РЕФЕРАТ

Розрахунково-пояснювальна записка складається з 9 розділів:

* вступ – в даному розділі описується сутність розробки, її актуальність. Складається з 1 сторінки;
* призначення, постановка задачі та огляд програмних аналогів і літератури – у цьому розділі описано призначення, постановка задачі, проведений огляд аналогів і літератури, а також аналіз існуючих систем. Складається з 15 сторінок;
* зовнішнє, логічне та внутрішнє проектування – у цьому розділі проведений огляд вхідних і вихідних даних, формалізація задачі, опис об’єктно-орієнтованого проектування, проектування інтерфейсу користувача, ескізи форм, проектування динаміки системи, вибір мови програмування, опис парадигм та інстрzументів програмування. Складається з 21 сторінки;
* реалізація спроектованого проекту – включає опис обраної архітектури програми, опис рівня доступу до даних, рівня бізнес-логіки, рівня користувацького інтерфейсу. Складається з 12 сторінок.
* тестування та налагодження програми– включає в себе вибір об’єкту та методів тестування, результати тестування та налагодження програми. Складається з 10 сторінок;
* безпека праці користувача при роботі на ЕОМ. Складається з 15 сторінок;
* висновки. Складається з 2 сторінок;
* список літератури – включає в себе бібліографічний список використаної літератури. Складається з 2 сторінок;
* додатки – містить технічне завдання і робочий проект.

Кількість таблиць: 14.

Кількість рисунків: 27.

ЗМІСТ

[Вступ 7](#_Toc453544370)

[1 Призначення, постановка задачі та огляд програмних аналогів та літератури 8](#_Toc453544371)

[1.1 Призначення та область застосування 8](#_Toc453544372)

[1.2 Постановка задачі 8](#_Toc453544373)

[1.3 Огляд програмних аналогів 9](#_Toc453544374)

[1.3.1 Огляд програмного продукту веб-клієнт єдебо 9](#_Toc453544375)

[1.3.2 Огляд програмного продукту Аверс: модуль «Приёмная комиссия» 10](#_Toc453544376)

[1.3.3 Огляд програмного продукту Magellan: модуль «Приёмная   
комиссия» 11](#_Toc453544377)

[1.3.4 Огляд веб-додатку «Вступ» 13](#_Toc453544378)

[1.4 Огляд літератури 14](#_Toc453544379)

[1.3.2 Використання бази єдебо 17](#_Toc453544380)

[1.3.3 Проектування REST серверу 18](#_Toc453544381)

[1.3.4 Багаторівнева архітектура додатку 20](#_Toc453544382)

[2 Зовнішне, логічне та внутрішнє проектування 23](#_Toc453544383)

[2.1 Зовнішнє проектування 23](#_Toc453544384)

[2.1.1 Вхідні данні 23](#_Toc453544385)

[2.1.2 Вихідні дані 23](#_Toc453544386)

[2.1.3 Формалізація задачі 24](#_Toc453544387)

[2.2 Внутрішнє проектування 29](#_Toc453544388)

[2.2.1 Вибір мови програмування та середовища розробки 29](#_Toc453544389)

[2.2.2 Вибір інструментів та парадигм програмування 33](#_Toc453544390)

[2.2.3 Визначення призначень об’єктів за допомогою карток crc 38](#_Toc453544391)

[2.2.4 Проектування інтерфейсу користувача 41](#_Toc453544392)

[3 Реалізація спроектованого проекту 44](#_Toc453544393)

[3.1 Рівень доступу до даних 45](#_Toc453544394)

[3.2 Рівень бізнес-логіки 47](#_Toc453544395)

[3.3 Рівень користувацького інтерфейсу 52](#_Toc453544396)

[4 Тестування та налагодження програми 55](#_Toc453544397)

[4.1 Тестування методом «білого ящика» 55](#_Toc453544398)

[4.2 Тестування методом «чорного ящика» 60](#_Toc453544399)

[4.3 Налагодження програми 62](#_Toc453544400)

[5 Охорона праці 65](#_Toc453544401)

[5.1 Аналіз шкідливих та небезпечних виробничих факторів 65](#_Toc453544402)

[5.2 Організаційні і технічні заходи щодо облаштування робочого місця. 67](#_Toc453544403)

[5.2.1 Вимоги до розташування робочих місць 67](#_Toc453544404)

[5.2.2 Вимоги до організації робочого місця 68](#_Toc453544405)

[5.3 Небезпеки пов’язані з шумом та освітленням робочої зони 69](#_Toc453544406)

[5.3.1 Шум, пов'язаний з відеодисплейним терміналом 69](#_Toc453544407)

[5.3.2 Параметри освітлення робочого місця та робочого приміщення 71](#_Toc453544408)

[5.4 Використання принтерів 74](#_Toc453544409)

[5.5 Дії при аварійних ситуаціях 76](#_Toc453544410)

[Висновки 80](#_Toc453544411)

[Література 82](#_Toc453544412)

[Додатки 84](#_Toc453544413)

ВСТУП

Під час вступної кампанії секретарям приймальної комісії та адміністрації університету необхідні звіти, що відображають у зручній формі статистику поданих заяв абітурієнтами до ВУЗів. Проект призначений саме для надання таких звітів для ректора університету. У звітах буде відображено кількість абітурієнтів денного або заочного відділення, що подали заяви до факультету, кількість оригіналів документів, бюджетних та контрактних місць, інформація про пільги абітурієнтів, пріоритетність заяв та загальну кількість поданих заяв за день.

Програмний продукт «Робоче місце ректора для аналізу статистики контингенту абітурієнтів на основі даних ЄДЕБО» є складовою частиною проекту для університету. Друга частина проекту призначена для роботи деканів [1].

Розробка та впровадження програмного продукту, що автоматично складає звіти без впливу людського фактору, дозволить значно спростити однорідну працю секретарів приймальної комісії та скоротити час на складання звітів для ректора університету.

Під час вивчення аналогів, було виявлено, що жоден з них не може представляти звіти у тому вигляді, який потрібен ректору ДНУЗТ. Також аналоги являються занадто дорогими комерційними проектами. Таким чином одними з підстав для розробки стала відсутність аналогів та висока вартість впровадження існуючого ПО.

1 ПРИЗНАЧЕННЯ, ПОСТАНОВКА ЗАДАЧІ ТА ОГЛЯД ПРОГРАМНИХ АНАЛОГІВ ТА ЛІТЕРАТУРИ

1.1 Призначення та область застосування

Областю застосування системи є область освіти. Робоче місце ректора призначене для доступу до автоматично згенерованих звітів для ректора університету, що відображають статистику поданих заяв на кожен факультет. Секретарям приймальної комісії надається можливість автоматизованого створення звітів у вигляді таблиці, що відображає кількість поданих заяв на денне та заочне відділення. Автоматичне або ручне оновлення даних з ЄДЕБО надає можливість отримувати актуальні дані у будь-який час.

Основними користувачами програмного продукту є секретарі приймальної комісії та ректор університету.

1.2 Постановка задачі

Необхідно розробити програмний продукт для автоматичного створення звітів, що відображають дані по вступній кампанії, для ректора університету. До функціональних характеристик програмного продукту висунуті наступні вимоги:

* програма повинна надавати розмежування прав доступу для ректора та секретарів приймальної комісії;
* при вході на сайт у ролі ректора програма повинна надавати звіти по всьому університету та по кожному факультету окремо на вибір;
* програма повинна користувачеві надавати доступ до сховища звітів, в якому зберігаються усі звіти за минулий час;
* при вході на сайт у ролі секретаря програма повинна надавати доступ до всіх типів звітів, які реалізовано на сайті;
* необхідні звіти по контингенту абітурієнтів повинні подаватися в узгодженому з секретарями приймальної комісії форматі;
* можливість роздрукування повинна бути реалізована для кожного типу звітів;
* програма повинна надавати час останнього оновлення даних локального сховища даних через ЄДЕБО;
* програма повинна надавати користувачу можливість оновити дані з ЄДЕБО у будь-який час.

1.3 Огляд програмних аналогів

Пошук та аналіз доступних програмних аналогів дозволив виявити та врахувати недоліки та переваги існуючих програмних продуктів для створення кращого автоматизованого робочого місця ректора університету для аналізу контингенту абітурієнтів з використанням даних ЄДЕБО.

1.3.1 Огляд програмного продукту веб-клієнт ЄДЕБО

Єдина державна електронна база з питань освіти (ЄДЕБО) – автоматизована система накопичення, оброблення, зберігання та захисту даних, у тому числі персональних, щодо закладів здійснюючих освітні послуги в Україні. Веб-клієнт дозволяє взаємодіяти з цією базою. Саме база ЄДЕБО буде використовуватися для отримання актуальної інформації про абітурієнтів для подальшого аналізу та створення звітів.

Вивчення інструкції, створеної для роботи з веб-додатком, можна виділити наступний функціонал цього програмного продукту[2]:

* подання електронних заяв в ВНЗ та їх подальший розгляд Міністерством освіти і науки України;
* внесення необхідної інформації для кожного вищого навчального закладу;
* формування рейтингів ВНЗ та абітурієнтів;
* формування наказу про зарахування абітурієнтів до ВНЗ та додатку до наказу;
* отримання службових повідомлень від веб-клієнту;
* верифікації даних як про ВНЗ так і про абітурієнтів;
* перенесення необхідних даних з загальної бази.

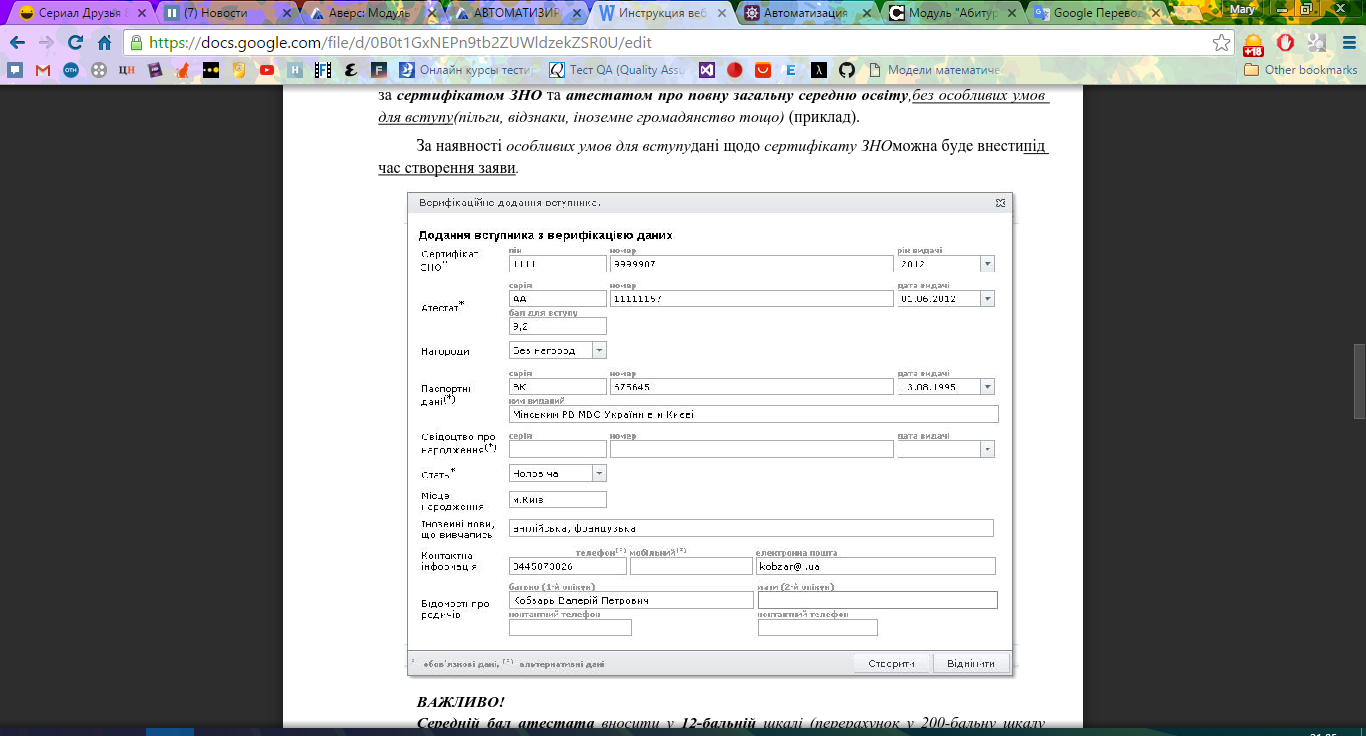
Веб-додаток ЄДЕБО має складний інтерфейс з великою кількістю вікон та полів для заповнення (рис. 1.1). Ця система забезпечує широкі можливості при роботі з внесеними у базу даними та є безкоштовною. Тим не менш веб-додаток не може забезпечити складання звітів необхідної форми для ректора ДНУЗТ.

Рисунок 1.1 – Приклад додавання даних абітурієнта до системи

1.3.2 Огляд програмного продукту АВЕРС: модуль «Приёмная комиссия»

Модуль «Приёмная комиссия» призначений для автоматизації всього циклу завдань приймальної комісії: реєстрація даних абітурієнтів, формування списків бажаючих вступити на обрану спеціальність, плану прийому, списків зарахованих, а також надання звітності за підсумками роботи [3].

Система вирішує має наступний функціонал:

* реєстрація анкетних даних та заяв абітурієнтів;
* самореєстрація абітурієнта на сайті освітньої організації;
* формування необхідного пакету документів для абітурієнта (розписка, заява, анкета і т.д.);
* затвердження списку дисциплін, за якими передбачені вступні випробування для кожної конкурсної групи;
* формування списків, рекомендованих до зарахування за середнім балом атестата (диплома) і / або вступних випробувань з урахуванням наявності у абітурієнта пільг під час вступу;
* отримання щоденної інформації про кількість поданих заяв та конкурсі прийому;
* ведення архіву абітурієнтів:
* повторна подача заяви на іншу спеціальність без дублювання введення даних.

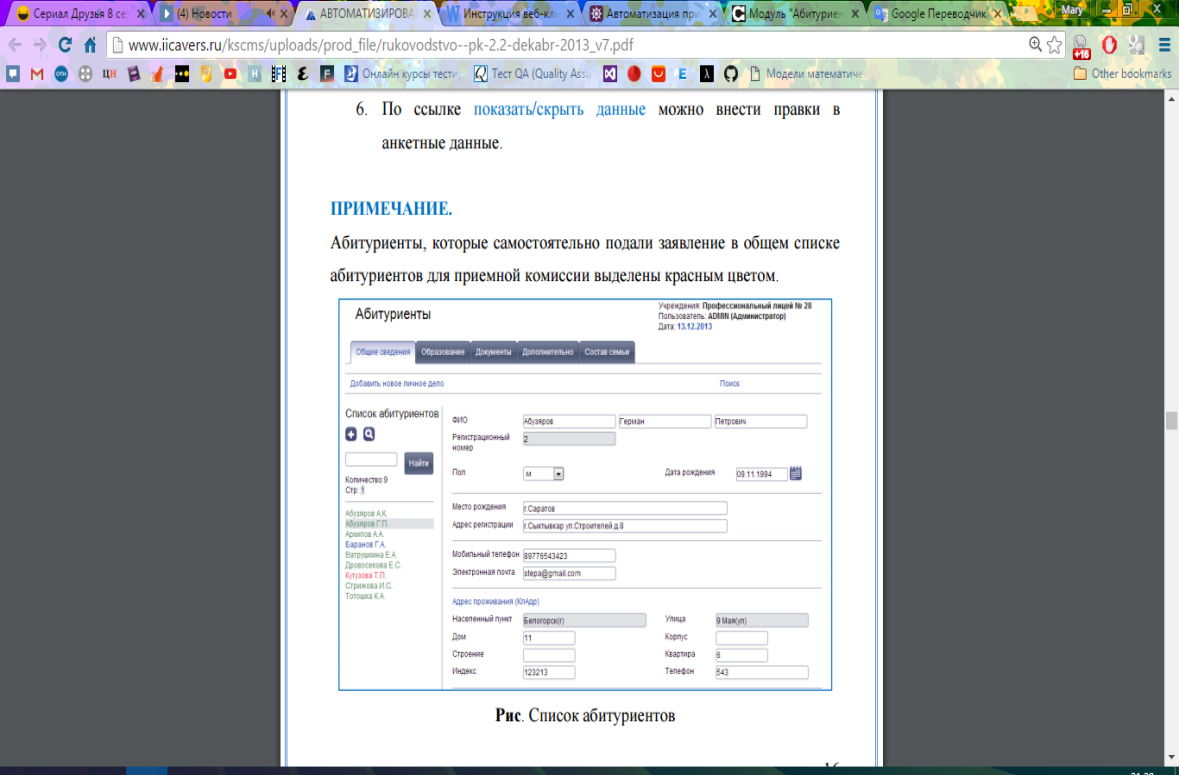
Модуль «Приёмная комиссия» має зручний та простий інтерфейс (рис. 1.2), але не має синхронизації з загальною державною базою абітурієнтів і всі дані вводяться вручну. Система використовується у Росії і є комерційним продуктом.

Рисунок 1.2 – Приклад додавання абітурієнта

1.3.3 Огляд програмного продукту Magellan: модуль «Приёмная комиссия»

Модуль «Приёмная комиссия» прискорює прийом документів від абітурієнтів, знижує кількість помилок при реєстрації заяв, формує складні аналітичні звіти. Модуль відповідає вимогам законодавства 2016 року в області проведено приймальної кампанії у Росії.

За допомогою модуля вирішені наступні завдання [4]:

* створення, редагування і зберігання особистих справи вступників;
* роздрукування повного пакета документів для вступника: особиста справа, заява, розписка, договір і так далі;
* формування щоденного звіту з прийому, що дозволяє контролювати хід приймальної кампанії в режимі реального часу;
* формування різних аналітичних звітів, в тому числі на основі аналізу результатів здачі вступних випробувань;
* отримання актуальних даних з єдиної бази абітурієнтів.

