МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук, фізики та математики

Кафедра інформатики, програмної інженерії та економічної кібернетики

Проєктування та розробка сервісної архітектури управління бізнес-процесами університету. Безпека системи

Кваліфікаційна робота(проєкт)

на здобуття ступеня вищої освіти «магістр»

Виконав: здобувач 2 курсу , 231М групи

Спеціальності 122 Комп'ютерні науки

Освітньо-професійної програми

«Комп'ютерні науки» другого

(магістерського) рівня вищої освіти

Гетьман Владислав Вкторович

Керівник: кандидат фізико-математичних наук,

доктор педагогічних наук, професор

Співаковський Олександр Володимирович

Рецензент: кандидат педагогічних наук, доцент

Гончаренко Тетяна Леонідівна

Херсон – 2021

Зміст

Список умовних позначень, символів, скорочень та термінів

CSS (Cascading Style Sheets) – каскадні таблиці стилів

CVSS (Common Vulnerability Scoring System) – загальна система оцінки вразливостей

FTP (File Transfer Protocol) – протокол передачі файлів по мережі

HTML (Hypertext Markup Language) – мова розмітки гіпертекстових документів

HTTP (Hypertext Transfer Protocol) – протокол передачі даних

ID (Identifier) – унікальна ознака об'єкта

IP (Internet Protocol) – інтернет протокол

JSON – Javascript object notation

JWT – JSON Web Token

OS (operating system) – операційна система

OWASP (Open Web Application Security Project) – це відкритий проект забезпечення безпеки веб-додатків

PTES (Penetration Testing Execution Standard) – стандарт виконання тестування на проникнення

SQL (Structured Query Language) – мова структурованих запитів

SSL (Secure Sockets Layer) – рівень захищених сокетів

TLS (Transport Layer Security) – протокол захисту транспортного рівня

URL (Uniform Resource Locator) – уніфікований локатор ресурсів

XSS (Cross – site Scripting) – міжсайтове виконання сценаріїв

ЗВО – заклад вищої освіти

ДСТУ – державний стандарт України

НД ТЗІ – нормативний документ системи технічного захисту інформації

ОС – операційна система

ПЗ – програмне забезпечення

СКБД – система керування базами даних

ВСТУП

Актуальність: Для держави, сьогодні одним з ключових векторів розвитку в сфері освітньої політики безумовно є впровадження передових досягнень ІТ індустрії у ЗВО. Одним із таких векторів є поліпшення інформаційного освітнього простору. Завдяки командній роботі технологічних лідерів цифрової індустрії навчальні заклади можуть впроваджувати останні інформаційні досягнення. Це дає змогу не лише підвищити якість освітнього процесу, а й дати студентам досвід, який згодом знадобиться в їх професійній діяльності. Впровадження передових іновацій в осітній процес навчальних закладів безумовно є одним із передових напрямків роботи уряду. Ця ініціатива черпає ресурс не тільки з комерційних рішень, а й з відкритого програмного забеспечення. Розробляється комплек рішення зі створення критеріїв оцінювання якості програмних засобів загальноосвітнього призначення для потреб Міністерства освіти і науки України. Моніторинг загально-доступних ресурсів України і світу, резюмує наступне: більшість ЗВО мають і підтримують власні інформаційні ресурси; спостерігається значна кількість сайтів з застарілим функціоналом або неактуальною інформацію. Отже, існує нагальна потреба створення на загальнодержавному рівні єдиного стандарту створення веб-сайту ЗВО з конкретним переліком логічних одиниць(наприклад, структура навчальних підрозділів як групи, факультети, спеціальності; функціональних єлементів для цільовох аудиторії користувачів як профайл студента, факультативні заняття, електронна бібліотека, розклад занять тощо). Це дасть змогу описати стандартні методи ведення електронних порталів та прискорить інтеграцію різнорівневих систем навчального закладу в єдиний ресурс.

Об’єкт дослідження: методи захисту систем управління університетами

Предмет: структура безпеки для системи управління університетом

Метою дослідження є аналіз, огляд та деталізація проектів ЗВО та IT компаній спрямованих на поліпшення наукової, освітньої та організаційної діяльності ЗВО, встановлення універсального стандарту до інформаційно-аналітичних систем управління діяльністю ЗВО. У ході роботи описано стан інформаційних технологій в управлінні діяльністю ЗВО та виявлено нагальні проблеми, проведено порівняльний аналіз систем управління бізнес-процесами ЗВО, які використовують більшість університетів світу . Виходячи з цього – впровадження та модифікація інструментів для гарантії захисту та цілісності цих систем.

Методи дослідження: в роботі використовувались методи аналізу і класифікації, системного моделювання, порівняння, проектування логічних структур даних.

Для реалізації мети поставлено наступні **завдання дослідження**:

а) Проаналізувати існуючі систем та встановити їх переваги та недоліки.

б) Дослідити конкретні бізнес процеси ХДУ.

в) На основі отриманих результатів, створити вимоги щодо технічних характеристик серверної частини системи.

г) Відповідно до вимог спроектувати серверну частину, зокрема розробити структуру бази даних та публічний API.

ґ) Реалізувати комплекс заходів спрямованих на підвищення захисту даної системи

д) Написати документацію до публічного API.

Очікується, що новостворений продукт буде відповідати вимогам максимальної кількості учасників освітнього процесу в вищих навчальних закладах.

Наукова новизна: завдяки даній роботі буде створено умовний каталог систем управління ЗВО та описано послідовність впровадження конкретних систем для забезпечення надійного захисту сервісів.

Практичне значення одержаних результатів: створено робочу систему, яка має в своїй основі глибоке наукове дослідження.

РОЗДІЛ 1
СИСТЕМИ УПРАВЛІННЯ УНІВЕРСИТЕТАМИ

1.1. Аналіз українських систем управління ЗВО

Головним завданням використання ІТ в управлінні ЗВО є оптимізація функціонування і розвитку університету завдяки впровадженню останніх інформаційних технологій.

Використання ІТ в управлінні ЗВО має забезпечувати:

1. В освітній діяльності:

– функціонування найсучаснішого децентралізованого навчально-методичного середовища університету;

– впровадження ІТ технологій в освітньому процесі;

– реалізацію проектів, які базуються на компонентах електронного навчання;

– надання доступу до навчальних методик в міжнародному освітньому просторі.

2. В науковій діяльності:

– демонстрація наукового потенціалу навчального закладу у світовому інформаційному просторі;

– забезпечення можливості отримання данних для наукових співробітників до інформаційних ресурсів міжнародних наукових центрів;

– проведення спільних досліджень в складі інтернаціональних консорціумів.

3. В управлінні університетом:

– контроль процесами отримання, зберігання і аналізу інформації про стан університету, пошуку та обробки даних;

– автоматичне виконання затверджених рішень;

– модернізація планування управління закладом освіти;

– підвищення якості обліку і ефективності використання фінансових і матеріально-технічних ресурсів.[13]

Більшість відчизняних розробників, які мали досвід створення подібних систем, кажуть: «Реалізації систем управлінської діяльності зазвичай охоплюють широкий спектр цілей: від формалізації процедур пошуку та обробки інформації до проведення змін в організаційній структурі контролю і перерозподілу зобов'язань. Характерною ознакою такого типу проектів є те, що від успішності результатів впровадження може залежати ефективність функціонування ЗВО в цілому або його певних стурктурних одиниць. А тому детальне планування і контроль не лише технічних, а й людських аспектів запровадження системи мають критичне значення. Велика частина українських ЗВО намагаються самостійно вирішити проблему інформатизації навчального процеса, до того ж, для працівників освіти стали звичними додатки, які дозволяють ,наприклад, формувати розклад занять, керувати аудиторним фондом навчального закладу або раховувати робоче навантаження для викладачі. Однак ефективність кожної з цих розробок є сумнівною, оскільки зараз відсутній єдина форма управління навчальним закладом. Наступною негативною рисою є те, що програми від різних розробнків не мають можливості інтеграцій між собою. Це призводить до того, що поступово все більше навчальних закладів підтримують ідеї покупки чи створення самостійної системи управління, яка дозволить об'єднати всі сфери діяльності». Логічно, що проблема вибору відповідної системи нині є доволі актуальною. Аналіз існуючих інформаційних систем управління ЗВО, встановлення переваг та недоліків цих систем, а також стурктуризація результів їх впровадження дозволить покласти край даній проблемі. Головною метою створення інформаційної системи менеджменту ЗВО є забезпечення університету універсальними засобами для створення планів, контролю і реалізації державної політики в галузі освіти на основі інноваційних технологій. Під час її створення та інтеграції необхідно вирішити деякі завдання:

