    {
        return $this->hasOne(Device::class, 'id',
'device_id');
```

НУБіП України

НУБіП України

НУБіП України

НУБіП України

НУБіП України

НУБіП України

НУБіП України

Додаток А.2

Компонент «User model»

```
<?php

namespace App\Models;

use Carbon\Carbon;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
    use HasFactory, Notifiable;

    protected $fillable = [
        'name',
        'card_number',
        'password',
        'device_id'
    ];

    public function device()
    {
        return $this->hasOne(Device::class, 'id',
'device_id');
    }

    public function logs()
    {
        return $this->hasMany(DeviceUseLog::class, 'user_id',
'id');
    }

    public function getLastEntry()
    {
        return $this->logs()->latest()->first();
    }

    public function isAbleToOpen()
    {
        return $this->getLastEntry()->created_at <
Carbon::now()->subMinutes(2);
    }
}
```

```
<?php

namespace App\Exceptions;
```

```
use Illuminate\Foundation\Exceptions\Handler as ExceptionHandler;
use Throwable;

class Handler extends ExceptionHandler
{
    /**
     * A list of the exception types that are not reported.
     *
     * @var array
     */
    protected $dontReport = [
        //
    ];

    /**
     * A list of the inputs that are never flashed for validation exceptions.
     *
     * @var array
     */
    protected $dontFlash = [
        'current_password',
        'password',
        'password_confirmation',
    ];

    /**
     * Register the exception handling callbacks for the application.
     *
     * @return void
     */
    public function register()
    {
        $this->reportable(function (Throwable $e) {
            //
        });
    }
}
```

```
<?php

namespace App\Http\Middleware;

use App\Models\Device;
use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;

class CheckApiToken
{
    /**
```

```
     * Handle an incoming request.
     *
     * @param \Illuminate\Http\Request $request
     * @param \Closure $next
     * @return mixed
     */
    public function handle(Request $request, Closure $next)
    {
        Log::alert($request->token);
        if ($request->token) {
            if (Device::where('token', $request->token)->first()) {
                return $next($request);
            }
        }

        return response()->json([
            'message' => 'Please provide valid token',
        ], 401);
    }
}
```

```
<?php

namespace App\Http\Requests\Auth;

use Illuminate\Auth\Events\Lockout;
use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\RateLimiter;
use Illuminate\Support\Str;
use Illuminate\Validation\ValidationException;

class LoginRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
```

```
            return [
                'name' => ['required', 'string'],
                'password' => ['required', 'string'],
            ];
        }

        /**
         * Attempt to authenticate the request's credentials.
         *
         * @return void
         *
         * @throws \Illuminate\Validation\ValidationException
         */
        public function authenticate()
        {
            $this->ensureIsNotRateLimited();

            if (! Auth::attempt($this->only('name', 'password'),
$this->boolean('remember'))) {
                RateLimiter::hit($this->throttleKey());

                throw ValidationException::withMessages([
                    'name' => __('auth.failed'),
                ]);
            }

            RateLimiter::clear($this->throttleKey());
        }

        /**
         * Ensure the login request is not rate limited.
         *
         * @return void
         *
         * @throws \Illuminate\Validation\ValidationException
         */
        public function ensureIsNotRateLimited()
        {
            if (! RateLimiter::tooManyAttempts($this-
>throttleKey(), 5)) {
                return;
            }

            event(new Lockout($this));

            $seconds = RateLimiter::availableIn($this-
>throttleKey());

            throw ValidationException::withMessages([
                'name' => trans('auth.throttle', [
                    'seconds' => $seconds,
                    'minutes' => ceil($seconds / 60),
                ]),
```

```
        ]);
    }

    /**
     * Get the rate limiting throttle key for the request.
     *
     * @return string
     */
    public function throttleKey()
    {
        return Str::lower($this->input('email')).'|'.$this->ip();
    }
}
```