Використовуючи WEB-версію модуля, можливо приймати документи від вступників в електронному вигляді (через інтернет), організувати для вступників особистий кабінет для контролю ходу приймальної кампанії. WEB-версія модуля може використовуватися як на стаціонарному комп'ютері і ноутбуку, так і на планшеті, що є надзвичайно зручним.

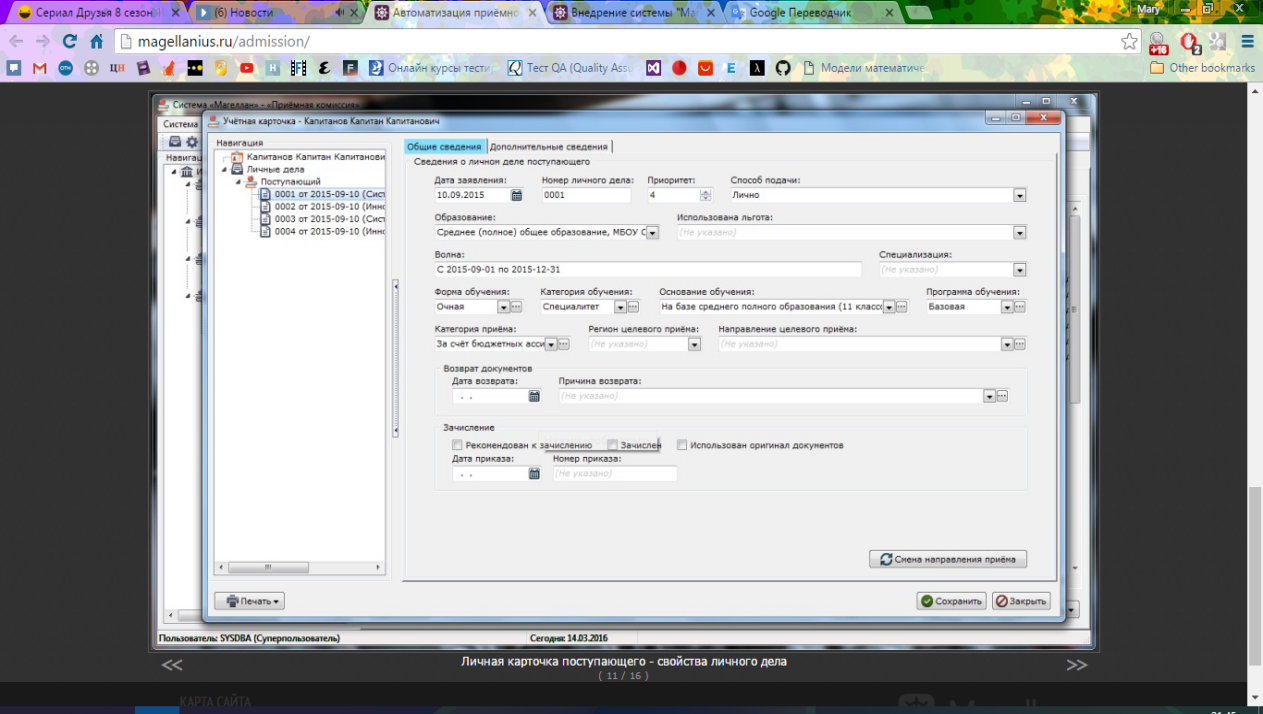
Інтерфейс програмного продукту складний, орієнтуватися звичайному користувачеві важко (рис 1.3). Модуль використовується на території Росії і є комерційним продуктом. Розробник гарантує повну підтримку продукту та навчання персоналу. Не зважаючи на запропоновану велику кількість варіантів звітів жоден з них не задовольняє вимогам ректора ДНУЗТ.

Рисунок 1.3 – Приклад додавання інформації про абітурієнта

1.3.4 Огляд веб-додатку «Вступ»

Веб-додаток «Вступ» було розроблено в рамках дипломного проекту студентами ДНУЗТ у 2015 році [5]. Розробка цього програмного продукту проводилася опираючись на вимоги приймальної комісії минулого року.

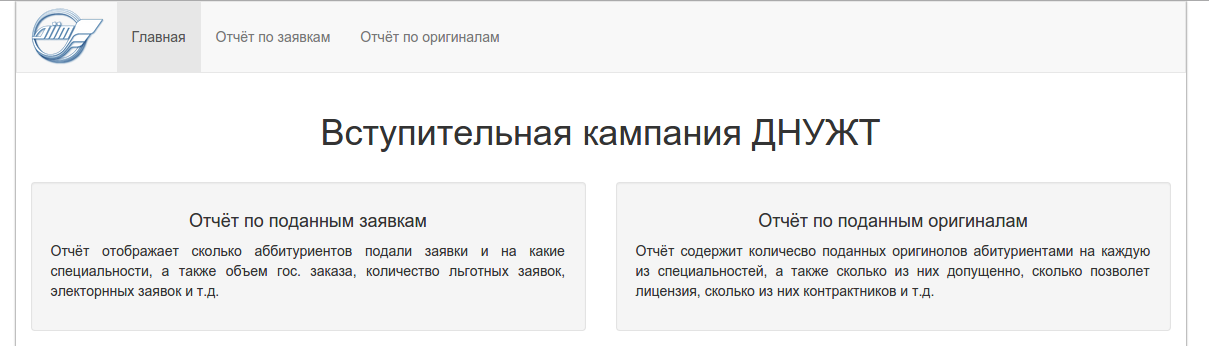
Програмний продукт має зрозумілий користувачеві з будь-яким рівнем роботи з ПК інтерфейс (рис.1.4). Було реалізовано лише необхідний функціонал, тобто перегляд звітів з актуальними даними та створення статистики.

Рисунок 1.4 – Головна сторінка веб-додатку «Вступ»

Готовий веб-клієнт дозволяє:

* переглядати звіти щодо подачі заявок на вступ в режимі онлайн;
* скласти та роздруковувати звіти у необхідній формі;
* отримувати актуальну інформацію з ЄДЕБО;
* завантажити електронну версію звіту;
* переглядати статистику і діаграми щодо проходження вступної кампанії.

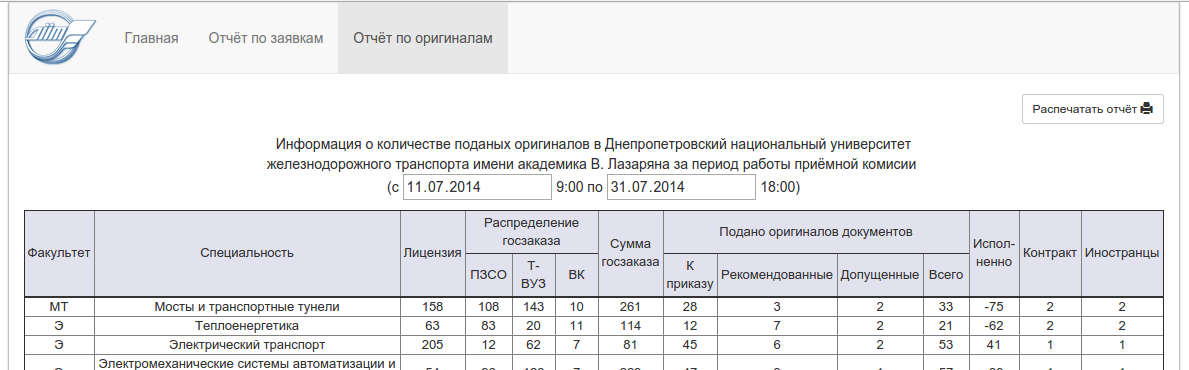
Створення нового програмного продукту було зумовлене зміненням правил вступу 2016 року, що спричинило неактуальність створених форм звітів 2015 року (рис 1.5).

Рисунок 1.5 – Приклад звіту по поданим оригіналам документів

* 1. Огляд літератури

Під час виконання аналізу літератури для розробки продукту, були розглянуті наступні питання: ЄДЕБО, шаблон MVC, та проектування REST сервера.

1.4.1 Основи концепції MVC

MVC – це архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення [6].

Цей шаблон поділяє систему на три частини: модель даних, вигляд даних та керування. Застосовується для відокремлення даних (модель) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача (рис. 1.6).

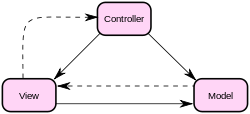


Рисунок 1.6 – Графічне представлення шаблону MVC

Мета шаблону – гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми.

У 2009 році Microsoft випустив каркас ASP.NET MVC з відкритими кодами, що дозволило цьому шаблону отримати неабияке поширення серед .NET розробників.

Однією з переваг цього шаблону є підтримка декількох представлень. Оскільки представлення відокремлене від моделі і не існує пряма залежність моделі від представлення, призначений для користувача інтерфейс може відображати кілька подань одних і тих же даних одночасно. Наприклад, кілька сторінок в веб-додатку можуть використовувати одні і ті ж об'єктні моделі. Інший приклад являє собою веб-додаток, що дозволяє користувачеві змінювати зовнішній вигляд сторінок. Ці сторінки відображають ті ж дані з однієї моделі, але показують їх по-іншому [6].

Вимоги до інтерфейсу користувача, як правило, змінюються швидше, ніж бізнес-модель. Користувачі можуть віддавати перевагу різним кольорам, шрифтам, рівням підтримки нових пристроїв, таких як мобільні телефони або КПК. Оскільки модель не залежить від представлень, додавання нових типів представлень до системи, як правило, не впливає на модель. В результаті, обсяг змін обмежується змінами представлень даних.

Гнучкість та незалежність рівнів проектування є надзвичайно корисною перевагою шаблону MVC, адже зміни в правилах вступною кампанії та формі необхідних звітів є доволі частим явищем, тому незалежність слою даних та їх представлення не робить ці зміни проблемою.

Модель

[Моделі](http://www.yiiframework.com/doc/guide/1.1/uk/basics.model) представляють структури даних, які використовуються у додатку, і часто є спільними для кількох піддодатків. Тому моделі:

* повинні містити властивості, які представляють конкретні дані;
* повинні включати у себе бізнес-логіку (наприклад, правила валідації) для того, щоб переконатися, що дані відповідають вимогам;
* можуть містити код для роботи із даними.

Іноді слідування останньому перерахованому правилу робить модель дуже товстою, тобто вона містить дуже багато коду в одному класі. Це може призвести до труднощів підтримки коду у тому випадку, якщо модель використовується декількома способами. Для невеликих та середніх додатків це припустимо.

Представлення

[Представлення](http://www.yiiframework.com/doc/guide/1.1/uk/basics.view) відповідають за відображення моделей у потрібному користувачу форматі. У загальному випадку представлення:

* повинні, головним чином, містити розмітку, таку як HTML, або простий код, який використовується для обходу, форматування та відображення даних;
* не повинні безпосередньо звертатися до бази даних. Цим повинен займатися DAL;
* не повинні безпосередньо звертатися до змінних, які отримуються із запиту користувача. Цю задачу повинен виконувати контролер. Представлення повинно використовуватися для задання зовнішнього вигляду даних, отриманих із контролера та моделі;
* може безпосередньо звертатися до властивостей та методів контролера або моделей. При цьому дані властивості та методи повинні відповідати за відображення.

Контролер

[Контролери](http://www.yiiframework.com/doc/guide/1.1/uk/basics.controller) — сполучна ланка, яка поєднує моделі, представлення та інші компоненти у робочий додаток. Контролер відповідає за обробку запитів користувача. Тому контролер:

* може звертатися до змінних, які отримуються із запиту користувача;
* може створювати екземпляри моделей та керувати ними. Приміром, у типової дії оновлення моделі контролер може спочатку створити екземпляр моделі, потім заповнити його даними із  та, після вдалого збереження моделі, перенаправити браузер користувача на сторінку створеної моделі. Слід відмітити, що саме зберігання моделі повинно бути реалізоване у моделі, а не в контролері;
* не повинен містити SQL-запитів. Їх краще тримати у DAL;
* не повинен містити HTML та іншої розмітки. Її варто винести у представлення.

У добре спроектованому MVC-додатку моделі зазвичай дуже тонкі та містять тільки поля для зберігання даних, ніяких методів. У той же час, контролери дуже товсті та містять більшу частину коду.

1.3.2 Використання бази ЄДЕБО

На теперішній час ЄДЕБО вважається єдиною державною базою даних, що містить достовірну та актуальну інформацію щодо даних абітурієнтів, студентів та наукової діяльності в ВНЗ України.

Державне підприємство «Інфоресурс» належить до сфери управління Міністерства освіти і науки України. Це підприємство є адміністратором Єдиної державної електронної бази з питань освіти (ЄДЕБО).

ЄДЕБО – автоматизована система збирання, сертифікації, оброблення, зберігання та захисту даних, у тому числі персональних, щодо давачів та отримувачів освітніх послуг в Україні [2].

Використовуючи дані бази можна дистанційно подати документи в будь-який вищий навчальний заклад України. Під час подачі заяви на участь у вступній кампанії, абітурієнт має змогу побачити своє місце в загальному рейтингу серед результатів інших вступників. Також, зменшується кількість питань щодо правдивості поданої інформації. Достатньо ввести ідентифікаційні дані абітурієнта, щоб побачити інформацію про нього, історію його навчання, вступів до будь-яких навчальних закладів України та (або) інших країн, заяв на участь у вступній кампанії і таке інше. Також зводяться до мінімального рівня маніпуляції із зарахуванням абітурієнтів, адже інформація вступників відкрита для всіх.

Дані з цієї бази використовуються під час виготовлення документів. Напиклад, про освіту державного зразка, про вчені звання та наукові ступені, ліцензій на надання освітніх послуг особам та сертифікатів про акредитацію та для отримання інших відомостей інформаційного характеру та обробки статистичних даних в інтересах навчальних закладів, установ освіти для здійснення контролю та прийняття управлінських рішень.

На теперішній час ЄДЕБО вважається єдиною державною базою даних, що містить достовірну та актуальну інформацію щодо даних абітурієнтів, студентів та наукової діяльності в ВНЗ України.

1.3.3 Проектування REST серверу

В основі роботи будь-якого веб-додатку лежать поняття "клієнт" і "сервер". Клієнт - це та частина додатку, яка відображає користувачеві інтерфейс, наприклад, просту сторінку сайту. Сервер - це та частина додатку, яка готує інформацію для відображення. В основі роботи звичайного сайту, навіть написаного за допомогою простого HTML, також лежать ці принципи.

З моменту появи інтернету технології почали стрімко розвиватися. З часом з'являються нові підходи. І в 2000 році Рой Філдінг розробив і описав у своїй статті підхід REST.

REST — підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів [7].

Дані повинні передаватися у вигляді невеликої кількості стандартних форматів (наприклад HTML, XML, JSON). Мережевий протокол (як і HTTP) повинен підтримувати кешування, не повинен залежати від мережевого прошарку, не повинен зберігати інформацію про стан між парами «запит-відповідь». Стверджується, що такий підхід забезпечує масштабування системи і дозволяє їй еволюціонувати з новими вимогами.

Текстовий формат обміну даними між комп'ютерами JSON може бути з легкістю прочитаний людиною та часто використовується під час асинхронної передачі структурованої інформації між клієнтом та сервером (технологія REST). Формат дозволяє описувати об'єкти та інші структури даних [8].

JSON знайшов своє головне призначення у написанні веб-програм, а саме при використанні технології AJAX.

Використання AJAX (Asynchronous JavaScript and XML) у роботі сайту стало надзвичайно поширеним явищем, що свідчить про його інноваційність і відповідність сучасним тенденціям. AJAX – це підхід до веб-розробки, який дає можливість веб-сторінці довантажувати необхідну інформацію без перезавантаження самої сторінки (рис. 1.7).

AJAX почали використовувати у 2005 році. Він не є окремою технологією, а концепцією використання декількох технологій, що існували і раніше. Переваги AJAX очевидні: підвищення інтерактивності і динамічності веб-сторінок за рахунок зменшення об’єму інформації, що завантажується; зменшення навантаження на сервер, що також дуже важливо, враховуючи постійне зростання потоків інформації в мережі «Інтернет». Крім того, AJAX забезпечує покращення функціональності сайту.

Через запит до сервера генерується сторінка, яку буде бачити користувач. Запити користувача будуть звертатися до AJAX-модулю, який забезпечує роботу з сервером через динамічні звернення.

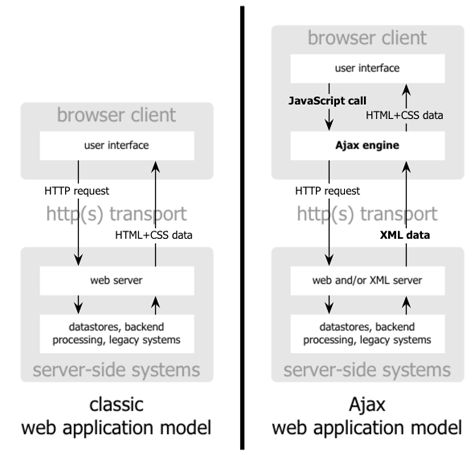


Рисунок 1.7– Схема роботи AJAX

Найбільш яскравим і відомим прикладом використання AJAX можна назвати Google Maps. Взагалі, використання AJAX на сайтах Google підтвердило його ефективність. Активно використовується AJAX в різноманітних форумах, чатах, соціальних мережах.

До основних недоліків AJAX слід віднести безпеку (можливість прочитати вихідний код у браузері), неможливість реєстрації браузерами в історії відвідування сторінок (не працюватиме кнопка «Назад»), проблеми індексації пошуковими системами (динамічно завантажений контент недоступний для пошукових роботів). Тому доцільно використовувати AJAX тільки для окремих частин контенту сайту.

1.3.4 Багаторівнева архітектура додатку

Один з підходів до розробки веб-додатків, створений щоб зосередитися на чітко визначених рівнях архітектури додатку. Цей підхід аналогічний тому, як архітектор проектує будівлю.

За допомогою багаторівневої архітектури, фахівці можуть спроектувати і розробити «поверхи» – так звані рівні незалежно один від одного, за умови, що з'єднання між рівнями (інтерфейси) ретельно продумане.