- створити моделі управлінської й освітньо-виховної діяльності університету у форматі інформаційної бази даних;
- створити та наповнювати єдину базу підтримки адміністративної, освітньої та навчально-методичної діяльності університету;
- створити та впровадити нові методики управління освітнім процесом в університеті на основі найновітніших інформаційних технологій;
- максимально зменшити час, необхідний для надходження критичної інформації
- оптимізувати роботу співробітників університету;
- забезпечити інформацією певні категорії користувачів системи;
- ввести єдині шаблони для електронних документів, які враховують існуючу нормативну базу;
- створити систему загального планування, сервіси прогнозування розвитку університету. [13]

Для вирішення таких задач навчальні заклади або користуються готовими програмними рішеннями або створюють власні системи. Нині на вітчизняному ІТ ринку налічується дестяки програмних продуктів, що дозволяють поліпшити навчальний процес університету. До найпопулярніших відносяться:

- АСУ «СТЕП 5 ПРОФ»,
- АСУ навчальним процесом «Директива»,
- АСУ «Університет» (ТОВ «UNITEX+»),
- Пакет комп’ютерних систем ПП «Політек-софт»,
- Програмний комплекс «АЛЬМА-МАТЕР»
- АСУ «Вищий навчальний заклад» НДІ ПІТ,
- ІАС «Університет» Херсонський державний університет,
- Електронна система управління ЗВО «Сократ»

Ефективність впровадження та доступність подібних систем управління в навчальних закладах різна і залежать від технічного забезпечення та підготовки персоналу, який буде робити інтеграцію.

Обираючи систему потрібно приділити увагу таким основним моментам:

– які підрозділи навчального закладу будуть представлені в цій системі;

– які навчальні процеси будуть інформатизовані;

– основні компоненти та тип системи.

До підрозділів, що найчастіше модернізують, відносяться основні структурні елементи ЗВО, зокрема, ректорат, деканати, освітні відділи, навчальна частина, приймальна комісія. Усі ці підрозділи є компонентами розглянутих систем. У деяких системах додано ще й фінансовий відділ, бібліотеку, відділ кадрів, гуртожитки, студентське самоврядування. До основних модулів, які представлені в системах належать:

- планування освітнього процесу;
- управління навчальним процесом;
- управління зарахуванням студентів;
- управління інформаційними ресурсами;
- управління фінансово-господарською діяльністю;
- управління науковою роботою.

Управління навчальним процесом майже ідентичне в усіх запропонованих системах і включає в себе наступні елементи:

- обсяг, планування, розподіл та контроль навчального навантаження викладачів;
- створення та ведення розкладу навчальних занять;
- облік та рух студентського контингенту. [13]

1.2. Аналіз іноземних систем управління ЗВО

За кордоном, системи для вищої освіти допомагають університетам та вищим навчальним закладам з залученням студентів, вступом, плануванням робочого процесу викладачів та управлінням фінансовою допомогою. Часто програмне забезпечення вищої освіти пропонує такі інструменти, як інформаційна система студентів, звітність про акредитацію викладачів, параметри eLearning та LMS, залучення випускників та управління документами. Навчальні заклади, які шукають програмне забезпечення вищої освіти, можуть також зацікавити програмним забезпеченням для управління аудиторією, програмним забезпеченням Gradebook, програмним забезпеченням для управління, обліку, грантами, стипендії та прийому.

Під час пошуку було виявлено декілька найбільш популярних систем, які відповідали вимогам та задовольняли критерії пошуку. Це:

Classter. Система є першопрохідцем у галузі освітніх технологій, даючи широкий спектр хмарний SaaS, який поєднує в собі: інформаційні системи для учнів, системи управління закладами освіти та системи контролю навчанням. Платформа пропонує різноманітні інструменти для управління інформацією, яке може використовувати будь-який заклад освіти. Повністю інтегрований з Office 365, Google G-Suite та іншими подібними системами: від ERP та державних баз даних до SMS-служб та додатків для BI [1].

Еллуціан. Система, створена для закладів вищої освіти. Вона надає послуги більш ніж 2500 закладам майже в 50 країнах. Інструментарій системи розширює діяльність та збагачує досвід понад 20 мільйонів студентів. Будучи провідним продуктом на ринку технологій освіти, кейс Ellucian є всеосяжним та створений найкращими практиками товариства, а також 50-річним досвідом трансформації галузевої екосистеми [2].

Граделінк CIC Система описує себе наступним чином:”Адміністратори, допоможіть вашому закладу виконати свою місію та покращити результати студентів за допомогою простих у користуванні навчальної книжки та програмного забезпечення SIS, що економить час.

Хмарні технології, готові стандарти, з інтегрованими планами занять. Звітування ніколи не було таким простим. Почніть по днях, а не по тижнях!”[3].

PowerVista RollCall. Програмне рішення, створене для галузевих та навчальних організацій. Просте в обслуговуванні та експлуатації. RollCall - це інстумент для управління закладом, який покриває базові потреби в безпеці, надає можливість автоматично вести звітність. Доступні установки для одного користувача та сервера / веб / хмари [4].

iGradePlus. Мультифункціональна та проста у використанні веб-система зберігання навчальних матеріалів та керування, що пропонує широкий вибір інструментів для вчителів та закладів освіти. Система включає в себе управління студентами, групами, факультетами, контроль відвідуваності, створення різноманітних звітів та широкі можливост для зв’язку студентів та викладачів [5].

CampusAnyware. CampusAnyware – це інформаційна система для студентів, яка працює у хмарному середовищі. Компоненти цієї системи дозволяють викладачам, студентам та співробітникам віддалено отримувати доступ до основних функцій закладу, коли вони їм потрібні. Програма досить просто переходить у робочий процес прийому та підбору персоналу, TextAnyware надсилає оновлення безпосередньо до спільноти університету, а документи зберігаються в цифровому форматі в DocAnyware [6].

Administrative Solutions 3. Комплексне програмне рішення для одного або декількох закладів. Воно має функціонал для майже всіх етапів освітнього процесу: вступ, планування навчання, контроль групи, відвідуваність, акаунти, фін / допомога, оплата житла, можливості кар’єри, веб-портали, бібліотеку та управління практикою [7].

mySkoolApp Веб-рішення, яке забезпечує моніторинг стану ЗВО через управління вступом, контроль відвідувань, інформування спільноти, тощо [8].

Veracross - це інтегрована система управління інформацією ЗВО, яка з'єднує всіх членів навчального процесу. Veracross підвищує ефективність закладу. Забезпечуючи комплексний доступ до всієї відповідної шкільної інформації, Veracross допомагає працювати продуктивно [9].