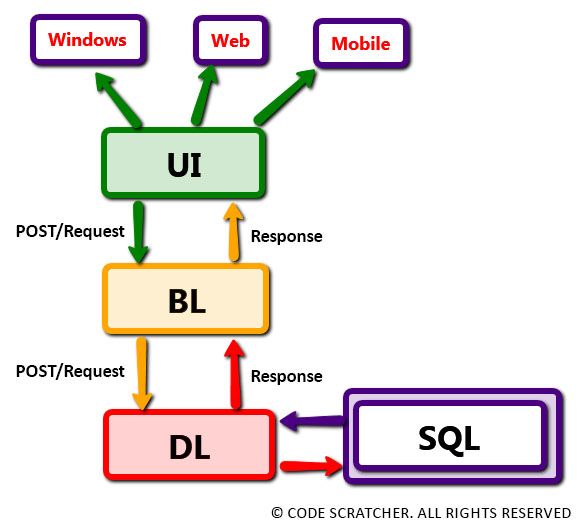
Рівні повинні бути незалежні один від одного, наскільки це можливо. Кожен шар повинен мати чітко визначену спрямованість. Для того, щоб проектувати шари правильно, необхідно ні чітко обумовлювати завдання і обов'язки кожного шару. Рівні повинні займатися лише своєю справою. Якщо один рівень відповідає за взаємодію з користувачем, тільки цей рівень може взаємодіяти з користувачем. Інші рівні, яким необхідно отримати інформацію від користувача, повинні робити це через інтерфейс.

Чітко визначені протоколи повинні бути налаштовані для рівнів, щоб взаємодіяти один з одним. Взаємодія між шарами відбувається тільки через ці протоколи. Шари не прив’язані безпосередньо до будь-якої конкретної області застосування. Наприклад, архітектура може однаково добре працювати для онлайн-систему замовлень і інтернет-форумів.

Є кілька загальних підходів до архітектури додатків, які змінюються в залежності від кількості використовуваних рівнями. Одна загальна схема розбити додаток на три рівні (рис. 1.8):

* Presentation Layer: цей рівень обробляє призначений для користувача інтерфейс;
* Business Rules Layer: цей рівень обробляє бізнес-правила;
* Data Access Layer: базова модель бази даних, яку використовує програма.

Створення окремого рівня для бізнес-логіки дозволяє відокремити правила від проектування баз даних і логіки подання. Бізнес-логіка може бути змінена без шкоди для інших слоїв додатку.

Рисунок 1.8 – Триривнева архітектура

Можна виділити такі переваги трирівневої архітектури [9]:

* масштабованість;
* конфігурованість — ізольованість рівнів один від одного дозволяє (при правильному розгортанні архітектури) швидко і простими засобами переконфігурувати систему при виникненні збоїв або при плановому обслуговуванні на одному з рівнів;
* високий рівень безпеки;
* висока надійність;
* низькі вимоги до швидкості каналу (мережі) між терміналами і сервером застосунків;
* низькі вимоги до продуктивності і технічних характеристик терміналів, як наслідок зниження їхньої вартості. Терміналом може виступати не тільки комп'ютер, але і, наприклад, мобільний телефон.

Недоліки випливають з переваг. Можна виділити наступні недоліки трирівневої архітектури:

* вища складність створення застосунків;
* складніша у розгортанні і адмініструванні;
* високі вимоги до продуктивності серверів застосунків і сервера бази даних, а, отже, і висока вартість серверного обладнання;
* високі вимоги до швидкості каналу (мережі) між сервером бази даних і серверами застосунків.

2 ЗОВНІШНЕ, ЛОГІЧНЕ ТА ВНУТРІШНЄ ПРОЕКТУВАННЯ

2.1 Зовнішнє проектування

2.1.1 Вхідні данні

Одним з видів вхідних даних, що використовує програмний продукт є логін та пароль користувача.

Також до складу вхідних даних належить інформація про:

* ВНЗ (назва, номер факультету);
* факультети (назва, номер спеціальності);
* спеціальності (назва, кількість бюджетних місць, загальна кількість місць);
* заяву абітурієнта (номер заяви, пріоритетність вступу, назва факультету, заочне або денне відділення, контракт або бюджет);
* інтервал дат для відображення статистики у звіті.

Усі дані текстового та числового формату представлені в табличному вигляді та отримуються з ЄДЕБО.

2.1.2 Вихідні дані

Вихідними даними є автоматично сформовані звіти для ректора у HTML форматі (рис. 2.1), що відображується у веб-браузері або роздруковується на принтері.

У звітах відображається статистика поданих заяв на спеціальності всіх або обраних факультетів.

Рисунок 2.1 – Макет звіту для ректора факультет

Зокрема звіт містить наступну інформацію:

* «Ліцензія» – кількість місць на спеціальності факультету, виділених Міністерством освіти і науки України;
* «Держ. замовлення» – кількість мінімально припустимих та максимально можливих бюджетних місць;
* «Поза конкурсом» – кількість заяв абітурієнтів, які розглядаються поза конкурсом;
* «З пріоритетом» – розподілення заяв абітурієнтів за пріоритетом вступу на спеціальність;
* «Разом» – загальна кількість поданих заяв на поточний час;
* «Рекомендовано за держ. замовленням» – кількість заяв, рекомендованих системою ЄДЕБО наприкінці вступної кампанії;
* «Оригінали Держ. замовл.» та «Оригінали Контракт» – кількість оригіналів документів, принесених абітурієнтами після оприлюднення системою ЄДЕБО рекомендованих до зарахування абітурієнтів.

2.1.3 Формалізація задачі

Формалізація поставленої задачі на рівні зовнішнього проектування представлена у вигляді діаграми варіантів використання (рис. 2.2), послідовностей (рис. 2.3),блок-схеми роботи програми (рис. 2.4) та діаграми класів (рис. 2.5).

У ролі актора у діаграмі варіантів використання виступає потенційний користувач системи, це може бути працівник приймальної комісії або ректор університету. Актор може виконувати наступні дії:

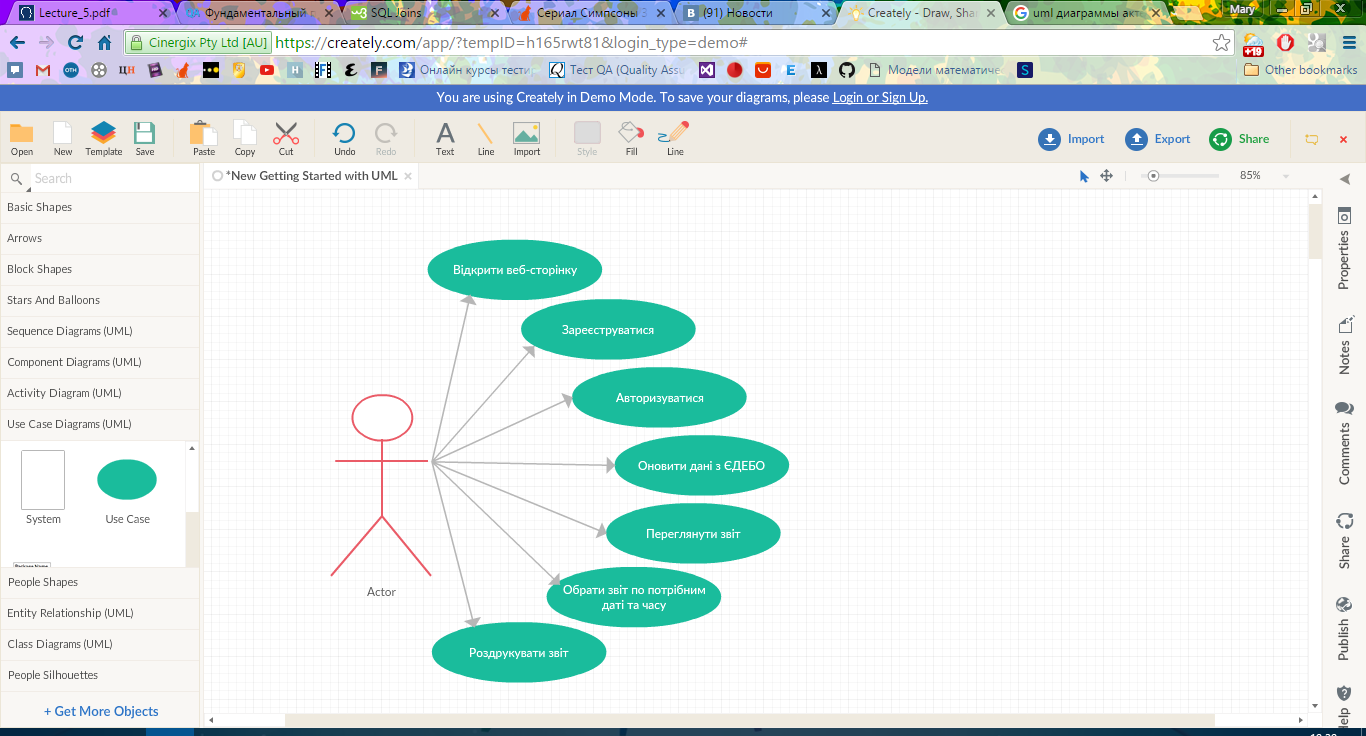
* відкрити веб-сторінку;
* зареєструватися;
* авторизуватися;
* оновити дані з ЄДЕБО;
* обрати звіт по потрібним даті та часу;
* обрати звіт по змісту;
* роздрукувати потрібний звіт;
* переглянути обраний звіт.

Рисунок 2.2 – Діаграма варіантів використання

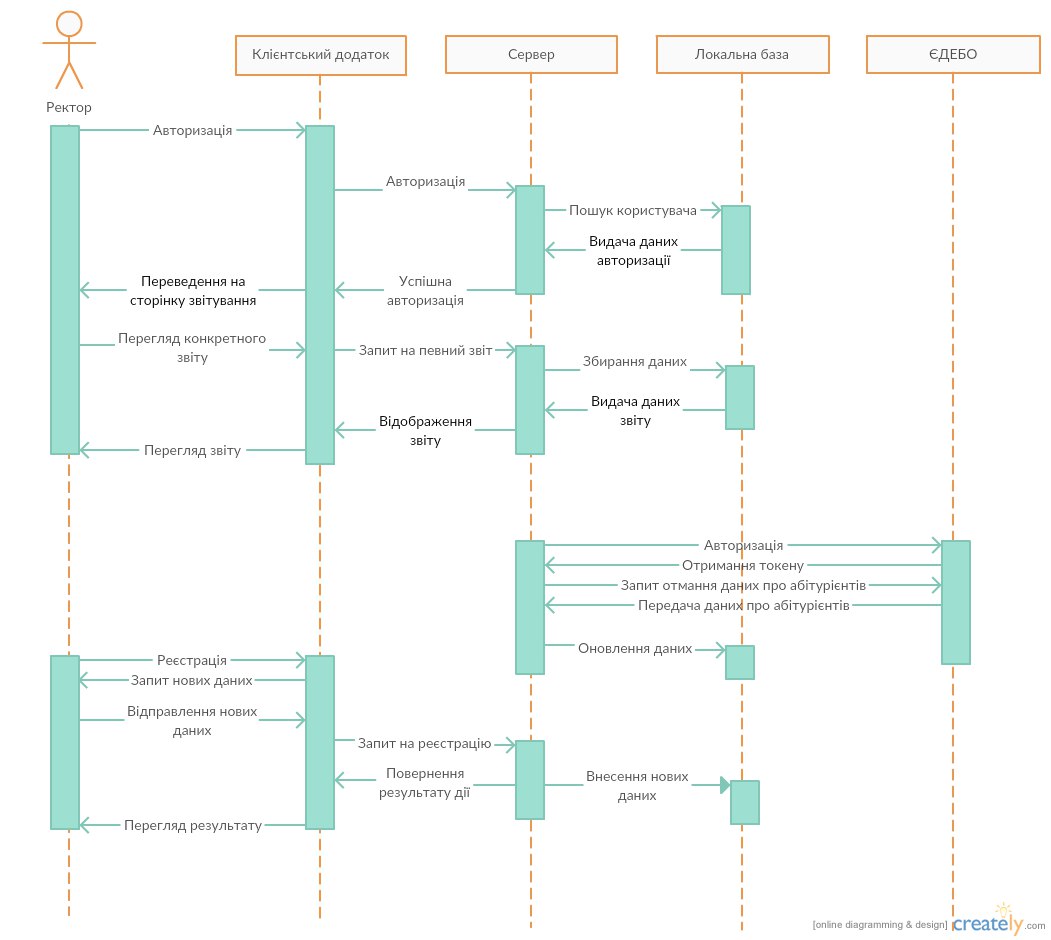
Для розгляду взаємодії об’єктів у часі була спроектована діаграма послідовностей. Також ця діаграма демонструє особливості передачі і прийому повідомлень між об'єктами.

У представленій діаграмі послідовностей актор – це також ректор або секретар приймальної комісії. В якості об’єктів у діаграмі виступають:

* клієнтський додаток;
* сервер;
* локальна база, тобто сховище даних, що зберігає необхідні дані з ЄДЕБО;
* база ЄДЕБО.

Ця діаграма дозволяє прослідити взаємодію між об’єктами системи, лінію життя об’єктів, обмін повідомленнями між об’єктами, дозволяє краще зрозуміти і спроектувати необхідну систему.

Розроблена діаграма послідовностей представлена на рис. 2.3

Рисунок 2.3 – Діаграма послідовностей

Для представлення у графічному вигляді основного алгоритма роботи була спроектована блок-схема. Саме представлення алгоритму програми у графічному вигляді дозволило виявити недоліки первинного проектування та покращити цей алгоритм.

Блок-схема відображає такі основні блоки:

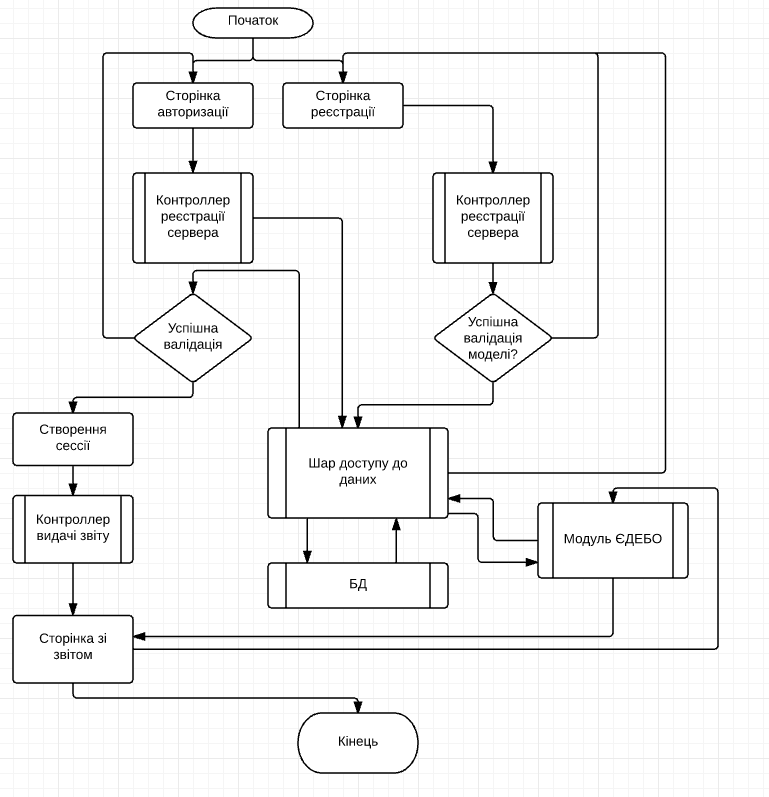
* авторизація;
* реєстрація;
* контролер реєстрації сервера;
* перевірка успішності валідації моделі;
* створення сесії;
* контролер видачі звіту;
* шар доступу до даних;
* БД;
* модуль ЄДЕБО.

Рисунок 2.4 – Блок-схема роботи програми

Під час проектування для виконання поставленої задачі проект було розділено на класи. Діаграма класів є однією з форм статичного опису системи з точки зору її проектування, показуючи її структуру. Діаграма класів не відображує динамічну поведінку об'єктів зображених на ній класів. На діаграмах класів показуються класи, інтерфейси і відносини між ними.

Кожен клас має назву, атрибути і операції. Клас на діаграмі показується у вигляді прямокутника, розділеного на три області. У верхній міститься назва класу, в середній – опис атрибутів (властивостей), в нижній – назви операцій.

Атрибути класу визначають склад і структуру даних, що зберігаються в об'єктах цього класу. Кожен атрибут має ім'я і тип, що визначає, які дані він представляє. Кожна операція має сигнатуру, яка містить ім'я операції, тип значення і список параметрів, який може бути порожнім.

Проектування діаграми класів дозволило виділити основні класи, потрібні для реалізації робочого місця декана та визначити відносини між класами та їх основні операції (методи).

Таким чином було виділено наступні класи, необхідні для проекту:

* клас ReportsController – контролер, відповідальний за прийняття запитів на повернення звітів;
* клас AuthSession – містить логіку та інформацію для окремої сесії;
* клас AuthSessions – зберігає, створює, видаляє та оброблює сесії;
* клас ForRectorViewModel – модель для відправлення даних для звіту ректора;
* клас IGenericRepository – містить абстракцію для доступу до БД;
* клас GenericRepository – містить реалізацю абстракції доступу до БД;
* клас GenericStorageRepository – надає можливість переглядати, доповнювати, редагувати та видаляти інформацію, що міститься у сховищі даних;
* клас GenericRoleRepository – надає можливість переглядати, доповнювати, редагувати та видаляти інформацію про ролі користувачів;
* клас GenericUserRepository – надає можливість переглядати, доповнювати, редагувати та видаляти інформацію про користувачів;
* клас Role – містить модель даних для ролей користувачів локальної бази;
* клас AuthController.cs – контролер, відповідальний за прийняття запитів на повернення звітів та списку історії звітів;
* клас StorageModel.cs – містить модель сховища локальної бази;
* клас User – містить модель даних для користувачів локальної бази.