Аналізуючи ці системи управління ЗВО, було визначено деякі категорії, які властиві даним системам. Після їх порівняння, було створено таблицю, яка містить співставлення систем по різних ознаках. Воно базується на існуванні окремої функції або структури. Основними з них являються:

- Управління освітніми матеріалами. Можливість зберігати, редагувати та поширювати лекції, лабораторні та семінари всередині системи управління ЗВО;
- Управління харчуванням. Можливість замовити їжу на території університету. Відповідно наявність функціоналу для закладів харчування, які працюють на території ЗВО або є його партнерами;
- Звітність / аналітика. Можливість ствоювати звіт діяльності університету. Автоматичний збір та аналіз даних для наповнення звітності;
- Управління оцінками. Інструментарій для викладачів та студентів, що допомагає вести та контролювати журнал оцінок;
- Управління бібліотекою. Онлайн база книжок. Замовлення та бронювання книг. Контроль за їх обігом;
- Управління освітніми програмами. Система створення та контролю за навчальними програмами дозволяє викладачам розподіляти навчальні години, планувати заняття. Студентам вона надає можливість моніторити навчальний процес онлайн;
- Управління фінансуванням. Контроль за стипендіями, соціальними виплатами та грантами для наукової діяльність;
- Онлайн-платежі. Система, яка надає можливість провести більшу частину фінансових операцій онлайн;
- Контроль доступа. Комплекс заходів, які роблять систему безпечною та закритою для не авторизованих користувачів;

- Фінансовий менеджмент. Внутрішня бухгалтерія та контроль закупівель ЗВО;
- Портал для батьків / абітурієнтів;
- Облік випускників. База даних, яка містить список студентів, які завершили навчання в ЗВО. Дає можливість моніторити та проводити аналіз успішності випускників;
- Управління кадрами;
- Функціонал для гуртожитків;
- Благодійність в межах ЗВО;
- Планування внутрішньої та зовнішньої діяльності ЗВО;
- Інформація для студентів / записи. Таблиці розкладів, культурних заходів та ін;
- Інструменти для студентського самоврядування;
- Чати груп, факультетів. Контакти відділів ЗВО, викладацького складу;
- Календар подій;
- Інтерактивне навчання. Онлайн лабораторії, додатковий медіа-матеріал;
- Опитування / Голосування;
- Контроль безпеки;
- Функціонал для старост та кураторів;

1.3 Висновки до розділу

У цьому розділі був проведений аналіз вітчизняні та закордонні системи управління ЗВО. Було створено умовний каталог цих систем та проведено їх порівняння. Описано поняття структурного підрозділу подібних систем. Переглянуто основні види функціоналів, які має та чи інша система. Було показано основні переваги та недоліки таких систем. Управління бізнес-процесами університета являє собою багатогранний предмет для досліджень, де інформація змінюється та з’являється з неймовірною швидкістю, а

інновації впливають не лише на певні заклади, а й на цілі державні інституції. В наступному розділі буде розглянуто основні компоненти, алгоритми та методи захисту, що будуть використані в практичній частині роботи, а також інструменти, які буде використано для створення безпечного автономного середовища.

РОЗДІЛ 2
РЕАЛІЗАЦІЯ СИСТЕМИ БЕЗПЕКИ

2.1. Вибір архітектури системи

Вибір архітектури є критичним етапом при створенні будь-якого додатку. При розробці даного типу сервіса, цілком логічним вибором є клієнт-серверна архітектура, що показала з кращої сторони впродовж останніх років. Практична частина даної роботи буде створена саме спираючись на таки тип архітектури. Вона стала популярною саме завдяки стрімкому розвитку Інтернета а також значному розповсюдженню баз даних для збереження інформації. Таку архітектуру можна представити як абстарктну мережу, в якій усі ресурси розгорнуто на окремих серверах, які використовуються для конкретних цілей, а основними компонентами такої системи можуть бути наступні компоненти:

- Сервери, які віддають данні додаткам, що до них звертаються;
- клієнти, які роблять запити до конкретних сервісів;
- мережа, що моніторить і підтримує контакт між серверами та клієнтами.

Кожен із таких клієнтів може бути як сервером так і клієнтом даного сервера і вони мають свої обов’язки. Можна виокремити наступні основні рівні обв’язків:

- рівень представлення даних (так званий користувацький інтерфейс , який містить компоненти для взаємодії користувача із системою.);

- прикладний рівень (виконує основну логіку додатку);

- рівень управління даними (дозволяє використовувати дані та мати до них доступ). [21]

У дволанковій клієнт-серверній архітектурі виконується взаємодія двох основних компонентів клієнта та сервера. Важливо пам’ятати, що від делегування конкретних функцій для таких клієнтів та серверів можуть

існувати умовні моделі «товстих» та «тонких» клієнтів, де «товстими» клієнтами є такі пристрої як планшети, мобільні телефони та ін [20]. Трьохланкова клієнт-серверна архітектура базується на розділенні прикладного рівня та рівня управління даними. Створюється спеціальний програмний рівень, в якому базується логіка додатку, програми проміжного рівня працюють під управлінням окремих додатків, проте запуск таких додатків має бути воконаним з використанням спеціальних веб-серверів. Управління даними здійснюється відокремленим сервером даних. Дволанкова архітектура простіша, адже усі запити обробляються одним сервером, однак через це вона більш вазливою і потребує спеціальних заходів для підвищення продуктивності сервера. Триланкова архітектура більш складна, але завдяки тому, що функції розподілені між серверами другого і третього рівня, ця архітектура має наступні переваги:

1. Гнучкість і масштабованість;
2. Високий рівень безпеки;
3. Високу продуктивність.

Хоча трьохланкова архітектура і дає змогу реалізовувати більш гнучкі додатки, проте для даного типу додатку вона не підходить з ряду причин. По-перше, при використанні даного типу архітектури з одним сервером, якщо він несподівано припинить роботу чи трапиться помилка при передачі запиту, користувач не зможе працювати з системою. Звичайно, можна створити ще декілька екземплярів першого сервісу, що приєднані до балансувальника, однак при модерації коду одного з модулів, доведеться міняти весь проект і перезбирати його для кожного з таких серверів окремо. В контексті даного проекту такий результат є негативним, адже кожна підсистема повинна бути незалежною. По-друге, через велику кількість робіт та залежність від інструментів представлення даних, що повинні розгортатися в робочому середовищі або в хмарному сервісі, залежність одного сервера від таких модулів може значно зповільнити загальну роботу додатку в цілому. Оскільки важливою вимогою до роботи системи є її

швидкість, то використання такого типу архітектури не підходить. Також, якщо всі модулі додатку будуть розташовані в одному місці, то загальна характеристика системи, щодо її масштабованості буде замалою, а впровадження нового функціоналу може бути майже неможливим, Такі проблеми з'являються при використанні триланкової архітектури в більшості комплексних систем. Однак, рішення для усіх цих проблем є - можна розділити кожен із модулів системи на менші, більш автономні. Таке рішення надає мікросервісна архітектура. Офіційно, термін «мікросервіси» не має чіткого трактування, але їх суть полягає у тому, що вони представляють архітектурний стиль, в якому складні додатки представлені як сукупність маленьких, самодостатніх, децентралізованих сервісів, кожен з яких відповідає за окремий процес. Мікросервіси працюють один з одним виконуючи певноі дії. Вони «спілкуються» через API, які не залежать від мови програмування. Звідси, можна побачити основні переваги даного підходу розробки:

1. Швидке масштабування та розгортання. Кожен мікросервіс розгортається окремо. Якщо розробник змінює щось в одному з них, то він може застосувати ці зміни, не впливаючи на інші мікорсервіси. При чому, зміни можуть застосовуватись настільки часто, наскільки це необхідно, щоб конкретна частина додатку відповідала новим потребам.
2. Гнучкість додатків. Будь-який мікросервіс можна легко замінити чи оновити. Його можна переписати на іншу мову в межах прийнятного часу без необхідності переписувати систему вцілому.
3. Потенційно легша система для підтримки та тестування. Мікросервіси, зазвичай, є маленькими за обсягом. Завдяки цьому, розробникам простіше зрозуміти структуру системи. Таку систему набагато простіше підтримувати, оскільки достатньо змінити конкретний компонент і ця зміна не пошкодить інші частини додатку. Більш того, таку систему набагато легше тестувати. При чому

тестувальникам також необхідно перевіряти лише певні елементи системи, а не усю систему в цілому.

4. Помилка в одному мікросервісі не нашкодить системі. Помилки у окремому мікросервісі не повинні зламати систему.

5. Мікросервіси, написані на різних мовах програмування можуть працювати разом. [22]

Хоча мікросервісні системи мають величезні переваги над іншими, вони не ідеальні, адже мають ряд своїх недоліків, які потрібно розуміти. Основними недоліками є:

- складність розробки;
- підтримка роботи та обробка запитів є набагато складнішою в порівнянні з використанням монолітного підходу.
- підтримка безпеки для такого типу додатків є набагато складнішою, оскільки кожен запит повинен бути захищений спільним токеном, або використовувати один і той же механізм захисту
- оскільки усі підсистеми рохташовані в одному місці, в мікросервісній архітектурі додаток не є розподіленим, тому при використанні декількох сервісів при обробці запиту, кожен сервіс повинен передавати маркери того, що користувач має права на виконання даної операції.

Хоча, мікросервісна архітектура має багато недоліків в порівнянні з монолітним підходом, проте переваги, які вона надає , є вагомим аргументом в її користь. Оскільки хмарні середовища широко підтримують принципи контейнеризації, то розгортання системи, що використовує мікросервісну архітектуру є найкращим варіантом.

2.1 Огляд інструментів та основних компонентів захисту системи

Як зазначалося в попередньому розділі, для реалізації додатку було необхідно створити велику кількість окремих підрозділів та надати певні

привілеї деяким групам користувачів, а отже, додаток повинен складатися з маленьких частинок-сервісів, кожна з яких виконує певну, покладену на неї роботу. Було вирішено, що додаток буде складатися з декількох частин, кожна з яких виконує різну роботу – дозволяє користувачу взаємодіяти з іншими сервісами, відповідає за авторизацію і т.д.

Загальний список компонентів системи:

1. Сервіс користувацького інтерфейсу – відповідальний за всі види взаємодії користувача із системою, об’єднаний майже з усіма сервісами у системі, саме з цього сервісу починається робота усіх процесів у системі; 2. Сервіс авторизації та аутентифікації – відповідає за процес створення нового користувача, оновлення профілю та основних даних про користувача, генерує токени для доступу користувача до інших систем, також є центральним компонентом для перевірки прав користувачів на виконання різних типів дій;

3. Сервіс отримання даних – відповідає за отримання даних із бази відповідно до отриманих користувачем привілеїв;

4. Сервіс обробки – використовується сервісом отримання даних для швидкої обробки даних ;

6. Сервіс моніторингу та діагностики системи – використовується для логування помилок та збоїв у роботі сервісів системи та додатку у цілому.[22] Для реалізації кожного із сервісів використовується ряд фреймворків та засобів, для реалізації серверної частини використовується Django - високорівневий відкритий Python-фреймворк (програмний каркас) для розробки веб-систем.

Для тих компонентів, що мають користувацький інтерфейс було вирішено використати бібліотеку React (старі назви: React.js, ReactJS) — відкриту JavaScript бібліотеку для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків. Для забезпечення взаємодії між кожним сервісом використовується REST - підхід

до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів.

Деякі сервіси мають джерело збереження даних, а саме SQL бази даних, як наприклад сервіс аутентифікації та авторизації, де дані про користувача зберігаються у вигляді таблиць. Для зібрання усіх сервісів у контейнери використовується Docker. Сам додаток розміщено у хмарному середовищі у вигляді контейнера. Деякі з сервісів використовують сторонні API для роботи, як наприклад, сервіс моніторингу та діагностики системи

2.3 Методи захисту системи

Процес входу користувача в систему є чи найвразливішим етапом у системі. Він вміщує у себе основні функції передачі та прийому даних від інших сервісів через інтерфейси API та передачу JSON даних на клієнтській стороні з використанням Javascript фреймворка ReactJS. Основними кроками авторизації користувача в системі є:

1. Вхід користувача на веб-сайт сервісу;

2. Введення користувацького пароля та елекронної пошти в відповідній формі входу;

3. Вибір ролі користувача, яка буде надавати певні привілеї або обмеження як на функціонал так і на отримувані данні.

При вході у профіль користувач може поміняти свої дані, такі як пароль, опис профілю та інше. При цьому відбувається передача данних до сервісу авторизації та аутентифікації. Оскільки для даного додатку необхідно зберігати та отримувати дані, то було вирішено додати сервіс авторизації та аутентифікації, що зберігає дані про користувача, а також видає ключі для використання інших сервісів у проекті. Для процесу авторизації найбільш якісним є JSON Web Tokens (JWT).

JSON Web Token (JWT) - це JSON об'єкт, який визначений у відкритому стандарті RFC 7519. Він вважається одним з безпечних способів

передачі інформації між двома учасниками. Для його створення необхідно визначити заголовок (header) із загальною інформацією по токена, корисні дані (payload), такі як id користувача, його роль і т.д. і підписи (signature). Простими словами, JWT - це лише рядок у наступному форматі header.payload.signature.

Припустимо, що ми хочемо увійти в свій профіль на сайті. У нашому випадку є три учасники - користувачів, сервіс авторизації та аутентифікації. Сервер аутентифікації буде забезпечувати користувача токеном, за допомогою якого він пізніше зможе взаємодіяти з додатком. Додаток використовує JWT для перевірки аутентифікації користувача в такий спосіб:

1. Спершу користувачів заходить на сервер аутентифікації за допомогою аутентификационного ключа (це може бути пара логін / пароль, або Facebook ключ, або Google ключ, або ключ від іншого облікового запису).
2. Потім сервер аутентифікації створює JWT і відправляє його користувачеві.
3. Коли користувач робить запит до API додатка, він додає до нього отриманий раніше JWT.
4. Коли користувач робить API запит, додаток може перевірити по передачі із запитом JWT є користувач тим, за кого себе видає.

У цій схемі сервер додатки налаштований так, що зможе перевірити, чи є вхідний JWT саме тим, що був створений сервером аутентифікації (процес перевірки буде пояснень пізніше більш детально).

JWT складається з трьох частин: заголовок header, корисні дані payload та підпис signature. Для створення токену, треба пройти декілька етапів.

Крок 1 - створюємо HEADER. Хедер JWT містить інформацію про те, як повинен обчислюватися JWT підпис. Хедер - це теж JSON об'єкт, Який виглядає наступним чином:

header = { "alg": "HS256", "typ": "JWT"}

Поле typ не говорити нам нічого нового, окрім того, що це JSON Web Token. Цікавіше тут буде поле alg, яке визначає алгоритм хешування. Він буде використовуватися при створення підпису. HS256 - не що інше, як HMAC-SHA256, для його обчислення потрібен лише один секретний ключ. Ще може вживатися інший алгоритм RS256 - на відміну від попереднього, він є асиметричним і створює два ключі: публічний і приватний. За допомогою приватного ключа створюється підпис, а за допомогою публічного тільки перевіряється справжність підпису, тому нам не потрібно турбуватися про його безпеку.

Крок 2 - створюємо PAYLOAD. Payload - це корисні дані, які зберігаються всередині JWT. Ці дані також називають JWT-claims (заявки). У прикладі, Який розглядаємо ми, сервер аутентифікації створює JWT з інформацією про id користувача - userId.

payload = { "userId": "b08f86af-35da-48f2-8fab-cef3904660bd"}

В прикладі, було покладено тільки одну заявку (claim) в payload, але можна покласти стільки заявок, скільки завгодно. Існує список стандартних заявок JWT payload - ось деякі з них: iss (issuer) - визначає додаток, з якого відправляється токен; sub (subject) - визначає тему токена; exp (expiration time) - час життя токена. Ці поля можуть бути корисними при створення JWT, але вони не є обов'язковими.

Крок 3 - створюємо SIGNATURE. Підпис обчислюється наступним чином: алгоритм base64url кодує хедер і payload, створені на 1 і 2 кроці. Алгоритм з'єднує закодовані терміни через точку. Потім отриманий рядок хушується алгоритмом, заданим в хедері на основі секретного ключа.

Крок 4 - об'єднати всі три JWT компонента разом. Тепер, коли є всі три складових, можна створити JWT. Це досить просто- треба з'єднати всі отримані елементи в рядок через точку.