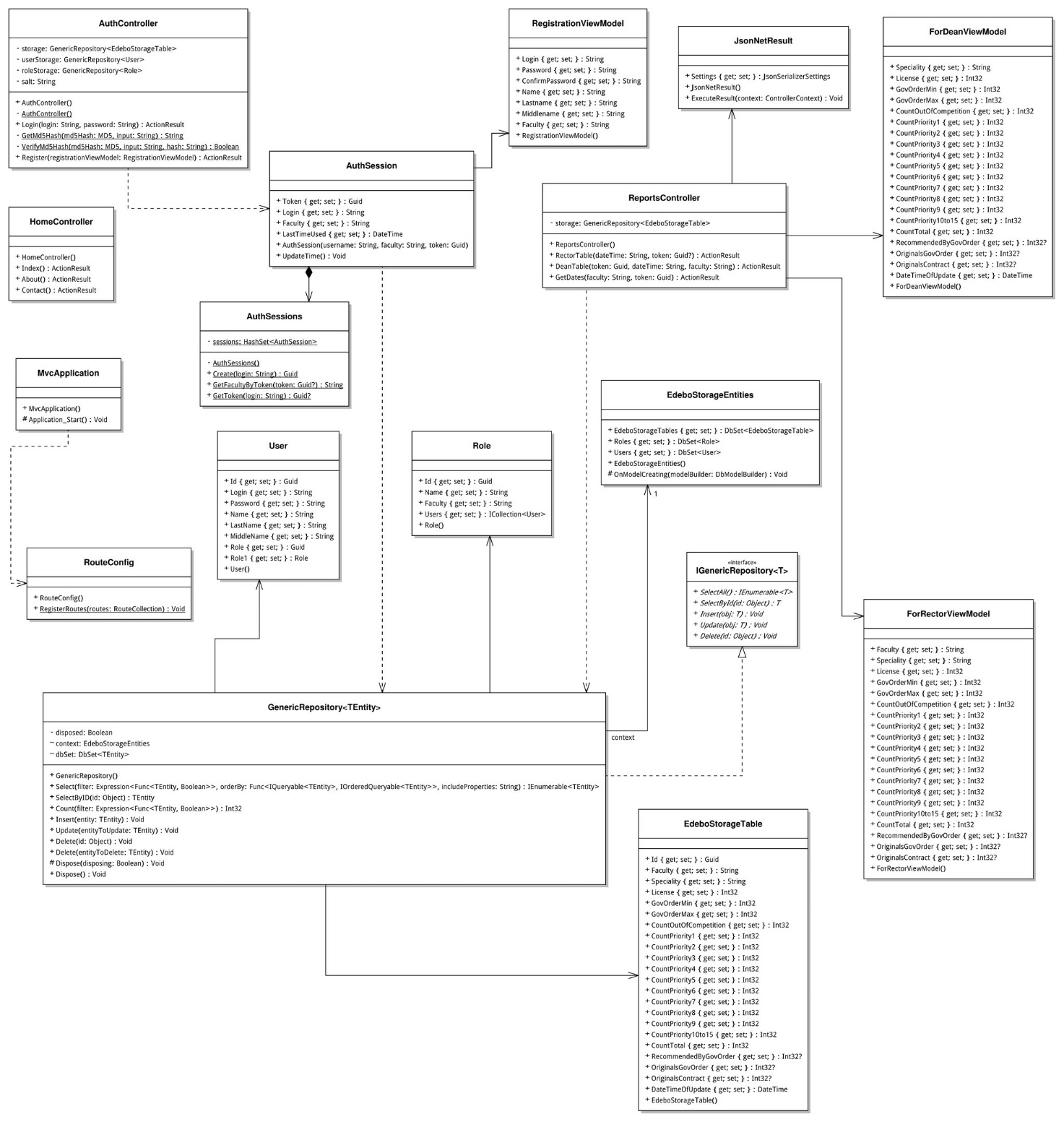


Рисунок 2.5 – Діаграма класів

2.2 Внутрішнє проектування

2.2.1 Вибір мови програмування та середовища розробки

Для вирішення поставленою задачі програмними засобами у якості основних мов програмування було обрано C# для написання серверної частини сайту та JavaScript для надання веб-сторінці інтерактивності та для можливості реалізації SPA. Цей вибір підкріплюють опитування, проведені серед програмістів у 2016 році (рис. 2.6). Мови C# та JavaScript знаходяться у трійці лідерів як в Україні так і за кордоном.

C# – об'єктно-орієнтована мова програмування. Розроблена в 1998-2001 роках групою інженерів під керівництвом Андерса Хейлсберг в компанії Microsoft як мова розробки додатків для платформи Microsoft .NET Framework і згодом був стандартизований як ECMA-334 і ISO / IEC 23270 [10].

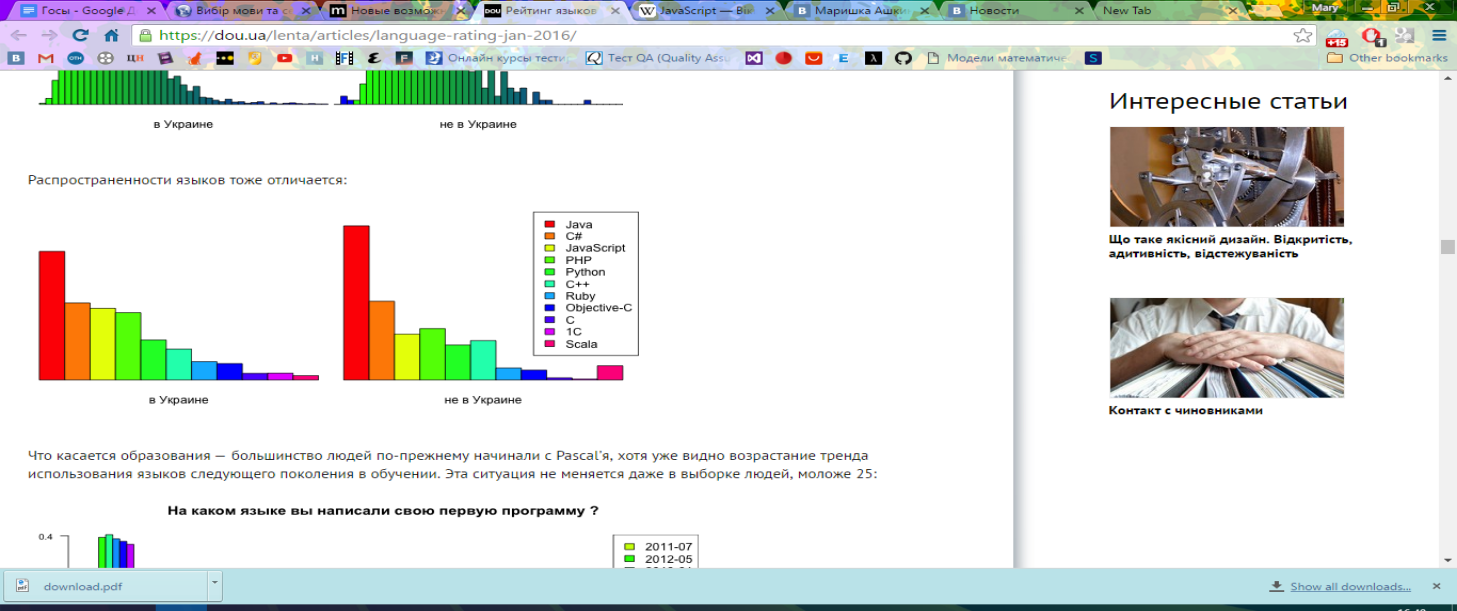
C# відноситься до сім'ї мов з C-подібним синтаксисом, з них його синтаксис найбільш близький до C++ і Java. Мова має статичну типізацію, підтримує поліморфізм, перевантаження операторів (в тому числі операторів явного і неявного приведення типу), делегати, атрибути, події, властивості, узагальнені типи і методи, ітератори, анонімні функції з підтримкою замикань, LINQ, виключення, коментарі в форматі XML.

Рисунок 2.6 – Розповсюдження мов програмування

При порівнянні C# та С++ за деякими критеріями випливає наступне:

Швидкість розробки

С# дозволяє стартувати розробку швидше, а це дозволяє швидше отримати прототип рішення. Швидкість розробки на С# на початкових етапах проекту значно вище в порівнянні з С++.

Однак, коли інфраструктура проекту створена, основні підходи і бібліотеки обрані, а білд налаштований, швидкість розробки на С++ і швидкість розробки на С# стають приблизно однаковими.

Продуктивність коду і вимогливість до ресурсів

Очевидним є факт того, що по можливості оптимізації некерованого коду куди ширше, ніж по можливості оптимізації керованого коду. Таким чином, пікова продуктивність коду досяжна тільки в некерований виконанні, тобто в межі, майже будь-яке завдання на С++ може бути вирішена з меншими вимогами до ресурсів. Тому в важких завданнях, пов'язаних з обробкою великої кількості даних, С++ має сильні переваги перед С#.

Якщо говорити про сукупності суб'єктивних «простоти розробки», «краси коду» і об'єктивної продуктивності, то використовуючи C # простіше написати код, що задовольняє цим критеріям одночасно.

Бібліотеки

Відмінність асортименту С++ і С# бібліотек в тому, що С++ бібліотек більше, вони мають велику історію, за яку стали непогано налагоджені і оптимізовані, часто кросплатформені, багато з відкритим кодом. Однак при всіх позитивних сторонах С ++ бібліотеки як мають дуже різну, часто навіть архаїчну архітектуру, часто вже не об'єктний, а структурно-процедурний інтерфейс.

Інша неприємна особливість С ++ бібліотек - це створення і перевизначення своїх базових типів. С++: Багато бібліотек заводять свої типи рядків, контейнерів, скасовують деякі базові типи. Базові ж С++ бібліотеки дають не так багато, як дають стандартні бібліотеки С#.

У С# перерахованих вище проблем значно менше. Величезна кількість бібліотек з .NET йде в базі, плюс до них безліч вільно доступних бібліотек, це покриває практично всі першорядні завдання розробки під Windows. Наявність великої кількості стандартних типів майже позбавляє від бібліотек, де базові типи перевизначені. І в силу того, що бібліотеки С# порівняно молоді, інтерфейси бібліотек, як правило, краще вписуються в ті чи інші шаблони проектування, що часто спрощує їх вивчення.

Наочно ступінь переваг С# над С++ відображено на рис.2.7, який відображає індекс задоволеності мовою програмування та бажання використвовувати цю мову у наступних проектах. Індекс розраховано спираючись на опитування, проведені у 2015 році. С# тут займає третю позицію, а індекс дорівнює 0,82.

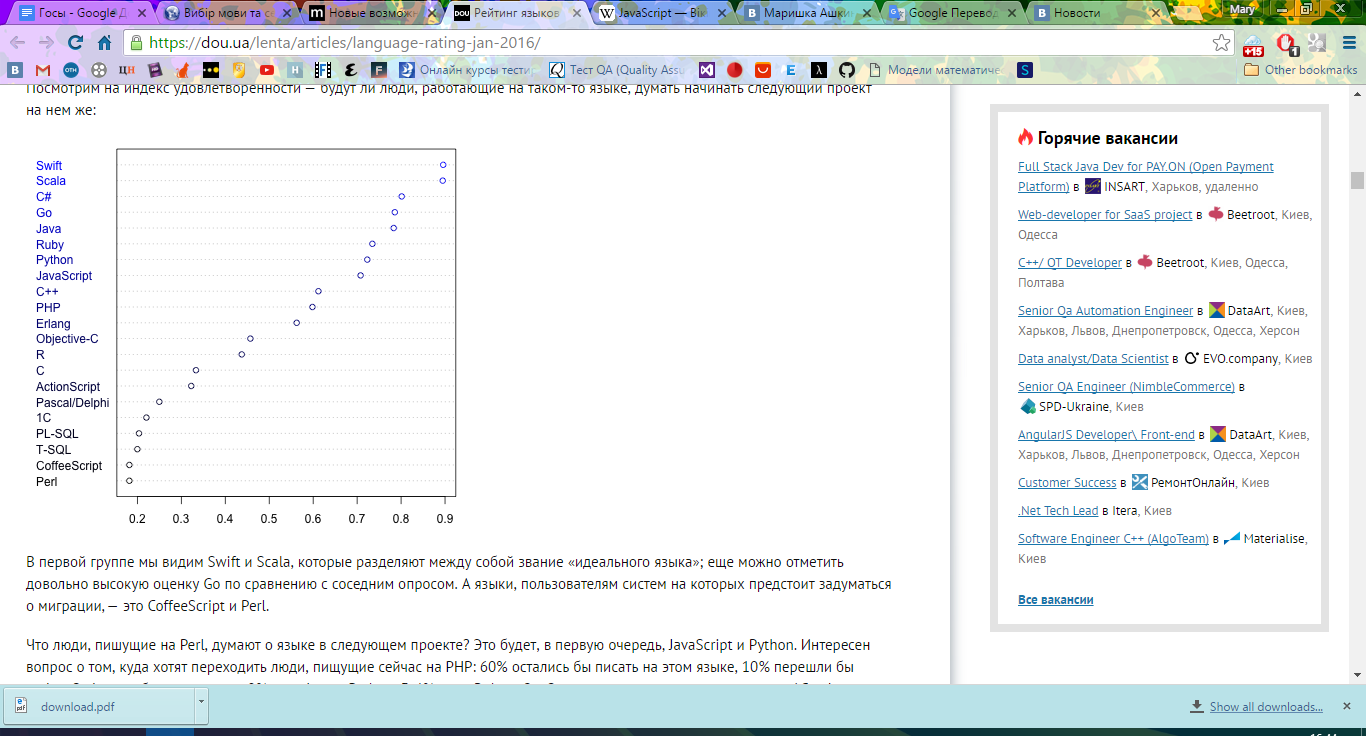


Рисунок 2.7 – Індекс задоволеності використання мови

програмування для наступного проекту

Як мову сценаріїв для веб-сторінок було обрано Java Script. JavaScript (JS) – динамічна, об'єктно-орієнтована мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується як частина браузера, що надає можливість коду на стороні клієнта (такому, що виконується на пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки [11].

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

### 2.2.2 Вибір інструментів та парадигм програмування

Парадигми програмування

Парадигма програмування в сучасній індустрії програмування дуже часто визначається набором інструментів програміста (мова програмування, операційна система тощо).

Парадигма програмування визначає те, як програміст бачить виконання програми. Наприклад, в об'єктно-орієнтованому програмуванні програміст розглядає програму як набір взаємодіючих об'єктів, тоді як у функціональному (або функційному) програмуванні програма представляється у вигляді ланцюжка обчислення функцій. Під час реалізації даного проекту були використані обидві ці найпоширеніші парадигми програмування.

Функційне програмування – парадигма програмування, яка розглядає програму як обчислення математичних функцій та уникає станів та змінних даних. Функційне програмування наголошує на застосуванні функцій, на відміну від імперативного програмування, яке наголошує на змінах в стані та виконанні послідовностей команд.

Іншими словами, функційне програмування є способом створення програм, в яких єдиною дією є виклик функції, єдиним способом розбиття програми є створення нового імені функції та задання для цього імені виразу, що обчислює значення функції, а єдиним правилом композиції є оператор суперпозиції функцій. Жодних комірок пам'яті, операторів присвоєння, циклів, ні, тим більше, блок-схем чи передачі управління.

Принципово немає перешкод для написання програм у функціональному стилі на мовах, які традиційно не вважаються функціональними. У мові C # версії 3.0 і вище можна використовувати лямбда-вирази для написання програми в функціональному стилі. Лямбда-вираз в програмуванні – спеціальний синтаксис для визначення функціональних об'єктів, запозичений з лямбда-обчислення, який застосовується як правило для оголошення анонімних функцій. Створення спеціальної бібліотеки LINQ, що дозволяє писати функціональні запити у C# і зробило цю мову програмування мультипарадигменною мовою. Окрім мови C# лямбда-вирази підтримуються в багатьох інших мовах програмування, серед них: Common Lisp, Ruby, Python, PHP, F#, Visual Basic .NET, C ++, Java, Scala та інші.

В проекті лямбда-вирази дозволили спростити написання SELECT та ORDER BY, необхідних для вибору певної інформації зі сховища даних для їх перенесення в звіти ректора. Бібліотека LINQ дозволяє працювати з програмними об’єктами, а система сама конвертує написані вирази у SQL-запити, тому таке рішення стало можливим і значно облегшило реалізацію функції доступу до необхідних даних.

Друга використана парадигма програмування – об’єктно-орієнтована парадигма. Об'єктно-орієнтоване програмування представляє програму як набір об'єктів і їх взаємодій. Основними поняттями ООП є наступні [12]:

* об'єкт – елементарна сутність, описувана певними властивостями (що зберігаються у вигляді атрибутів об'єкта) і поведінкою (реалізованим у вигляді методів);
* клас описує структуру властивостей і поведінки одного типу об'єктів. Кожен об'єкт програми є екземпляром деякого класу;
* класи можуть успадковувати атрибути і методи їх батьківських класів, в той же час додаючи свої власні. Ієрархія класів дозволяє моделювати суті розв'язуваної задачі на декількох рівнях деталізації і в подальшому використовувати клас, який відповідає рівню деталізації, необхідного для вирішення конкретної підзадачі;
* інкапсуляція передбачає, що деякі деталі реалізації класу приховані від взаємодіючих з ним об'єктів. У кожного класу є інтерфейс, що описує взаємодію об'єктів цього класу з іншими об'єктами, і реалізація, що описує те, як ця взаємодія відбивається.

Об’єктно-орієнтована технологія створення програмного забезпечення була задумана і розроблена як інструмент подолання складності. Вона успадкувала всі найкращі надбання структурного та модульного програмування, використавши їх для реалізації ряду принципово нових підходів до проектування програмного забезпечення. Головним завданням об’єктно-орієнтованого підходу є забезпечення способу структурування програми та керування складними взаємозв’язками між великою кількістю компонентів системи.

ООП дає можливість створювати системи, що розгортаються. Це одна з основних переваг ООП, і саме вона відрізняє даний підхід від традиційних методів програмування. Розширюваність означає, що існуючу систему можна змусити працювати з новими компонентами, причому без внесення до неї будь-яких змін.