Дуже важливо розуміти, що використання JWT не приховує і не маскує дані автоматично. Причина, чому JWT використовуються - це перевірка, що дані були дійсно відправлені авторизованим джерелом. Дані всередині JWT

закодовані і підписані і це не одне і теж, що зашифровані. Мета кодування даних - перетворення структури. Підписані дані дозволяють одержувачу даних перевірити аутентифікацію джерела даних. Таким чином закодованість і підпис даних не захищає їх. З іншого боку, головна мета шифрування - це захист даних від несанкціонованого доступу. Оскільки JWT тільки закодована і підписана, і оскільки JWT не зашифровані, JWT не гарантує ніякої безпеки для чутливих (sensitive) даних.

Крок 5 - перевірка JWT. У системі управління ВНЗ було використано JWT, який підписаний за допомогою HS256 алгоритму і тільки сервер аутентифікації і сервер додатка знають секретний ключ. Сервер програми отримує секретний ключ від сервера аутентифікації під час установки аутентифікаційних процесів. Оскільки додаток знає секретний ключ, коли користувач робить API-запит з доданим до нього токеном, додаток може виконати той же алгоритм підписування до JWT, що в кроці 3. Додаток може потім перевірити цей підпис, порівнюючи його зі своїм власним, обчисленим хешуванням. Якщо підписи збігаються, значить JWT валідний, тобто прийшов від перевіреного джерела. Якщо підписи не збігаються, значить щось пішло не так - можливо, це є ознакою потенційної атаки. Таким чином, перевіряючи JWT, додаток додає довірчий шар (a layer of trust) між собою і користувачем.

2.4. Висновки до розділу

В даному розділі було проаналізовано основні підходи та методики створення безпечної системи. Питання вбереження даних буде вирішено з використанням операції токенізації і вибору необхідних ключових значень для закритих груп користувачів, які не мають загального доступу до API для отримання даних. Оскільки дані будуть конфіденційними, було вирішено використовувати систему балансування доступу по ролям в організації . Архітектура має вирішальну роль на всіх стадіях її розробки, тому було

приведено всі за і проти використання різних архітектур, однак вибір веб-сервісної архітектури, а саме мікросервісної є найбільш вдалим саме для системи управління бізнес процесами університету, оскільки це надасть можливість швидко обробляти дані та захистить сервіс від зайвих навантажень та виходу з ладу.

РОЗДІЛ 3
АНАЛІЗ МОЖЛИВИХ ВРАЗЛИВОСТЕЙ ТА ЇХ ЗАПОБІГАННЯ

3.1. Основні типи атак та вразливостей

Сервери зберігають веб-сторінки та видають їх користувачу за запитом, обробленому через HTTP, якій у всій архітектурі має головну роль як протокол, відповідальний за передачу даних від клієнта до сервера, і навпаки. HTTP-сервер - це компонент програмного забезпечення, який розуміє веб-адреси (URL) і HTTP. З боку програмного забезпечення, веб-сервер включає в собі певну кількість елементів, які керують доступом до файлів користувачів, які розташовані на сервері. Саме тому будь-яка діра в безпеці, базі даних, ОС або в мережі може надати можливість для атаки. Зазвичай, мережа схильна до атаки з боку хакерів, які ведуть свою діяльність по різних векторах вразливостей. Під атакою на WEB-сервер розуміється порушення стандартної роботи сервера, видалення або зміна його вмісту або отримання доступу до машини без відповідного дозволу. Беручи до уваги безліч технологій, що використовуються як складові веб – сервера, постає задача проведення аналізу існуючих вразливостей у веб-серверів та способів захисту від них.

3.1.1 SQL-ін’єкції

База даних – це одна з найзручніших структур для зберігання інформації. Здебільшого, доступ до бази здійснюється з допомогою SQL – мови, яка виконується завдяки інтерпретатору [19]. Виконання такої мови включає час запуску, який інтерпретує код і реалізує інструкції, які йому були поставлені. [20]Через те, як інтерпретуються мова, виникає прогалина в безпеці, відома під назвою ін’єкція коду [31]. Впровадження ін'єкції виникає тоді, коли незахищені дані надсилаються інтерпретатору як компонент команди або запиту. Процес звернення до сховища даних, ідентичний,

незалежно від того, чи був цей запит створений завдяки діям невідомого користувача або адміністратора додатку [29]. Веб-додаток працює як дискреційне керування доступом до сховища інформації, створюючи команди для отримання, запису або модифікації даних у базі на основі привілеїв користувача. Успішна ін'єкція, яка модифікує запит, може оминути стандартні засоби надання доступу додатка і отримати несанкціоновані можливості [32]. Якщо логіка додатку міняється, спираючись на результати запиту, хакер може підмінити запит, щоб злаати логіку програми.

Багато програм, які надають функцію входу в систему на основі заповнюваних полів у браузері, використовують базу даних для зберігання конфіденційної інформації користувача і виконують простий скрипт SQL для перевірки входу в систему.

3.1.2 XSS-атака

XSS атака - це атака на сервер, що вставляє будь-який HTML /JavaScript код в результат роботи логіки в тих випадках, коли логіка не коректно обробляє інформацію, яку надіслав користувач. У зв’язку з тим, що фільтація не працює - сервер не помічає дані на присутність в них заборонених знаків -, <, >, ',«.

Код може включати в себе скрипт, який може пошкодити цілісність системи на комп'ютері жертви завдяки вразливостям веб-браузера. Також міжсайтовий скриптинг може мати шкідливий JavaScript код, який відправляє захищені дані сеансу на інший сервер [34].

Міжсайтовий сценарій може бути використаний для того щоб надати неправильну інформацію жертві [25]. Вміст может мати функціонал входу в систему, який в разі успіху буде пересилати конфіденційні дані на сторонній веб-сервер замість цільового сервера.

Даний тип атак незвичайний тим, що шкідливий код з сервера виконується на комп'ютері користувача, причому виконує цей код сама жертва [26].

3.1.3 CSRF -атака

Підробка міжсайтових запитів (CSRF) - це виконання шкідливих дій в веб-додатку через користувача який проходять перевірку автентичності [40]. Вразливості CSRF з’являються, коли програми використовують лише cookie-файли HTTP для ідентифікації користувача, який працює з системою [41]. Оскільки браузери додають cookie в запити автоматично, не зважаючи на джерела запиту, хакер може написати веб-сайт, який копіює междоменной запит до незахищеного додатком. Зловмисник створює подібні HTTP-запити і змушує жертву надсилати їх через теги, XSS або інші методи [42]. Атакуючий має можливіть виконати будь-яку операцію з зміни стану: вхід в систему, отримання прихованих даних, інформацію по транзакціях.

3.1.4 Broken authentication

Цей тип атак дозволяє хакеру або взяти контроль, або оминути методи аутентифікації, які використовуються веб-додатком, перехопивши ID користувача [43]. Для того, щоб ідентифікувати користувачів, web-додаток використовує сесійні куки [44]. Після того, як користувач ввійшов у системо, і додаток його авторизував, в пам’ять браузера зберігається унікальний ідентифікатор, який браузер пізніше надає серверу при кожному запиті [46]. Таким чином web-додаток розуміє, що це реальний користувач. У разі, якщо зловмисник захопить ідентифікатор сесії, він отримає доступ з правами як у профіля користувача [47]. Тому ціль атаки некоректної аутентифікації - отримати один або декілька профілів користувачів та отримати їх права [44].

Наступні типи вразливостей дозволяють хакеру оминути методи аутентифікації:

- Дані входу користувача не захищені при збереженні.
- Дані користувача (логін і пароль), які можна відгадати
- Ідентифікатори сеансів показані в URL-адресі
- Сеанс не закривається або не стає недійсним після виходу з системи.
- Ідентифікатори сеансів не повертаються після успішного входу.
- Ідентифікатори сеансів та інші облікові дані надсилаються через не захищені канали [47].