Таким чином при розробці робочого місця ректора були використані дві парадигми програмування: ООП та функційне програмування. Саме використання їх переваг дозволило розробити сучасний програмний додаток, який виконує усі поставлені перед ним задачі.

Інструменти програмування

Розробка сучасного програмного продукту вимагає знання та вміння використовувати багату кількість технологій програмування. Зокрема для розробки робочого місця ректора для аналізу контингенту абітурієнтів була обрана програмна платформа .NET Framework 4.6, фреймворк для оформлення веб-сторінок Twitter Bootstrap, мова стилів CSS 3 та мова гіпертекстової розмітки HTML 5. У якості СУБД був обраний MS SQL Server.

Для серверної частини проекту була обрана мова програмування С#, яка тісно пов'язана з використанням платформи .NET Framework. Це програмна платформа, випущена компанією Microsoft в 2002 році. Основою платформи є загальномовне середовище виконання Common Language Runtime (CLR), яке підходить для різних мов програмування. Вважається, що платформа .NET Framework стала відповіддю компанії Microsoft на набрала на той час велику популярність платформу Java компанії Sun Microsystems (нині належить Oracle).

Основні переваги платформи у веб-розробці:

* компіляція частин веб-додатка, які виконуються, завдяки якій вони працюють швидше, ніж інтерпретуємі сценарії;
* відновлення розгорнутих веб-додатків «з льоту» без перезапуску сервера;
* використання відомої мови програмування Visual Basic, що тепер повністю підтримує об’єктно-орієноване програмування (ООП), і нової мови Visual C# - більше модифікованої версії С, що підтримує ООП і контроль типів;
* автоматичне керування станом елементів керування веб-сторінок (називаних серверними елементами керування – server controls). Завдяки цій особливості їхнє поводження мало відрізняється від звичайних елементів керування Windows;
* можливість створення нових, нестандартних серверних елементів керування на основі існуючих;
* забезпечення безпеки веб-додатків за допомогою убудованих захисних механізмів сервера Windows і інших методів аутентифікації й авторизації;
* інтеграція з ADO.NET, що дозволяє одержувати доступ до БД і застосовувати інструменти для конструювання БД із Visual Studio .NET;
* повна підтримка мови Extensible Markup Language (XML), каскадних таблиць стилів (CSS) і інших нових стандартів Web;
* вбудовані механізми кешування часто запитуваних веб-сторінок на сервері, локалізації вмісту (пристосування його для використання різних мов і регіональних стандартів) і визначення можливостей браузера.

Зараз важко уявити сучасний веб-сайт у оформлені якого не використовується CSS, не стало виключенням і робоче місце декана для аналізу контингенту абітурієнтів. CSS – це мова стилів, яка визначає відображення HTML-документів. Наприклад, CSS працює з шрифтами, кольором, полями, рядками, висотою, шириною, фоновими зображеннями, позиціонуванням елементів і багатьма іншими речами.

HTML також може використовуватися для оформлення веб-сайтів, але CSS надає більші можливості. HTML використовується для структурування вмісту сторінки, а CSS використовується для форматування цього структурованого вмісту.

Фреймворк для оформлення веб-сторінок Twitter Bootstrap також включає в себе HTML і CSS шаблони оформлення для веб-форм, кнопок, міток, блоків навігації та інших компонентів веб-інтерфейсу, включаючи JavaScript-розширення.

Саме цей фреймворк було обрано для розробки робочого місця ректора, тому що він є надзвичайно простим у використанні, не потребує глибокого знання HTML і CSS технологій, і також він надає велику кількість можливостей у оформленні веб-сайтів. Компоненти платформи Bootstrap використовують єдиний стиль і шаблони за допомогою центральної бібліотеки. Дизайн і макети веб-сторінок узгоджуються один з одним, що дозволяє спроектувати простий, красивий і зрозумілий інтерфейс.

Необхідно відмітити, що Bootstrap користується великою популярність серед професійних веб-розробників. Підтвердженням цього є те, що саме з його використанням були розроблені такі веб-сайти та сервіси як сайт NASA, FIFA, Microsoft Power BI, Visual Studio Code, HBO NOW та багато інших.

Проект, що розробляється, потребував створення сховища даних. Для реалізації цієї вимоги був обраний Microsoft SQL Server. Вибір саме цієї СУБД зумовлений тим що, взаємодія між програмою, написаною на C# з використанням .NET у середовищі розробки Visual Studio 2015, та Microsoft SQL є налагодженою та простою у реалізації, адже всі перераховані технології належать до одного стеку технологій.

Вибір СУБД також зумовлений високою надійністю MS SQL, адже усі дані перед записом на жорсткий диск шифруються, а їх розшифрування проводитися при читанні. Можливість шифрування надзвичайно важлива для даного проекту, адже у сховищі даних будуть зберігатися реальні особисті дані абітурієнтів.

2.2.3 Визначення призначень об’єктів за допомогою карток CRC

Важливим етапом створення програмного забезпечення є проектування. На цьому кроці закладається архітектура системи. Одним із способів проектування є метод CRC-карток. Цей метод проектування є складовою UML-проектування.

CRC картки (Class-Responsibility-Collaboration) – це зручний спосіб для визначення класів і їх взаємодії. Використання карток мінімізує рівень складності на перших етапах проектування. Головним призначенням CRC карток є концентрування розробники на головних абстракціях завдання і запобігання занурення у подробиці та другорядні деталі.

Зверху картки пишеться назва класу, в лівій половині – за що він відповідає, в правій – з ким співпрацює.

У таблицях 2.1 – 2.14 наведено CRC-картки на класи проекту.

Таблиця 2.1 – CRC-картка на клас ReportsController

|  |  |
| --- | --- |
| ReportsController | |
| Відповідальність | Зв’язки |
| Прийняття запитів на повернення звітів та списку історії звітів | Використовує AuthSession, AuthSessions, ForRectorViewModel, GenericStorageRepository |

Таблиця 2.2 – CRC-картка на клас AuthController

|  |  |
| --- | --- |
| AuthController | |
| Відповідальність | Зв’язки |
| Прийняття запитів на повернення звітів та списку історії звітів | Використовує AuthSession, AuthSessions, GenericRoleRepository, GenericUserRepository |

Таблиця 2.3 – CRC-картка на клас AuthSession

|  |  |
| --- | --- |
| AuthSession | |
| Відповідальність | Зв’язки |
| Містить логіку та інформацію для окремої сесії | Використовується у ReportsController, AuthController, |

Таблиця 2.4 – CRC-картка на клас AuthSessions

|  |  |
| --- | --- |
| AuthSessions | |
| Відповідальність | Зв’язки |
| Зберігає, створює, видаляє та оброблює сесії | Використовує Role, User, GenericRoleRepository, GenericUserRepository  Використовується у ReportsController, AuthController, |

Таблиця 2.5 – CRC-картка на клас ForRectorViewModel

|  |  |
| --- | --- |
| ForRectorViewModel | |
| Відповідальність | Зв’язки |
| Модель для відправлення даних для звіту ректора | Використовується у ReportsController |

Таблиця 2.6 – CRC-картка на клас IGenericRepository

|  |  |
| --- | --- |
| IGenericRepository | |
| Відповідальність | Зв’язки |
| Містить абстракцію для доступу до БД | Використовується у GenericRepository |

Таблиця 2.7 – CRC-картка на клас GenericRepository

|  |  |
| --- | --- |
| GenericRepository | |
| Відповідальність | Зв’язки |
| Містить реалізацію абстракції доступу до БД | Використовує IGenericRepository, Використовується у ReportsController, GenericStorageRepository, GenericRoleRepository, GenericUserRepository |

Таблиця 2.8 – CRC-картка на клас StorageModel

|  |  |
| --- | --- |
| StorageModel | |
| Відповідальність | Зв’язки |
| Містить модель даних для сховища локальної бази | Використовується у GenericStorageRepository |

Таблиця 2.9 – CRC-картка на клас Role

|  |  |
| --- | --- |
| Role | |
| Відповідальність | Зв’язки |
| Містить модель даних для ролей користувачів локальної бази | Використовується у AuthSessions, GenericRoleRepository |

Таблиця 2.10 – CRC-картка на клас User

|  |  |
| --- | --- |
| User | |
| Відповідальність | Зв’язки |
| Містить модель даних для користувачів локальної бази | Використовується у AuthSessions, GenericUserRepository |

Таблиця 2.11 – CRC-картка на клас admission

|  |  |
| --- | --- |
| admission | |
| Відповідальність | Зв’язки |
| Містить логіку поведінки GUI та представлення даних на клієнтській частині |  |

Таблиця 2.12 – CRC-картка на клас GenericStorageRepository

|  |  |
| --- | --- |
| GenericStorageRepository | |
| Відповідальність | Зв’язки |
| Містить реалізацю абстракції доступу до таблиці зі сховищем БД | Використовує GenericRepository, StorageModel |

Таблиця 2.13 – CRC-картка на клас GenericRoleRepository

|  |  |
| --- | --- |
| GenericRoleRepository | |
| Відповідальність | Зв’язки |
| Містить реалізацю абстракції доступу до таблиці з ролями користувачів у БД | Використовує GenericRepository, Role  Використовується у AuthController, AuthSessions |

Таблиця 2.14 – CRC-картка на клас GenericUserRepository

|  |  |
| --- | --- |
| GenericUserRepository | |
| Відповідальність | Зв’язки |
| Містить реалізацю абстракції доступу до таблиці з користувачами у БД | Використовує GenericRepository, User  Використовується у AuthController, AuthSessions |

2.2.4 Проектування інтерфейсу користувача

Проектування інтерфейсу користувача є надзвичайно важливою задачею у цьому проекті. Інтерфейс користувача (англ. user interface, UI) – засіб зручної взаємодії користувача з інформаційною системою. Сукупність засобів для обробки та відображення інформації, максимально пристосованих для зручності користувача; у графічних системах інтерфейс користувача реалізовується багатовіконним режимом, змінами кольору, розміру, доступністю багатокористувацьких налаштувань.

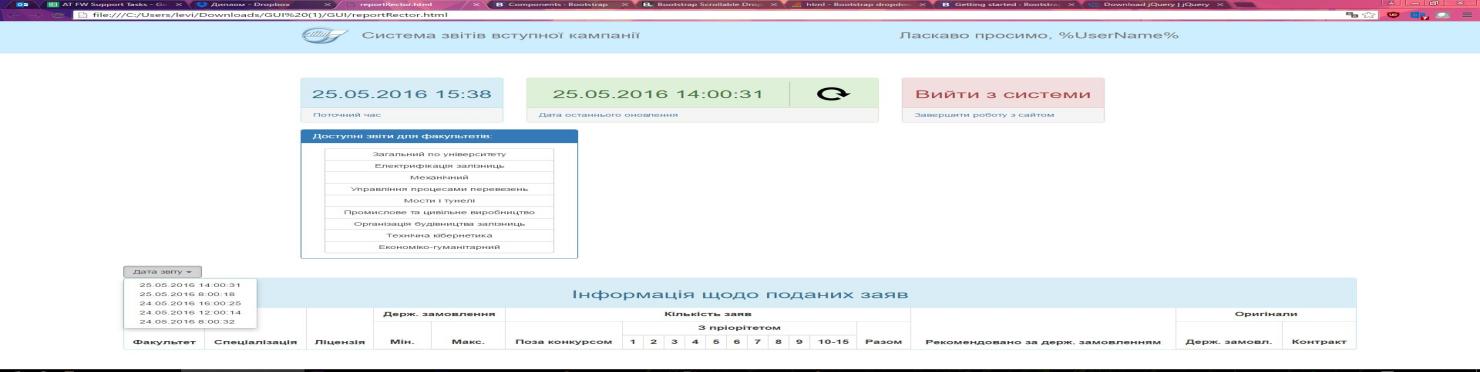
На головній сторінці веб-сайту користувач бачить меню, завдяки якому він може зручно працювати з сайтом (рис. 2.8). Великим та помітним шрифтом виділені дата останнього оновлення даних та поточний час, адже ця інформація є вкрай важливою для роботи користувача з актуальними даними у звітах.

Рисунок 2.8 – Меню сайту

Уся інформація виводиться у вигляді таблиць (рис. 2.9). Таке представлення даних у найбільш зручному вигляді дозволяє легко знаходити усю необхідну інформацію. Формат таблиці узгоджено з працівниками приймальної комісії.

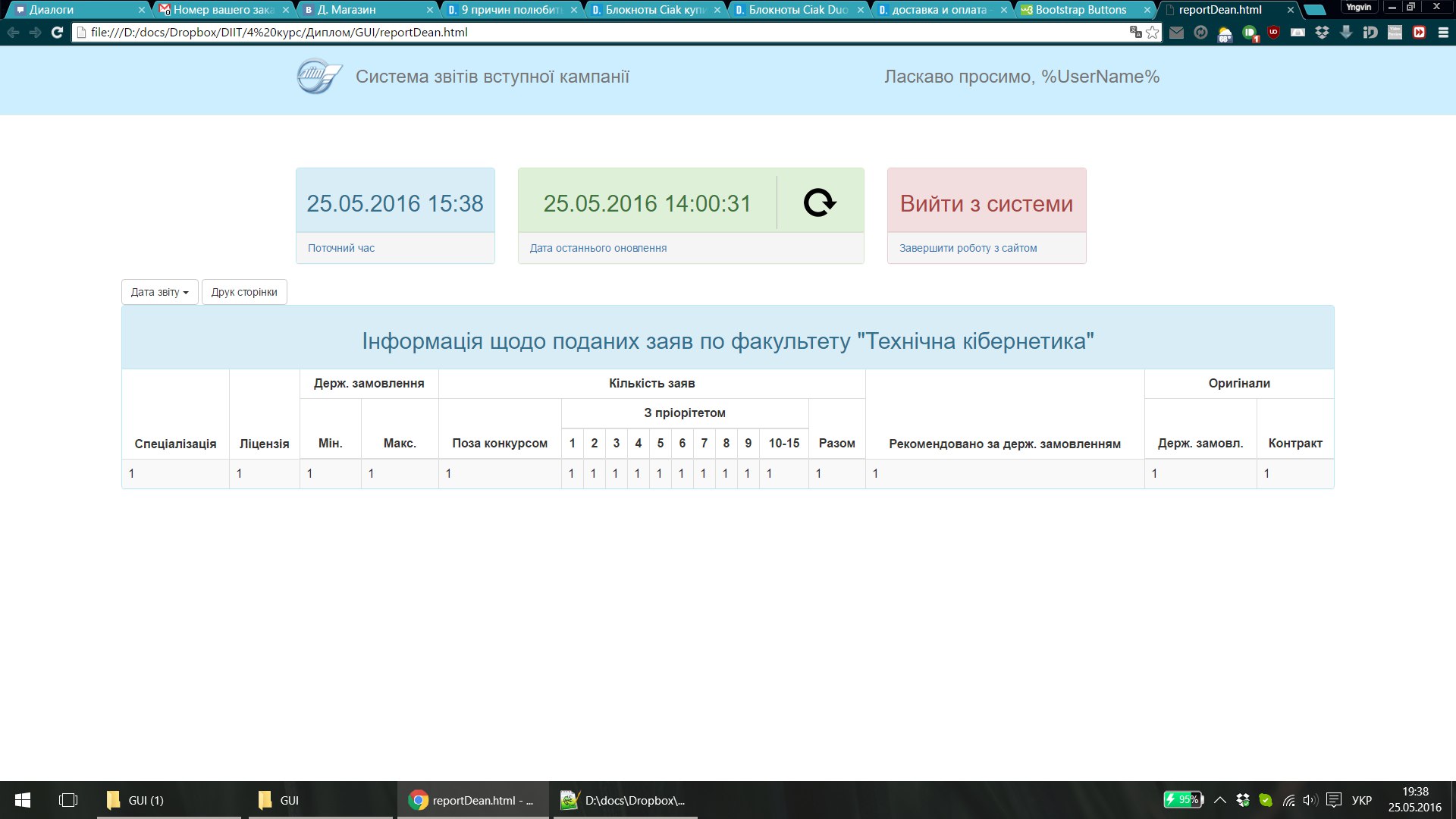
Для вибору необхідного звіту за змістом (загальний по університету, по окремому факультету), розроблений зручний список з переліком усіх можливих варіантів.

Рисунок 2.9 – Приклад таблиці

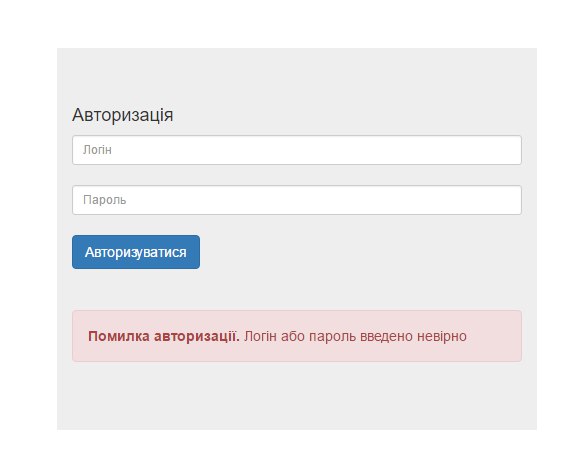
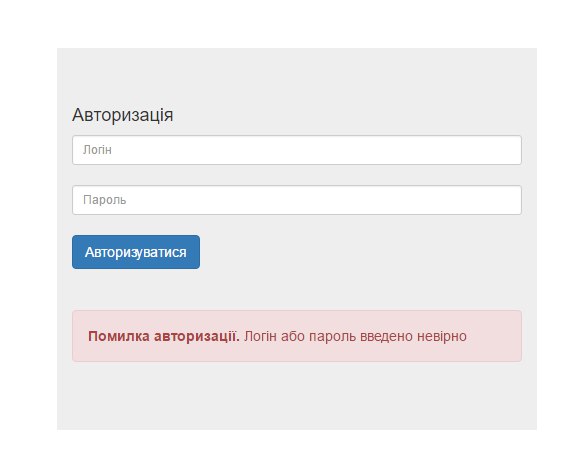
Для авторизації користувачів розроблена відповідна сторінка. Кнопка «Авторизуватися» виділена темнішим кольором для звернення уваги користувача (рис 2.10).