3.1.5 Атака неправильної конфігурації

Атака неправильної конфігурації зосереджені на вразливостях налаштування серверів [28]. Це одна з вразливостей, яку складно знайти. Вразливість базується на тому, як оброблюються параметри в додатку, а не від программного коду [29].

Неправильна конфігурація безпеки може статися на будь-якому рівні додатків, включаючи платформу, веб-сервер, сервер додатків, базу даних і фреймворк [30].Тому неправильні конфігурації безпеки можуть різнитись від налаштування фреймворків до встановлення прав доступу облікових записів [32]. Багато програм мають ненадійні та просто зайві функції, такі як налагодження та забезпечення якості, включені відповідно до існуючих стандартів [34]. Саме ці непотрібні функції дають хакерам можливості порушити процес аутентифікації та отримання доступу до приватної інформації [35]. Якщо додаток використовує веб-сервер, фреймворк, платформи додатків, базу даних або містить будь-якої код - це може стати причиною неправильної конфігурації безпеки системи [36]. Загальновідомим фактом є те, що неправильна конфігурація є серйозною уразливістю, з якою

стикаються безліч компаній [36]. Це пов’язано з тим, що найбільш поширені неправильні конфігурацій в центрах обробки даних є:

- конфігурації за замовчуванням, які не змінювалися і не модифікувались;
- використання тимчасових конфігурацій;
- проблеми з плануванням поведінки програми та вимог до підключення.

Без коректного розуміння неправильна конфігурація безпеки створює нові ризики для робочих середовищ. Приклади неправильних налаштувань безпеки:

- Зайві порти адміністрування, які відкриті для будь-якого додатку. Вони роблять додаток не захищеним для віддалених атак.
- Вихідні підключення до різних інтернет-сервісів.
- Відсутність оновлень інтегрованих програм.

3.2 Способи захисту системи від атак

3.2.1 Захист від SQL-ін’єкцій

За замовчуванням, Django має вбудований захист від даного типу атак, однак є декілька рекомендацій, яких слід дотримуватись:

1. Проводити оновлення всіх компонентів веб-додатків, включаючи бібліотеки, плагіни, програмне забезпечення веб-сервера і програмне забезпечення бази даних.

2. Дотримуватись принципів мінімальних привілеїв у зовнішніх посиланнях при створенні облікових записів, які використовуються для підключення до бази даних SQL.

3. Налаштувати логування системи так, щоб клієнт не отримував повідомлення про помилки баз даних. Хакери можуть використовувати інформацію з повідомлень про помилки, щоб підлаштувати свої запити для успішної атаки [47].

3.2.2 Захист від XSS-атак

1. Використання HttpOnly. Коли веб-сервер зберігає файли cookie, він може додати певні атрибути, щоб гарантувати, що файли cookie будуть заблоковані при використанням шкідливого JavaScript. HttpOnly гарантує, що файли cookie будуть відправлятися саме на той домен, з якого вони були створені.

2. Кодувати вихідні змінні. Щоб захиститись від XSS атак, додатку необхідно впевнитись, що всі вихідні дані закодовані, а потім повернуті цільовому користувачу. Кодування вихідних змінних замінює розмітку HTML певними об’єктами, які називаються «entities». При використанні entities браузер показує об'єкти, але не ініціює їх. При цьому коли веб-браузер отримує об'єкти, вони трансформуються назад в HTML і відображаються, але не запускаються.

3. Використовувати crossing boundaries policy. При дотриманні подібної політики всі авторизовані користувачі на сайті повторно мусять вводити свої конфіденційні дані, перш ніж їм буде надано доступ до ресурсів сайта [41]. Навіть якщо у авторизованого користувача є файл cookie, який дає змогу ввійти в систему, він в буль-якому випадку повинен повторно вводив свої ім'я та паролі для отримання доступу до певних сторінок. Ця стратегія ефективна для захисту від атаки XSS тому, що вона дійсно обмежує шанс захоплення сеансу хакером.

3.2.3 Захист від CSRF –атаки

Захист від CSRF вимагає:

- створення GET-запитів без побічних ефектів
- контроль, щоб не-GET-запити могли створюватись тільки з клієнтського коду [42].

Щоб це зробити, необхідно:

1. Передача стану уявлення (REST) - це принципи проектування систем, які наділяють певні типи дій (перегляд, створення, видалення, оновлення) різним HTTP-verbs. REST-фул типи будуть тримати код в чистоті і допоможуть масштабувати ресурс швидше. Також, REST базується на тому, що запити GET використовуються лище для перегляду сторінок.

Відсутність зайвого в GET-запитах зменшить шкоду, яка може бути завдана навмисно створеними URL-адресами - зловмиснику доведеться продумати шлях створення шкідливих POST-запитів [43].

2. Використовувати Anti-Forgery Tokens. Якщо дії по модифікації обмежені non-GET-запитами – це не гарантує безпеку. POST-запити можна відправляти із сценаріїв і сторінок, розміщених на інших доменах. Щоб гарантувати, що обробляються виключно дійсні HTTP-запити, необхідно включати унікальний токен в кожну HTTP відповідь щоб сервер перевіряв його при поверненні в повторних запитів, що використовують метод POST.

3. Перевіряти, що файли Cookies мають атрибут Cookie SameSite.

Команда Google Chrome створила атрибут до заголовка Set-Cookie, щоб запобігти CSRF. Атрибут cookie з одним сайтом дозволяє розробникам інспектувати, контролювати, чи надсилати файли cookie разом із запитом, ініційованим доменами сторонніх розробників.

3.2.4 Захист від Broken authentication атаки

1. Необхідно впроваджувати лише безпечний вбудований менеджер сеансів на стороні сервера, який завжди створює ідентифікатор сеансу після входу в систему. Ідентифікатори сеансів не повинні показуватись в URL-адресі, бути надійно збереженими і видалятись після виходу з системи або абсолютних тайм-аутів [46].

2. Впровадити належні засоби валідації надійності пароля. Головною проблемою є надійність пароля. Політика надійних паролів зводить на нуль можливість відгадати пароль за допомогою ручних або автоматичних засобів:

- Довжина пароля має бути не менше ніж 10 символів та не обмежувати максимальну довжину.
- Складність пароля. Механізми паролів повинні надавати можливість вводити будь-який символ.
- Паролі повинні бути чутливі до регістру.

Спеціалісти по кібер-безпеці вважають,що: «Механізм зміни пароля повинен мати мінімальний набір ознак, який має сенс для програми та користувачів.

- Пароль має відповідати 3 з 4 правилам складності
- Мінімум один символ верхнього регістру (A-Z);
- Мінімум один рядковий символ (a-z);
- Мінімум одна цифра (0-9);
- Мінімум один спецсимвол (НЕ пунктуація);
- Не більше 2 ідентичних символів підряд».

3. Заборонити розповсюджені паролі. Вимагати мінімального зміни схожості між старими і новими паролям [48].

4. Вимагати повторного входу для чутливих функцій. Щоб знизити ризик CSRF і перехоплення сеансів, важливо введення облікових даних перед оновленням конфіденційної інформації профілю (пароль, електронна пошта, важливі операції) [48].

3.2.5 Захист від атаки неправильної конфігурації

1) Зменшити кількість уразливостей за допомогою повторюваного тестування

2) Тримати програмне забезпечення в оновленому стані

3) Вимикати всі облікові записи за замовчуванням і регулярно змінювати паролі

4) Розробити сильну архітектуру додатків і шифрувати дані, які містять приватну інформацію.

5) Переконатись, що параметри безпеки встановлено на захищені значення.

6) Виконувати регулярні перевірки та сканування для ідентифікації помилок у системі

7) Використовувати конфігурацію для виробництва, розробки і постановки, оскільки невідповідності відкривають діри для багатьох неправильних конфігурацій.

8) Автоматизувати систему, де це можливо, щоб уникнути людських помилок [49].

9) Перевірити, чи немає жодних облікових записів за замовчуванням і чи були змінені їхні паролі. Облікові записи за замовчуванням обов'язково викликають проблеми.