Рисунок 2.10 – Сторінка авторизації

Також користувачу допоможуть повідомлення, випадаючі списки та кнопки (рис. 2.11 – 2.13). Наявність випадаючого списку дозволить легко знайти необхідний за датою та часом звіт по факультету.

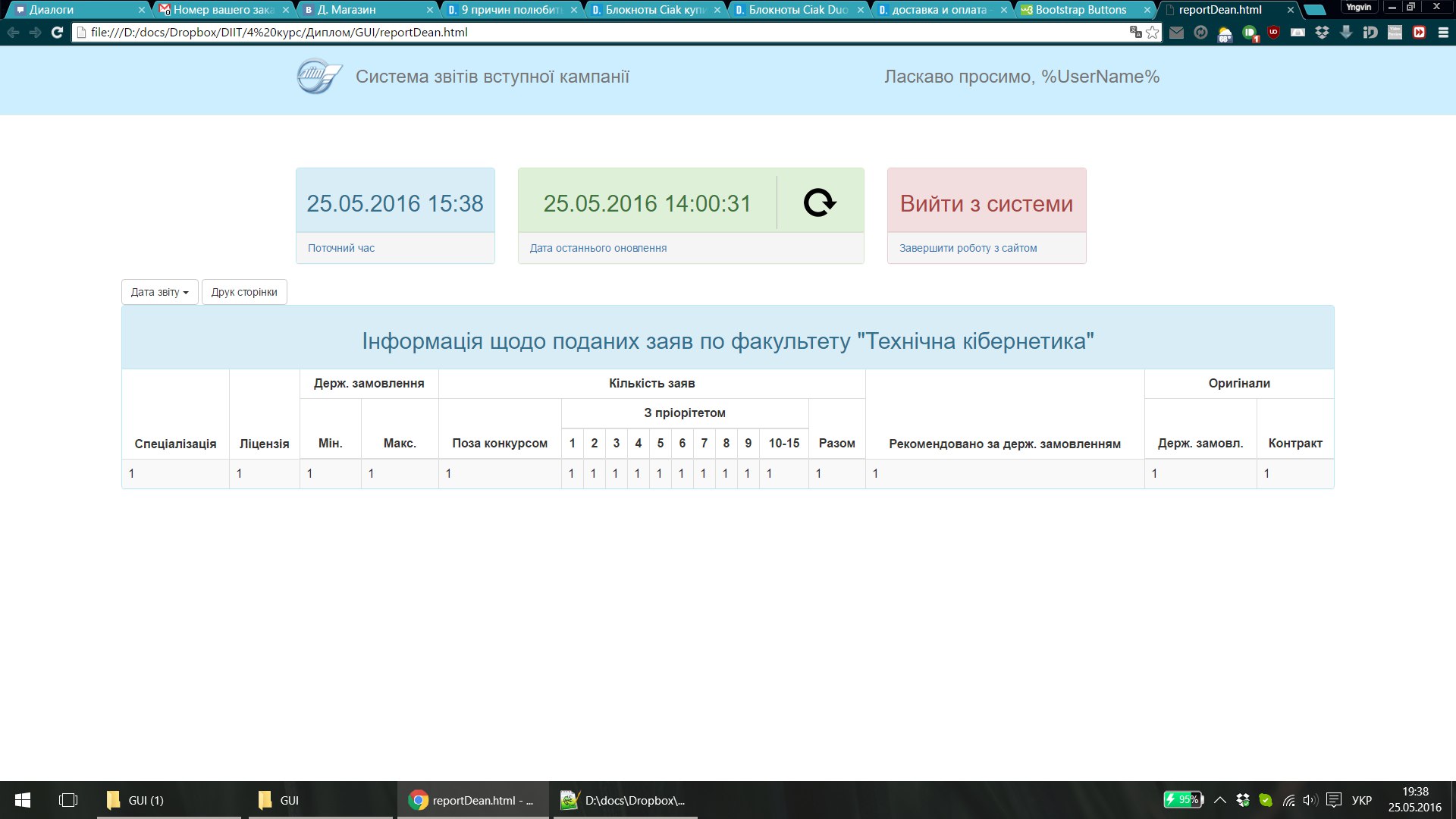
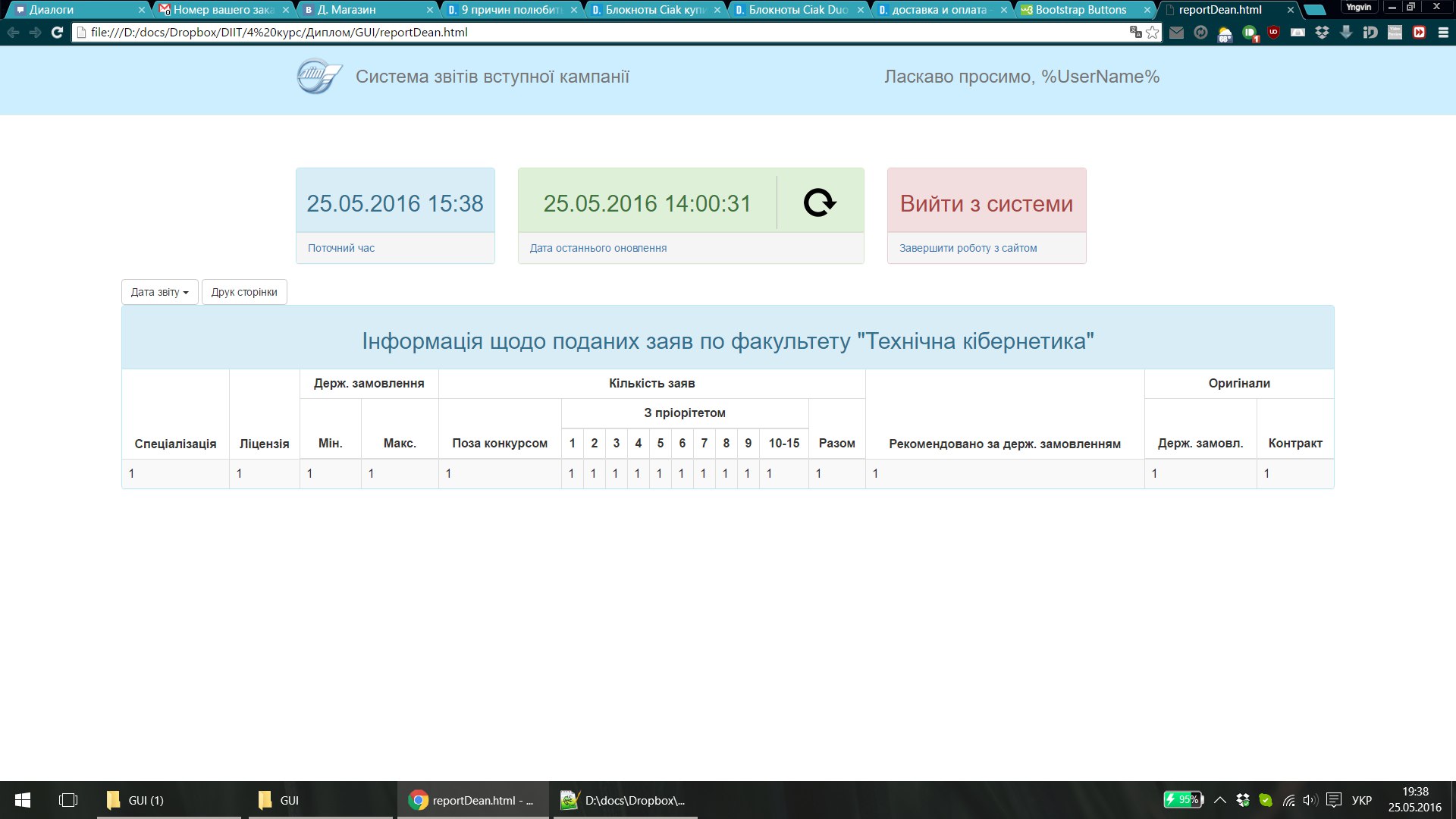
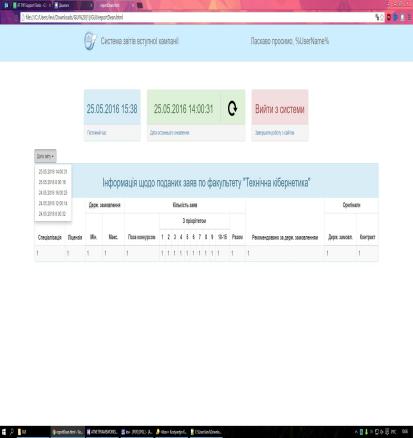
Рисунок 2.11 – Приклад повідомлення

Рисунок 2.12 – Приклад кнопок

Рисунок 2.13 – Приклад випадаючого списку

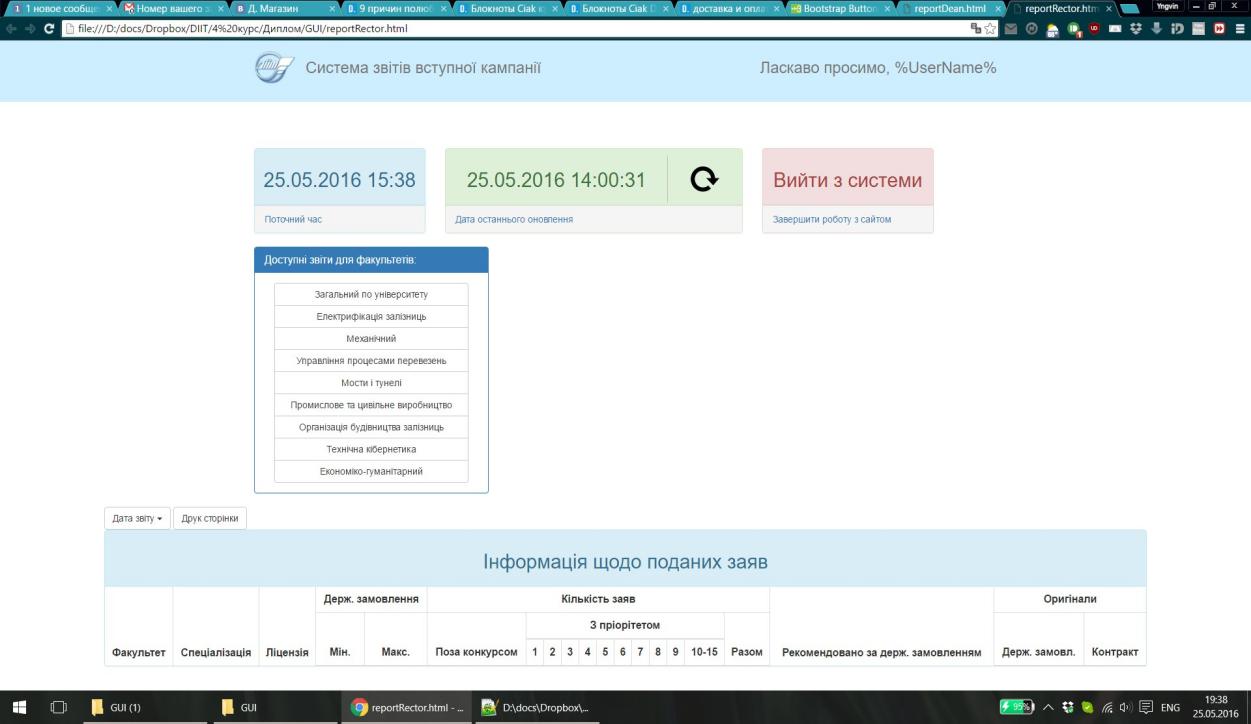
Загальний дизайн інтерфейсу програмного продукту виконано у мінімалістичному стилі та без використання яскравих кольорів. Це дозволить зосередити увагу користувача на головному – таблиці звітів, що відображає контингент абітурієнтів, що подали заявки на факультет. Веб-сайт не перевантажений зайвим функціоналом, тому навіть не дуже досвідчений користувач зможе виконати необхідні задачі без витрачання часу на пошук потрібної кнопки або пошуку певного пункту в громіздкому меню (рис. 2.14).

Рисунок 2.14 – Загальний вигляд веб-сайту

# Реалізація спроектованоГО проЕКТУ

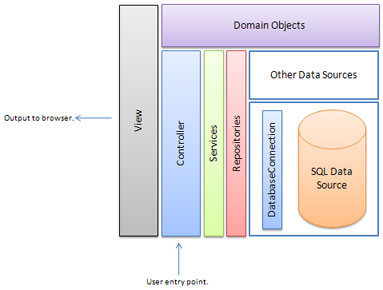
Реалізація проекту відбувалася з використанням багаторівневої архітектури. Проект було поділено на три рівні: користувацький інтерфейс, рівень бізнес-логіки, та рівень доступу до даних, який забезпечує взаємодію з спроектованим сховищем даних [1], яке зберігає необхідну інформацію про абітурієнтів (рис. 3.1).

Рисунок 3.1 – Представлення багаторівневої архітектури

Рішення поділити проект на відповідні рівні означає, що:

* кожен рівень займається конкретним завданням;
* логіка різних рівнів не повторюється і не перетинається;
* спосіб звернення до нижчого рівня чітко визначений;
* спосіб поставки інформації вищому рівню чітко визначений;
* рівні слабо пов'язані;
* рівні розташовані вертикально, хоча є наскрізна функціональність, яка може пронизувати піраміду зверху вниз;
* рівні можуть розміщуватися фізично на одному комп'ютері (в межах одного рівня), а можуть бути на різних машинах, наприклад, в розподілених додатках;
* логіка різних рівнів інкапсульована та абстрагована, відповідно, різним верствам не потрібно робити ніяких припущень про те, як реалізований код інших рівнів додатку;
* можливість заміни логіки виконання при збереженні абстракції, що дає можливість змінити логіку одного шару без шкоди для інших.

## Рівень доступу до даних

Рівень доступу до даних (або DAL) реалізовано з використанням шаблона «Репозиторій». Найбільшою перевагою шаблону є можливість абстрагуватися від конкретних підключень до джерел даних, з якими працює програма та працювати з ними через спеціальний інтерфейс. Таким чином це надає гнучкості при роботі з різними типами підключень до різноманітних СУБД. Змінення у майбутньому обраного рівня з даними не стане проблемою.

«Репозиторій» є одним з найпоширеніших шаблонів, який використовується у програмних продуктах, що працюють з даними.

У даному проекті шаблон реалізовано через інтерфейс IGenericRepository, який описує основні CRUD операції (Create, Read, Update, Delete) для роботи з даними. У даному проекті інтерфейс репозиторія був реалізований наступним чином:

public interface IGenericRepository<T> : IDisposable

{

IEnumerable<T> SelectAll();

T SelectById(object id);

void Insert(T obj);

void Update(T obj);

void Delete(object id);

}

Операції SelectById, Insert, Update та Delete реалізовані в класі GenericRepository:

public virtual TEntity SelectByID(object id)

{

return dbSet.Find(id);

}

public virtual void Insert(TEntity entity)

{

dbSet.Add(entity);

context.SaveChanges();

}

public virtual void Update(TEntity entityToUpdate)

{

dbSet.Attach(entityToUpdate);

context.Entry(entityToUpdate).State = EntityState.Modified;

context.SaveChanges();

}

public virtual void Delete(object id)

{

TEntity entityToDelete = dbSet.Find(id);

Delete(entityToDelete);

context.Entry(entityToDelete).State = EntityState.Deleted;

context.SaveChanges();

}

## 3.2 Рівень бізнес-логіки

Рівень бізнес-логіки (BLL – Business Logic Layer) містить класи, які описують сутності і основну логіку роботи програми (основні алгоритми роботи програми, обчислення, обробка даних і так далі).

Для робочого місця ректора найголовнішим є отримання необхідних звітів, тому на цьому рівні реалізовано клас ReportsController, який є контролером, який відповідає за прийняття запитів на повернення необхідних звітів.

Для отримання необхідних даних зі сховища необхідні запити Select та ОrderBy, які в цьому проекті були реалізовані за допомогою лямбда-виразів:

EdeboStorageTable resultTuple = storage.Select(filter:  storage  =>  storage.DateTimeOfUpdate == dateTime, orderBy: storage => storage.OrderBy(orderedStorage =>orderedStorage.Faculty). ThenBy(orderedStorage => orderedStorage.Speciality)).FirstOrDefault();

Конвертація однієї моделі даних в іншу, так званий мапінг даних, реалізовано наступним чином:

ForRectorViewModel viewModel = new ForRectorViewModel()

{

CountOutOfCompetition = resultTuple.CountOutOfCompetition,

CountPriority1 = resultTuple.CountPriority1,

CountPriority2 = resultTuple.CountPriority2,

CountPriority3 = resultTuple.CountPriority3,

CountPriority4 = resultTuple.CountPriority4,

CountPriority5 = resultTuple.CountPriority5,

CountPriority6 = resultTuple.CountPriority6,

CountPriority7 = resultTuple.CountPriority7,

CountPriority8 = resultTuple.CountPriority8,

CountPriority9 = resultTuple.CountPriority9,

CountPriority10to15 = resultTuple.CountPriority10to15,

CountTotal = resultTuple.CountTotal,

DateTimeOfUpdate = resultTuple.DateTimeOfUpdate,

Faculty = resultTuple.Faculty,

GovOrderMax = resultTuple.GovOrderMax,

GovOrderMin = resultTuple.GovOrderMin,

License = resultTuple.License,

OriginalsContract = resultTuple.OriginalsContract,

OriginalsGovOrder = resultTuple.OriginalsGovOrder,

RecommendedByGovOrder = resultTuple.RecommendedByGovOrder,

Speciality = resultTuple.Speciality

};

Також, була реалізована авторизація та система сесій для користувачів. Система сесій є типовою для RESTful додатків і полягає у тому, що після авторизації клієнту видається токен – певний ключ, часто у вигляді GUID, який у всіх запитах (крім авторизації) присутній як обов’язковий параметр і служить маркером, що допомагає серверу визначити від кого саме прийшов запит і, як наслідок, перевірити чи можна надати інформацію по цьому запиту згідно з правами доступу.