ВИСНОВКИ

Для виконання поставлених завдань було проведено аналіз функціоналів існуючих систем управління ЗВО, та вироблено таблицю критеріїв для їх порівняння. Проведено огляд існуючих систем керування бізнес-процесами та структурними підрозділами закладів освіти на прикладах вітчизняних та зарубіжних систем, виділено список основних можливостей, котрі реалізуються ними. Також було проаналізовано можливі слабкі місця та недоліки даних систем, які було враховано при розробці нашої системи та

На основі проведеного аналізу розроблено базові вимоги щодо можливостей серверної частини та структуру захисту даних. Суттєву частину роботи приділено аналізу існуючих технологій всіх рівнів для створення веб-додатків та веб-сервісів. Детально досліджено роботу клієнт-серверних додатків та мікро-сервісної архітектури. Розглянуто та обгрунтовано використання підходу з використанням REST full API при проектуванні системи. Приділено увагу безпеці сервісів, використано криптографічно стійкі рішення забезпечення доступу до інформації всередені сервісів.Відповідно до створених вимог розроблено архітектуру серверної частину спроектованої системи. Після проведення аналізу популярних технологій розробки веб-сервісів для реалізації основної частини системи обрано Django з бібліотекою DRF. Реалізовано структуру бази даних засобами PostgreSQL, моделі з використанням Django ORM. Розроблено публічний прикладний програмний інтерфейс (API), та сформовано документацію до нього. При написанні ключових частин використано спеціальну форму коментарів, що забезпечують інтеграцію опису функцій та їх параметрів в документацію популярних IDE. Останнє є корисним при подальшій розробці, особливо при використанні існуючої кодової бази сторонніми розробниками, що є можливим, зважаючи на модульність проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Головна сторінка системи Classter. — Режим доступу: https://www.classter.com.
2. Головна сторінка системи Елуціан. — Режим доступу: https://www.ellucian.com
3. Головна сторінка системи Граделінк. — Режим доступу: https://www.gradelink.com
4. Головна сторінка системи PowerVista RollCall. — Режим доступу: http://www.powervista.com
5. Головна сторінка системи iGradePlus. — Режим доступу: https://igradeplus.com
6. Головна сторінка системи CampusAnyware. — Режим доступу: https://www.edctechnology.com
7. Головна сторінка системи Administrative Solutions 3. — Режим доступу: http://www.alaquest.com
8. Головна сторінка системи mySkoolApp. — Режим доступу: https://www.myskoolapp.com
9. Головна сторінка системи Veracross. — Режим доступу: https://www.veracross.com
10. Киричек Г. Г. , Киричек О. Модель оцінки плагіату програмного коду на основі системи контролю версій//Восточно Европейский журнал передовых технологий. — 2012. — 2 (2). — С. 25—28.
11. Кравцов Д. Webпортал «Херсонський Віртуальний Університет».— 2010.—Режим доступу: http://dls.kherson.ua/(датазверн.11.04.2019).
12. Кучер В. Мікросервісна архітектура та її особливості. — 2018.
13. Львов М. ,Співаковський О. ,Щедролосьєв Д. Інформаційна система управління вищим навчальним закладом як платформа реалізації

управління академічним процесом // Комп'ютер у школі та сім'ї. — 2007. — No 2. — С. 3—6.

14. Львовский С. LATEX: подробное описание. — 2003.

15. Побудова ІКТ інфраструктури ЗВО: проблеми та шляхи вирішення / О. Співаковський [та ін.]. — 2014.

16. Сервіс «KSU Feedback». —Режим доступу: http://feedback.kspu.edu/(дата зверн. 11.04.2019).

17. Сервіс «Пошук книг в електронному каталозі бібліотеки». —2010. — Режим доступу: http://elibrary.kspu.edu/(дата зверн.11.04.2019).

18. Система дистанційного навчання «KSU Online». — Режим доступу: http://ksuonline.kspu.edu/ (дата зверн. 11.04.2019).

19. Співаковський О. В. Webпортал Херсонського державного університету. — 2013. — Режим доступу: http://kspu.edu/(дата зверн.11.04.2019).

20. Sankar R. Burpsuite – A Beginner's Guide For Web Application Security or Penetration Testing [Електронний ресурс] / Ravi Sankar. – 2018. – Режим доступу: https://kalilinuxtutorials.com/burpsuite/.

21. Ganore P. What Is A Web Server And How Does It Function? [Електронний ресурс] / Pravin Ganore. – 2017. – Режим доступу: https://www.milesweb.com/blog/hosting/web-server-function/.

23. How a Web server functions? [Електронний ресурс]. – 2006. – Режим доступу: https://www.eukhost.com/blog/webhosting/how-a-web-serverfunctions/.

24. Web Server and its Types of Attacks [Електронний ресурс]. – 2012. – Режим доступу: https://www.greycampus.com/opencampus/ethicalhacking/web-server-and-its-types-of-attacks.

25. Brewer J. Web Server Vulnerabilities and a Defense in Depth Strategy Using the Squid Proxy [Електронний ресурс] / Jim Brewer // GSEC Practical version 1.4b. – 2004. – Режим доступу: https://www.giac.org/paper/gsec/3729/web-server-vulnerabilities-defense-in-depthstrategy-squid-proxy/105970.

26. Common Website Security Vulnerabilities [Електронний ресурс] //

Common places. – 2019. – Режим доступу: https://www.commonplaces.com/blog/6-common-website-security-vulnerabilities/.

27. Web Server Vulnerabilities Attacks: How to Protect Your Organization [Електронний ресурс] // Tech Funnel. – 2018. – Режим доступу: https://www.techfunnel.com/information-technology/web-server-vulnerabilitiesattacks-how-to-protect-your-organization/.

28. Уязвимости веб-приложений [Електронний ресурс]. – 2019. – Режим доступу:https://www.ptsecurity.com/upload/corporate/ruru/analytics/Web-Vulnerabilities-2019-rus.pdf.

30. Top 8 Network Attacks by Type in 2017 [Електронний ресурс] // CALYPTIX. – 2017. – Режим доступу: https://www.calyptix.com/topthreats/top-8-network-attacks-type-2017/.

31. Евтеев Д. SQL Injection от А до Я [Електронний ресурс] / Дмитрий Евтеев. – 2008. – Режим доступу: https://www.ptsecurity.com/upload/corporate/ru-ru/analytics/PT-devteev-AdvancedSQL-Injection.pdf.

32. SQL инъекции. Проверка, взлом, защита [Електронний ресурс] // BVN2. – 2011. – Режим доступу: https://habr.com/ru/post/130826/.

33. How to Prevent SQL Injection Attacks [Електронний ресурс] eSecurityPlanet. – 2018. – Режим доступу до ресурсу: https://www.esecurityplanet.com/threats/how-to-prevent-sql-injection-attacks.html.

34. How to Protect Against SQL Injection Attacks [Електронний ресурс] //UC Berkeley. – 2019. – Режим доступу: https://security.berkeley.edu/education-awareness/best-practices-how-articles/systemapplication-security/how-protect-against-sql.

35. SQL_Injection_Prevention_Cheat_Sheet [Електронний ресурс] –Режим доступу:https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/SQL_Injection

36. Choudhary A. SQL Injection Attacks: Know How to Prevent Them [Електронний ресурс] / Archana Choudhary // Security Zone. – 2019. – Режим

доступу: https://dzone.com/articles/sql-injection-attacks-know-how-toprevent-them.

37. Cobb M. Cross-site scripting explained: How to prevent XSS attacks [Електронний ресурс] / Michael Cobb // 2009 – Режим доступу: https://www.computerweekly.com/tip/Cross-site-scripting-explained-How-toprevent-XSS-attacks.

38. Singh S. 5 Practical Scenarios for XSS Attacks [Електронний ресурс] /Satyam Singh // Pentest Tools. – 2018. – Режим доступу: https://pentesttools.com/blog/xss-attacks-practical-scenarios/.

39. Cross-site Scripting (XSS) [Електронний ресурс]. – 2018. – Режим доступу: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS).

40. Cross Site Scripting (XSS) Attack Tutorial With Examples, Types & Prevention [Електронний ресурс]. – 2019. – Режим доступу: https://www.softwaretestinghelp.com/cross-site-scripting-xss-attack-test/.

41. Excess XSS [Електронний ресурс]. – 2016. – Режим доступу: https://excess-xss.com.

42. Cross Site Scripting (XSS) Attack Tutorial With Examples, Types & Prevention [Електронний ресурс]. – 2019. – Режим доступу: https://www.softwaretestinghelp.com/cross-site-scripting-xss-attack-test/.

43. Методы защиты от CSRF-атаки [Електронний ресурс]. – 2016. –

44. Режим доступу до ресурсу: https://habr.com/ru/post/318748/.

45. Cross-Site Request Forgery (CSRF) [Електронний ресурс]. – 2018. – Режим доступу: https://www.owasp.org/index.php/CrossSite_Request_Forgery_(CSRF).

46. Testing for CSRF (OTG-SESS-005) [Електронний ресурс]. – 2019. – Режим доступу: https://www.owasp.org/index.php/Testing_for_CSRF_(OTG-SESS-005).

47. Testing for CSRF (OTG-SESS-005) [Електронний ресурс]. – 2019. – Режим доступу: https://www.owasp.org/index.php/Testing_for_CSRF_(OTG-SESS-005).

48. Broken Authentication and Session Management [Електронний ресурс]. – 2010. – Режим доступу: https://www.owasp.org/index.php/Broken_Authentication_and_Session_Management.

49. Charan H. Broken Authentication and Session Management—part Ⅰ [Електронний ресурс] / Hari Charan. – 2017. – Режим доступу: https://medium.com/@grep_security/broken-authentication-and-sessionmanagement-part-ⅰ-50e760c9f599.

50. Charan H. Broken Authentication and Session Management [Електронний ресурс] / Hari Charan // DZone. – 2017. – Режим доступу: https://dzone.com/articles/broken-authentication-and-session-managementpart.

51. Using Burp to Test for Sensitive Data Exposure Issues [Електронний ресурс] // PortSwigger. – 2018. – Режим доступу: https://support.portswigger.net/customer/portal/articles/1965730-using-burp-to-testfor-sensitive-data-exposure-issues.

52. Common Vulnerability Scoring System v3.0 [Електронний ресурс] – Режим доступу: https://www.first.org/cvss/cvss-v30-specificationv1.7.pdf.

53. Common Vulnerability Scoring System v3.0: Specification Document [Електронний ресурс]. – 2019. – Режим доступу: https://www.first.org/cvss/specification-document.

54. Common Vulnerability Scoring System Calculator Version 3 [Електронний ресурс] – Режим доступу: https://nvd.nist.gov/vulnmetrics/cvss/v3-calculator.

55. Common Vulnerability Scoring System v3.0: Specification Document [Електронний ресурс]. – 2019. – Режим доступу: https://www.first.org/cvss/specification-document.

56. Safonov L. APTSimulator — тестирование противодействия APT угрозам [Електронний ресурс] / Luka Safonov. – 2018. – Режим доступу : https://habr.com/ru/post/350066/.

57. Luka S. Обзор площадки для тестирования веб-уязвимостей OWASP

Top-10 на примере bWAPP [Електронний ресурс] / Safronov Luka. – 2015. –Режим доступу: https://habr.com/ru/post/250551/.

58. Инструменты Kali Linux [Електронний ресурс] – Режим доступу: https://kali.tools.

59. Alyshov O. WAPP Веб безопасность [Електронний ресурс] / Orkhan Alyshov – Режим доступу: https://orkhanalyshov.com/blog/42.

60. Using Burp Proxy [Електронний ресурс] // 2018 – Режим доступу : https://support.portswigger.net/customer/portal/articles/1783119-usingburp-proxy.

61. Nessus professional benefits [Електронний ресурс]. – 2019. – Режим доступу: https://www.tenable.com/products/nessus/nessus-professional.

КОДЕКС АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ ХЕРСОНСЬКОГО ДЕРЖАВНОГО УНІВЕРСИТЕТУ

Я, Гетьман Владислав Вікторович, учасник освітнього процесу Херсонського державного університету, **УСВІДОМЛЮЮ**, що академічна доброчесність – це фундаментальна етична цінність усієї академічної спільноти світу.

ЗАЯВЛЯЮ, що у своїй освітній і науковій діяльності **ЗОБОВ'ЯЗУЮСЯ**:

– дотримуватися:

- вимог законодавства України та внутрішніх нормативних документів університету, зокрема Статуту Університету;
- принципів та правил академічної доброчесності;
- нульової толерантності до академічного плагіату;
- моральних норм та правил етичної поведінки;
- толерантного ставлення до інших;
- дотримуватися високого рівня культури спілкування;

– надавати згоду на:

- безпосередню перевірку курсових, кваліфікаційних робіт тощо на ознаки наявності академічного плагіату за допомогою спеціалізованих програмних продуктів;
- оброблення, збереження й розміщення кваліфікаційних робіт у відкритому доступі в інституційному репозитарії;
- використання робіт для перевірки на ознаки наявності академічного плагіату в інших роботах виключно з метою виявлення можливих ознак академічного плагіату;

– самостійно виконувати навчальні завдання, завдання поточного й підсумкового контролю результатів навчання;

– надавати достовірну інформацію щодо результатів власної навчальної (наукової, творчої) діяльності, використаних методик досліджень та джерел інформації;

– не використовувати результати досліджень інших авторів без використання покликань на їхню роботу;

– своєю діяльністю сприяти збереженню та примноженню традицій університету, формуванню його позитивного іміджу;

– не чинити правопорушень і не сприяти їхньому скоєнню іншими особами;

– підтримувати атмосферу довіри, взаємної відповідальності та співпраці в освітньому середовищі;

– поважати честь, гідність та особисту недоторканність особи, незважаючи на її стать, вік, матеріальний стан, соціальне становище, расову належність, релігійні й політичні переконання;

– не дискримінувати людей на підставі академічного статусу, а також за національною, расовою, статевою чи іншою належністю;

– відповідально ставитися до своїх обов'язків, вчасно та сумлінно виконувати необхідні навчальні та науково-дослідницькі завдання;

– запобігати виникненню у своїй діяльності конфлікту інтересів, зокрема не використовувати службових і родинних зв'язків з метою отримання нечесної переваги в навчальній, науковій і трудовій діяльності;

– не брати участі в будь-якій діяльності, пов'язаній із обманом, нечесністю, списуванням, фабрикацією;

– не підроблювати документи;

– не поширювати неправдиву та компрометуючу інформацію про інших здобувачів вищої освіти, викладачів і співробітників;

– не отримувати і не пропонувати винагород за несправедливе отримання будь-яких переваг або здійснення впливу на зміну отриманої академічної оцінки;

– не залякувати й не проявляти агресії та насильства проти інших, сексуальні домагання;

– не завдавати шкоди матеріальним цінностям, матеріально-технічній базі університету та особистій власності інших студентів та/або працівників;

– не використовувати без дозволу ректорату (деканату) символіки університету в заходах, не пов'язаних з діяльністю університету;

– не здійснювати і не заохочувати будь-яких спроб, спрямованих на те, щоб за допомогою нечесних і негідних методів досягати власних корисних цілей;

– не завдавати загрози власному здоров'ю або безпеці іншим студентам та/або працівникам.

УСВІДОМЛЮЮ, що відповідно до чинного законодавства у разі недотримання Кодексу академічної доброчесності буду нести академічну та/або інші види відповідальності й до мене можуть бути застосовані заходи дисциплінарного характеру за порушення принципів академічної доброчесності.

20.10.2021 _____ Гетьман Владислав
(дата) (підпис) Вікторович
(ім’я, прізвище)