Список користувачів та їх ролей було реалізовано у двох таблицях, оскільки зв’язок багато-до-багатьох не потрібен – кожному користувачеві може відповідати лише одна роль, яка визначає інформацію якого факультету може переглядати конкретний користувач. Паролі зберігаються у зашифрованому вигляді (MD5 + приптографічна сіль) – це зроблено з міркувань безпеки. Видача токену відбувається після аутентифікації і створення сесії, яка буде міститися у списку сесій. Якщо запити по сесії не відбувалися більше 15 хвилин, вона видаляється і користувач повинен знову аутентифікуватися.

Нижчеприведений код демонструє роботу контролера, відповідного за аутентифікацію. Він приймає логін та пароль, потім шифрує пароль і відправляє запит на знаходження в базі користувача з отриманим логіном та зашифрованим паролем. Якщо такий не знайдений, то повертається помилка 403, якщо навпаки, то далі перевіряється чи існує вже сесія для такого користувача. Якщо так, то повертається вже створений токен, якщо ні, він генерується і створюється нова сесія.

public ActionResult Login(string login, string password)

{

string passwordHash = GetMd5Hash(MD5.Create(), password);

List<User> auth = (List<User>)userStorage.Select(filter: user => user.Login == login && user.Password == passwordHash);

if (auth.Count != 1)

{

Response.StatusCode = 403;

return Json(false, JsonRequestBehavior.AllowGet);

}

Guid? sessionToken = AuthSessions.GetToken(login);

if (sessionToken == null)

{

sessionToken = AuthSessions.Create(login);

}

string faculty = AuthSessions.GetFacultyByToken(sessionToken);

string lastUpdateTime = storage.Select().Max(columns => columns.DateTimeOfUpdate).ToString("dd.MM.yyyy HH:mm:ss");

return Json(new { sessionToken, faculty, lastUpdateTime }, JsonRequestBehavior.AllowGet);

}

Для сесії було створено модель, що зберігає токен, логін, факультет та дату останнього запиту.

public class AuthSession

{

/// <summary>

/// Токен авторизації

/// </summary>

public Guid Token { get; private set; }

/// <summary>

/// Ім'я користувача

/// </summary>

public string Login { get; private set; }

/// <summary>

/// Факультет, для якого можна видавати звіти

/// </summary>

public string Faculty { get; private set; }

public DateTime LastTimeUsed { get; set; }

public AuthSession(string username, string faculty, Guid token)

{

Token = token;

Login = username;

Faculty = faculty;

LastTimeUsed = DateTime.Now;

}

public void UpdateTime()

{

LastTimeUsed = DateTime.Now;

}

}

Клас, що зберігає сесії вміє створювати сесії, визначати інформацію по якому факультету може переглянути носій певного токену та шукати токен по логіну.

public static class AuthSessions

{

private static HashSet<AuthSession> sessions = new HashSet<AuthSession>();

public static Guid Create(string login)

{

sessions.RemoveWhere(session => (DateTime.Now - session.LastTimeUsed).TotalMinutes > 15);

string faculty = string.Empty;

using (GenericRepository<User> storage = new GenericRepository<User>())

{

faculty = storage.Select(filter: user => user.Login == login)?.ToList().FirstOrDefault()?.Role1.Faculty;

}

Guid newToken = Guid.NewGuid();

sessions.Add(new AuthSession(login, faculty == null ? string.Empty : faculty, newToken));

return newToken;

}

public static string GetFacultyByToken(Guid? token)

{

sessions.RemoveWhere(session => (DateTime.Now - session.LastTimeUsed).TotalMinutes > 15);

if (token == null)

{

return null;

}

AuthSession authSession = sessions.FirstOrDefault(session => session.Token == token);

authSession?.UpdateTime();

return authSession?.Faculty;

}

public static Guid? GetToken(string login)

{

sessions.RemoveWhere(session => (DateTime.Now - session.LastTimeUsed).TotalMinutes > 15);

AuthSession authSession = sessions.FirstOrDefault(session => session.Login == login);

authSession?.UpdateTime();

return authSession?.Token;

}

}

## 3.3 Рівень користувацького інтерфейсу

Основне завдання цього рівня – надавати ергономічний інтерфейс користувачу відповідно до функціоналу, описаного в технічному завданні. Цей рівень просто повинен бути красивим, зручним і інтуїтивно зрозумілим. При цьому він повинен знати, яким чином взаємодіє з рівнем бізнес-логіки, як передає туди дані і як він їх звідти бере для відображення.

Cпроектовану систему вирішено реалізувати у якості SPA (Single Page Application), тому на рівні користувацького інтерфейсу реалізована технологія AJAX. Реалізація технології AJAX можлива завдяки прийому даних у форматі JSON та відправці серверу даних через параметризовані HTTP запити. Такий запит, наприклад, використано при авторизації користувача:

http://localhost:56270/Auth/Login?login=admn&password=diit

Написання інтерфейсу потребувало використання таких технологій:

* JavaScript;
* CSS 3;
* HTML 5;
* Bootstrap.

Усі компоненти користувацького інтерфейсу знаходяться на одній сторінці одночасно і скриваються, коли не потрібні. Це можливо завдяки використанню JavaScript. Також, розроблений скрипт дозволяє вставляти потрібні дані у потрібні блоки, наприклад, у таблиці звітів для декана.

Приклад обробки даних з рівня бізнес-логіки та їх відображення у таблиці звіту:

function fillRectorTable(token) {

//Getting info

var xmlhttp1 = new XMLHttpRequest();

var url = "http://localhost:56270/Reports/RectorTable?datetime=17.05.2016&token=" + token;

xmlhttp1.onreadystatechange = function () {

if (xmlhttp1.readyState == 4 && xmlhttp1.status == 200) {

var rows = JSON.parse(xmlhttp1.responseText);

var htmlToInsert = "";

for (i = 0; i < rows.length; i++) {

var j = parseInt(i);

htmlToInsert += '<tr>' +

insertTd(rows[j].Faculty) +

insertTd(rows[j].Speciality) +

insertTd(rows[j].License) +

insertTd(rows[j].GovOrderMin) +

insertTd(rows[j].GovOrderMax) +

insertTd(rows[j].CountOutOfCompetition) +

insertTd(rows[j].CountPriority1) +

insertTd(rows[j].CountPriority2) +

insertTd(rows[j].CountPriority3) +

insertTd(rows[j].CountPriority4) +

insertTd(rows[j].CountPriority5) +

insertTd(rows[j].CountPriority6) +

insertTd(rows[j].CountPriority7) +

insertTd(rows[j].CountPriority8) +

insertTd(rows[j].CountPriority9) +

insertTd(rows[j].CountPriority10to15) +

insertTd(rows[j].CountTotal) +

insertTd(rows[j].RecommendedByGovOrder) +

insertTd(rows[j].OriginalsGovOrder) +

insertTd(rows[j].OriginalsContract) +

'</tr>';

}

document.getElementById("mainTBody").innerHTML += htmlToInsert;

}

}

xmlhttp1.open("GET", url, true);

xmlhttp1.send();

1. ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ ПРОГРАМИ

Тестування програми – це перевірка відповідності між очікуваною та реальною поведінкою програми. Тестом називається набір вхідних і вихідних даних, які відповідають специфікації. Найвідомішими стратегіями тестування є стратегії чорного та білого ящика.

Стратегією «білого ящика» називають стратегію, яка передбачає, що внутрішній устрій, структура та реалізація системи, що тестується, відомі. За допомогою цієї системи здійснюється перевірка логіки та архітектури програми.

Стратегія «чорного ящика» – це тестування, основане на специфікаціях або тестування зовнішньої поведінки програми, тобто внутрішній устрій програмного продукту невідомий. Ця стратегія тестування працює виключно з зовнішніми інтерфейсами системи, що тестується. Основними цілями тестування «чорним ящиком» є виявлення:

* неправильної реалізації або відсутніх функцій;
* помилок пов’язані з інтерфейсом;
* помилок в структурах даних або в організації доступу до зовнішніх БД;
* помилок поведінки або недостатньої продуктивності системи.

Для забезпечення найбільшого тестового покриття коду програми часто використовують комбінацію стратегій білого та чорного ящика.

* 1. Тестування методом «білого ящика»

Серед методів тестування «білого ящику» одним з найбільш ефективних є метод покриття умов і рішень. Метод покриття умов полягає в тому, що підбір тестів виконується таким чином, щоб кожна умова (елементарне судження в умовних операторах) приймає як істинне так і хибне значення для проходження по обох гілках розгалуження. Метод покриття рішень вимагає такої кількості тестів, щоб при виконанні їх усіх по кожній траєкторії, що з'єднує сусідні елементи блок-схеми обчислення пройшло хоча б один раз. Це означає, що кожне рішення повинно приймати як істинні, так і помилкові значення. Саме це забезпечує використання всіх шляхів, що виходять з точок розгалуження.

Вхідними даними для тестування програми методами білого ящика є специфікації функції та текст програми. Таким чином далі буде надаватися:

* текст методу, який тестуємо;
* специфікація;
* тестування методом покриття умов;
* тестування методом покриття рішень.

Метод: RectorTable

Текст методу:

/// <summary>

/// GET-запит для отримання звіту для ректора

/// </summary>

/// <param name="token">Токен авторизації</param>

/// <param name="dateTime">Дата та час формування звіту</param>

/// <param name="faculty">Назва факультету для якого треба відобразити звіт (null якщо для всього університету)</param>

/// <returns>Таблицю зі статистикою вступної кампанії для ректора</returns>

public ActionResult RectorTable(string dateTime, Guid? token)

{

string faculty = AuthSessions.GetFacultyByToken(token);

if (faculty != "Усі")

{

Response.StatusCode = 403;

Response.End();

return null;

}

List<EdeboStorageTable> result = null;

DateTime dateForFilter;

if (dateTime != null && dateTime != "undefined")

{

dateForFilter = DateTime.ParseExact(dateTime, "dd.MM.yyyy HH:mm:ss", CultureInfo.InvariantCulture);

}

else

{

dateForFilter = storage.Select().Max(columns => columns.DateTimeOfUpdate);

}

result = storage.Select(filter: columns => columns.DateTimeOfUpdate == dateForFilter,

orderBy: storage => storage.OrderBy(orderedStorage => orderedStorage.Faculty).ThenBy(orderedStorage => orderedStorage.Speciality)).ToList();

//якщо не отримали дані, то видаємо помилку 400

if (result == null || result.Count == 0)

{

Response.StatusCode = 400;

Response.End();

return null;

}

//мапінг даних до viewmodel

List<ForRectorViewModel> forRectorList = new List<ForRectorViewModel>();

foreach (var tuple in result)

{

forRectorList.Add(new ForRectorViewModel()

{

CountOutOfCompetition = tuple.CountOutOfCompetition,

CountPriority1 = tuple.CountPriority1,

CountPriority2 = tuple.CountPriority2,

CountPriority3 = tuple.CountPriority3,

CountPriority4 = tuple.CountPriority4,

CountPriority5 = tuple.CountPriority5,

CountPriority6 = tuple.CountPriority6,

CountPriority7 = tuple.CountPriority7,

CountPriority8 = tuple.CountPriority8,

CountPriority9 = tuple.CountPriority9,

CountPriority10to15 = tuple.CountPriority10to15,

CountTotal = tuple.CountTotal,

Faculty = tuple.Faculty,

GovOrderMax = tuple.GovOrderMax,

GovOrderMin = tuple.GovOrderMin,

License = tuple.License,

OriginalsContract = tuple.OriginalsContract,

OriginalsGovOrder = tuple.OriginalsGovOrder,

RecommendedByGovOrder = tuple.RecommendedByGovOrder,

Speciality = tuple.Speciality

});

}

Response.StatusCode = 200;

string returnedDate = dateForFilter.ToString("dd.MM.yyyy HH:mm:ss");

//конвертація у JSON

return Json(new { returnedDate, forRectorList }, JsonRequestBehavior.AllowGet);

}

Специфікація: метод призначено для формування та видачі звіту для ректора.   
 Вхідні дані: string dateTime, Guid? token.   
 Вихідні дані: List<ForRectorViewModel> forRectorList, string returnedDate, string statusCode.   
 Тест 1

Перевірка обробки неправильного параметру dateTime.

Вхід: dateTime = “05.04.2016 11:00:00”; token = “da04905f-6b46-4dc9-a24a-92de3557349f”

Вихід: forRectorList = null, returnedDate = null, statusCode = 403;

Тест 2

Перевірка обробки неправильного параметру token.

Вхід: dateTime = “04.06.2016 12:00:00”, token = “da04905f-6b46-4dc9-a24a-92de3557349f”

Вихід: результат заповнення таблиці звіту відображено на рис 4.2, returnedDate = 04.06.2016 12:00:00, statusCode = 200;

Рисунок 4.2 – Звіт ректора, заповнений правильними даними

Тест 3

Перевірка обробки правильних вхідних даних.

Вхід: dateTime = “04.06.2016 12:00:00”; token = “фырфырфырфыр-6b46-4dc9-a24a-92de3557349f”

Вихід: forRectorList = null, returnedDate = null, statusCode = 400;

Таблиця 4.1 – Відображення покриття тестами умов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер  теста | Умова | | | |
| if (faculty != "Усі") | | if (dateTime != null && dateTime != "undefined") | |
| **+** | **-** | **+** | **-** |
| 1 | + | - | - | + |
| 2 | - | + | - | - |
| 3 | + | - | + | - |

* 1. Тестування методом «чорного ящика»

Тестування «чорного ящика» повинне виявити усі недоліки та помилки зовнішньої поведінки програми. Основними техніками тест-дизайну для «чорного ящика» є розбиття даних на еквівалентні класи, аналіз граничних умов, складання таблиці рішень, функціональні діаграми, припущення про помилку.

Для даного проекту є неможливим використання розбиття даних на еквівалентні класи та аналіз граничних умов, тому що відсутня логіка яка працює з діапазонами даних. Таким чином для тестування «чорним ящиком» був використаний метод припущення про помилку.

Метод: GetDates

Текст методу:

public ActionResult GetDates(string faculty, Guid token)

{

string dbFaculty = AuthSessions.GetFacultyByToken(token);

if (dbFaculty != faculty && dbFaculty != "Усі")

{

Response.StatusCode = 403;

Response.End();

}

IEnumerable<EdeboStorageTable> table = null;

if (faculty == "Усі")

{

table = storage.Select();

}

else

{

table = storage.Select(columns => columns.Faculty == faculty);

}

IEnumerable<string> dateTimeList = table.GroupBy(columns => columns.DateTimeOfUpdate).Select(grp => grp.First()).Select(sel => sel.DateTimeOfUpdate.ToString("dd.MM.yyyy HH:mm:ss")).OrderByDescending(x => x);

return Json(dateTimeList, JsonRequestBehavior.AllowGet);

}

Специфікація: метод призначено для отримання списку дат звітів для певного факультету.

Вхідні дані:  string faculty, Guid token.

Вихідні дані: IEnumerable<string> dateTimeList, string statusCode.

Тест 1

Перевірка обробки неправильного параметру faculty.

Вхідні дані: faculty = “ТК”, token = “da04905f-6b46-4dc9-a24a-92de3557349f”.

Вихідні дані: dateTimeList = null, statusCode = 400.

Тест 2

Перевірка обробки неправильного параметру token.

Вхідні дані: faculty = “Технічна кібернетика”, token = “фівфівфівфів6b46-4dc9-a24a-92de3557349f”.

Вихідні дані: dateTimeList = null, statusCode = 403.

Тест 3

Перевірка обробки правильних параметрів.

Вхідні дані: faculty = “Технічна кібернетика”, token = “ da04905f-6b46-4dc9-a24a-92de3557349f ”.

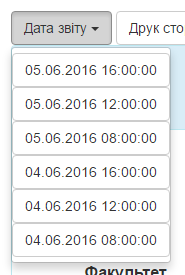
Вихідні дані: список дат відображено на рис.4.3 , statusCode = 200.

Рисунок 4.3 – Список дат звіту

4.3 Налагодження програми

Налагодження – це процес виявлення помилок програми та їх усунення. Налагодження спрямоване на встановлення точної природи відомої помилки, а потім на виправлення цієї помилки.

Налагодження програми буде проводитись з використанням методу індукції та методу просування від місця виникнення помилки до місця помилки. Таким чином, отримаємо налагоджену програму.

Відлагодження UI частини проекту відбувалася методом точок зупинок (breakpoints). Сама процедура виконувалася у браузері Google Chrome та за допомогою його вбудованих інструментів для розробника.

Інтерфейс для відлагодження у Google Chrome є дуже зручним, зрозумілим та інтуїтивним (рис 4.4). Розділений на три зони:

* + зона вихідних файлів, у ній знаходяться всі підключені до сторінки файли, включаючи JS / CSS;
  + зона тексту, у ній знаходиться текст файлів;

зона інформації та контролю, що містить багато метаінформації.

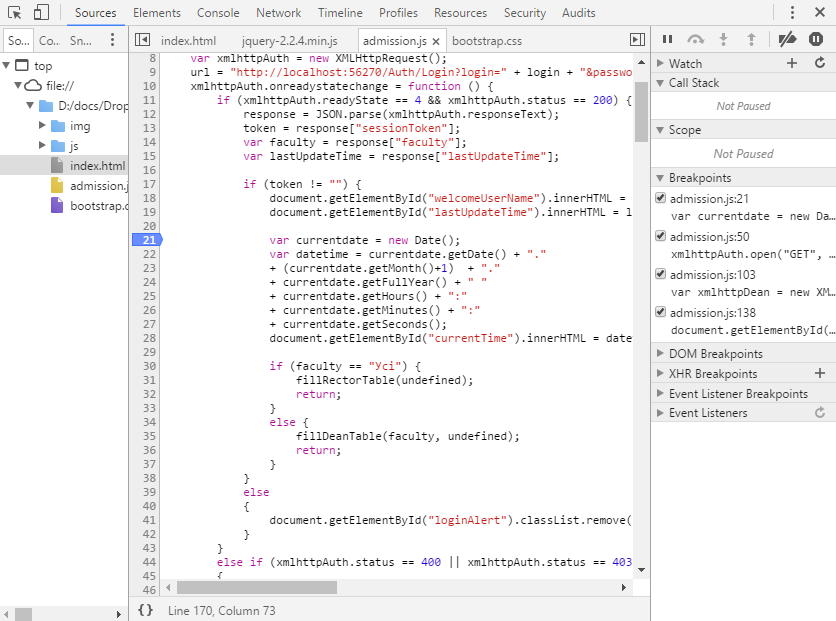
* + 

Рисунок 4.4 – Інтерфейс Google Chrome для налогодження роботи веб-сторінок

У зупиненому коді можна подивитися поточні значення змінних, виконувати команди тощо, загалом – налагоджувати його. Також, можна переглядати значення змінних на момент виконання останньої команди.

Інструментарій VisualStudio у плані відлагодження кращий, хоча у даному проекті використовувався той же функціонал, що і у Google Chrome.

Найбільшими проблемами при відлагодженні GUI були обробка різних статусів запиту та керування правами доступу. Далі детальніше.

На стороні клієнту спочатку була лише обробка 200 коду (OK). Оскільки метою було не просто сповіщення про помилку, а й повідомлення чому вона сталася, було вирішено скористатися й іншими доступними кодами, а саме:

* + 201 (Created – Створено) – при вдалому запиті на реєстрацію.
  + 400 (Bad Request – Неправильний запит) – при невірному форматі даних, що відправляються на сервер.
  + 403 ([Forbidden](https://uk.wikipedia.org/wiki/HTTP_403" \o "HTTP 403) – Заборонено) – при спробі отримати дані, що не доступні для користувача, наприклад, спроба отримати звіт не того факультету, для якого створено обліковий запис.
  + 406 (Not Acceptable – Неприйнятний запит) – коли формат відправлениних на сервер даних вірний, але невірний за змістом.

Оскільки з першої спроби все написати правильно не вийшло, було використано відлагодження на рядках, відповідних за прийняття відповіді на запит (рис. 4.5).

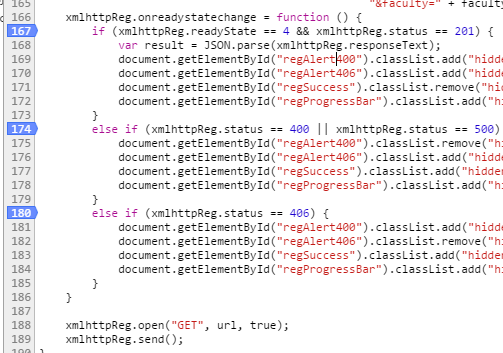
Таким чином вдалося проконтролювати та настроїти правильну роботу динамічної видачі сповіщень на AJAX запит

Рисунок 4.5 – Процес відлагодження програми

засобами Google Chrome

5 ОХОРОНА ПРАЦІ

Програмний продукт «Робоче місце ректора для аналізу статистики контингенту абітурієнтів на основі даних ЄДЕБО» призначене для допомоги у роботі секретарів приймальної комісії при складанні звітів, що необхідні ректору. Під час вступної кампанії секретарям приймальної комісії та адміністрації університету необхідні звіти, що відображають у зручній формі статистику поданих заяв абітурієнтами до ВНЗ. У звітах буде відображено кількість абітурієнтів денного або заочного відділення, що подали заяви до факультету, кількість оригіналів документів, бюджетних та контрактних місць, інформація про пільги абітурієнтів, пріоритетність заяв та загальну кількість поданих заяв за день.

Під час роботи з даною системою, користувач буде працювати з персональним комп’ютером, клавіатурою, монітором, або ноутбуком, тому слід зазначити правила користування електронно-обчислювальними машинами та відео-дисплейними терміналами.

5.1 Аналіз шкідливих та небезпечних виробничих факторів

Небезпечні виробничі чинники – це виробничі чинники, вплив яких на працівника в певних умовах призводить до травм, гострого отруєння або іншого раптового різкого погіршення здоров'я або до смерті [13].

До різкого погіршення здоров'я можна віднести отруєння, опромінення, удар електрострумом, тепловий удар та ін.

До небезпечних факторів відносяться електронебезпека та пожежонебезпека.

Шкідливі виробничі фактори – фактори, тривалий вплив яких на працюючого у визначених умовах приведе до захворювання, зниження працездатності i (або) негативного впливу на здоров'я нащадків [14]. У залежності від рівня і тривалості впливу шкідливі фактори можуть класифікуватися і як небезпечні.

Згідно санітарно-гігієнічним вимогам [15] умови праці ректора або члена приймальної комісії, який працює з робочим місцем ректора для аналізу контингенту абітурієнтів на ЕОМ, повинні відповідати I або II класу.

Одним з небезпечних факторів можуть бути пожежі. Вони становлять особливу небезпеку для життя людини, і можуть призвести до великих матеріальних втрат. Джерелами загоряння можуть виявитися електронні схеми ЕОМ, елементи, що перегрілися, які здатні викликати загоряння пальних матеріалів.

Для всіх споруджень та приміщень, у яких експлуатуються система управління процесами програмної інженерії на ЕОМ, повинна бути визначена категорія по вибухонебезпечній і пожежній безпеці відповідно до [13], і клас зони згідно ППЕ [16]. Відповідні позначення повинні бути нанесені на вхідні двері приміщення.

Також одним з джерел небезпеки при роботі з робочим місцем ректора для аналізу контингенту абітурієнтів на ЕОМ для ректора або члена приймальної комісії є ураження електричним струмом, викликані дотиком до оголених місць струмоведучих частин устаткування, або частин, що знаходяться під напругою.

Вплив струму на людину проявляється по-різному, в залежності від його величини:

* струм до 0,6 мА не відчувається людиною;
* струм завбільшки 6 мА призводить до скорочення м'язів тієї частини, тіла, що піддалася його впливу;
* струм 50 мА вражає органи дихання та серцево-судинну систему;
* при досягненні струму порога 100мА настає фібриляція серця і, потім, його зупинка;
* при впливі на тіло людини струму в 3 – 4 А виникає обвуглювання ділянок тіла.

За способом захисту людини від ураження електричним струмом відео термінали, ЕОМ, периферійні пристосування ЕОМ і оснащення для обслуговування, ремонту і налагодження ЕОМ повинні відповідати I класу захисту. Вимоги електричної і механічної безпеки і методи іспитів чи повинні бути заземлені.

5.2 Організаційні і технічні заходи щодо облаштування робочого місця.

Для якісної і зручної роботи з робочим місцем ректора для аналізу контингенту абітурієнтів на ЕОМ, необхідне проведення проектних заходів: правильна облаштованість, належне дотримання ергономічних характеристик основних елементів робочого місця [14], санітарно-гігієнічних вимог [15, 17] і т.п.

Облаштованість робочого місця користувача при роботі з підсистемою повинне забезпечувати відповідність всіх елементів робочого місця і їхнього розташування.

5.2.1 Вимоги до розташування робочих місць

Площа, яка виділена для одного робочого місця з ВДТ чи ЕОМ, повинна складати не менш 6 м2 , а обсяг – не менш 20 м3.

Робочі місця з ВДТ щодо світлових прорізів повинні розташовуватися так, щоб природне світло падало з боку, переважно ліворуч.

При розташуванні робочих місць із ВДТ необхідно дотримувати наступних вимог:

* робочі місця з ВДТ розташовуються на відстані не менш 1 м від стін зі світловими прорізами;
* відстань між бічними поверхнями ВДТ повинне бути не менш 1,2 м;
* відстань між тильною поверхнею ВДТ і екраном іншого ВДТ повинне бути не менш 2,5 м;
* прохід між рядами робочих місць повинний бути не менш 1 м.

Висота робочої поверхні столу для ВДТ повинна бути в межах 680 – 800 мм, а ширина - забезпечувати виконання належних операцій.

Розміри столу, що рекомендуються: висота – 725 мм, ширина ~ 600 – 1400 мм, глибина – 800 – 1000 мм.

Робочий стіл для ВДТ повинний мати простір для ніг висотою не менш 600 мм, шириною не менш 500 мм, глибиною на рівні колін не менш 450 мм, на рівні витягнутої ноги ~ не менш 650 мм,

Можливість обертання екрана ВДТ навколо горизонтальної і вертикальної осі.

5.2.2 Вимоги до організації робочого місця

При роботі з робочим місцем ректора для аналізу контингенту абітурієнтів на ЕОМ необхідно виконувати вимоги до організації робочого місця, що наведені нижче.

До самостійної роботи допускаються особи, які пройшли медичний огляд, навчання за професією, вступний інструктаж з охорони праці на робочому місці.

Конструкція робочого місця робітника при роботі з ЕОМ [10] (при роботі сидячи) повинна забезпечувати підтримку оптимальної робочої пози з наступними ергономічними характеристиками:

* стопи ніг – на підлозі чи на підставці для ніг;
* стегно – у горизонтальній площині;
* передпліччя – вертикально;
* лікті під кутом 70о – 90о до вертикальної площини;
* зап'ястя – зігнуті під кутом не більш 20о щодо горизонтальної площини;
* нахил голови 15о – 20о відносно вертикальної площини.

Дисплей та клавіатура мають розташовуватися на оптимальній відстані від очей користувача, але не ближче 600 мм, з урахуванням розміру алфавітно-цифрових знаків та символів.

Розташування екрану відео терміналу має забезпечувати зручність зорового спостереження у вертикальній під кутом ±30° від лінії зору працівника.

Клавіатуру слід розміщувати на поверхні столу або на спеціальній, робочій поверхні, яка регулюється за висотою поверхні окремо від столу на відстані 100 – 300 мм від краю, ближчого до працівника, що працює з робочим місцем ректора для аналізу контингенту абітурієнтів. Кут нахилу клавіатури має бути в межах 5 – 15°.

5.3 Небезпеки пов’язані з шумом та освітленням робочої зони

5.3.1 Шум, пов'язаний з відеодисплейним терміналом

Деякі відеодисплейні термінали (далі – ВДТ) є потенційними джерелами цілого ряду звуків, що містять Шум, пов'язаний з відеодисплейним терміналом

Деякі відеодисплейні термінали (далі – ВДТ) є потенційними джерелами цілого ряду звуків, що містять як коливання, які можна почути без спеціальних приладів, так і коливання ультразвукового діапазону, які можна почути тільки використовуючи спеціальні для цього прилади. Цей шум справляє негативний вплив на функціональний стан користувачів. Робота з робочим місцем ректора вимагає використання дисплею.

Відомо, що шум несприятливо діє на людину, особливо при тривалому впливі. У користувача, діяльність якого пов'язана з переробкою інформації, що часто супроводжується елементами творчості, це виражається у зниженні розумової працездатності (наприклад, швидкість обробки тексту зменшується на 10 – 15%, зростає кількість помилок), у прискоренні розвитку зорового втомлення, зміні відчуття кольорів, підвищенні витрати енергії (приблизно на 17%), появі головного болю, розвитку безсоння, послабленні уваги та ін.

Шум може бути фактором, що сприяє розвитку стресу. Відзначено взаємозв'язок між скаргами на шум від ВДТ, з одного боку, та емоційними порушеннями і поганим настроєм – з другого. Вплив шуму на вегетативну нервову систему може проявлятися при рівнях, близьких до припустимого, і призводити до порушення периферійного кровообігу за рахунок спазму капілярів шкіри та слизових оболонок, а також до інших негативних наслідків.

Для вимірювання шуму застосовують різні шумо-вимірювачі, частотні аналізатори та інші спеціальні прилади. Частотні аналізатори служать для виділення будь-якої смуги частот для подальшої направленої корекції шуму як за об'єктивними показниками приборів, так і згідно з суб'єктивним сприйняттям користувача, який працює в цьому середовищі. Вимірювання шуму на робочих місцях ректора або секретаря приймальної комісії здійснюється згідно з ДСН 3.3.6-037-99.

Нормованими параметрами шуму на робочих місцях є рівні середньоквадратичних звукових тисків (дБ) та рівні звука (дБА), що вимірюються по шкалі «А» шумовимірювача. Останні найбільш близькі до фізіологічного сприйняття людиною.

Згідно з ДСН 3.3.6-037-99 шум у приміщенні, де виконують роботу, пов'язану з виробленням концепцій, створенням нових програм, викладацькою роботою, творчістю, не повинен перевищувати 40 дБА. Праця керівників виробництва, пов'язана з контролем групи людей, що виконують переважно розумову роботу, не повинна супроводжуватися шумом вище 50 дБА. Висококваліфікована розумова робота, що вимагає зосередженості, може проводитись у приміщеннях, де рівень шуму не перевищує 55 дБА. Під час виконання розумової роботи за особистим графіком з інструкцією (операторська та близькі до неї види діяльності) і точних зорових робіт рівень шуму не повинен перевищувати 65 дБА.

Сумарний вплив численних джерел шуму у приміщенні у результаті багаторазового відбиття звукових хвиль може значно перевищити енергію прямого звука від тих же джерел. Шум від окремих приладів не повинен перевищувати фоновий більше ніж на 5 дБ.

Найчастіше рівні акустичного випромінювання, які виходять від ВДТ, охоплюють діапазон частот від 6,3 до 40 кГц. Домінуючими є частоти від 16 до 40 кГц, які пов'язані з частотою горизонтальної розгортки. Шум, можливо, виникає у осерді перетворювача горизонтальної розгортки. Не виключено, що всередині ВДТ існують вторинні джерела шуму.

Рівні звукового тиску на відстані приблизно 50 см від багатьох ВДТ у напрямі максимуму випромінювання знаходяться у межах від 30 до 68 дБ (середнє значення приблизно 51 дБ).

В діапазоні 16 – 20 кГц максимальний зареєстрований рівень тиску склав 61 дБ (середнє значення приблизно 53 дБ).

Основні заходи боротьби з шумом:

* усунення або ослаблення причин шуму в самому його джерелі у процесі проектування;
* використання засобів звукопоглинання;
* раціональне планування виробничих приміщень.

5.3.2 Параметри освітлення робочого місця та робочого приміщення

Результати досліджень показали, що виникнення одного з істотних джерел негативних фізіологічних впливів на користувачів ВДТ пов'язане з дискомфортними зоровими умовами через неправильно спроектоване освітлення: пряма та відбита від екранів близькість, вуалюючі відбиття, несприятливий розподіл яскравості в полі зору, невірна орієнтація робочого місця відносно світлових отворів. Погана якість символів, що представлені на екрані, також може викликати зоровий дискомфорт, бути стресовим фактором та ін.

Вимоги до освітлення для візуального сприймання користувачами інформації з двох різних носіїв (з екрана ВДТ та паперового носія) різні. Надто низький рівень освітленості погіршує сприймання інформації при читанні документів, а надто високий призводить до зменшення контрасту зображення знаків на екрані. Тому в полі зору користувача має бути забезпечений відповідний розподіл яскравості. Відношення яскравості екрана ВДТ до яскравості оточуючих його поверхонь не повинно перевищувати у робочій зоні 3:1.

Наближено можна вважати, що при 10%-ому зменшенні освітленості працездатність знижується на 1%. Коли за характером роботи вимагається комбінація цих двох носіїв інформації, освітленість можна варіювати від 300 до 700 лк, причому чим рідшою є зміна полів зору в процесі роботи (з екрана на документ та навпаки), тим вищим може бути рівень освітленості. 300 – 500 лк - оптимальна освітленість робочих приміщень для роботи з відео-терміналами. Стрибки яскравості при зміні полів зору мають бути мінімальними, тобто інтенсивність освітлення поверхні, де знаходяться рукописи та документи, не повинна перевищувати яскравості екрана дисплея.

Природне освітлення повинно здійснюватися у вигляді бічного освітлення та відповідати нормальним рівням за ДБН В.2.5-28-2006 «Естественное и искусственное освещение. Нормы проектирования».

При природному освітленні слід передбачити наявність сонцезахисних засобів, що знижують перепади яскравостей між природним світлом та свіченням екрана ВДТ. З цією метою можна використовувати плівки з металізованим покриттям або жалюзі з вертикальними ламелями, що регулюються.

Розташовувати робоче місце користувача, обладнане ВДТ, необхідно таким чином, щоб в поле зору користувача не потрапляли вікна або освітлювальні прилади; вони не повинні знаходитися й безпосередньо за його спиною. Слід добиватися зменшення відбиття на екрані від різних джерел штучного та денного світла. Коли штучне освітлення змішується з природним, рекомендується використовувати лампи, за спектральним складом найбільш близькі до сонячного світла. Бажано обирати світильники з розсіювачами, а всі блискучі деталі освітлювального обладнання, які можуть потрапити в поле зору, мають бути замінені на матові.

Штучне освітлення у приміщеннях з відео-дисплейними терміналами треба здійснювати у вигляді комбінованої системи освітлення з використанням люмінесцентних джерел світла у світильниках загального освітлення. Ці джерела світла слід розташовувати над робочими поверхнями у рівномірно-прямокутному порядку. Для запобігання засвітленню екранів ВДТ прямими світловими потоками, лінії світильників повинні бути розташовані з достатнім бічним зміщенням відносно рядів робочих місць або робочих зон, а також паралельно до світлових отворів. Бажане розміщення вікон з одного боку робочих приміщень. При цьому кожне вікно повинно мати світлорозсіюючі штори з коефіцієнтом відбивання 0,5 – 0,7.

На робочому місці має бути забезпечена рівномірна освітленість за допомогою переважно відбитого або розсіяного світлорозподілу. Світлових відблисків з клавіатури, екрана та від інших частин ВДТ у напрямку очей користувача не повинно бути. Для їх виключення необхідно застосовувати спеціальні екранні фільтри, захисні козирки або розташовувати джерела світла паралельно напрямку погляду на екран ВДТ з обох його сторін. Дискомфорт від відбиття світла знижується при збільшенні яскравості екрана та зниженні рівня навколишнього освітлення в рамках зазначених норм.

Пульсація освітленості люмінесцентних ламп, що використовуються, не повинна перевищувати 10%. При використанні місцевого джерела екран повинен мати можливість орієнтації у різних напрямках та бути обладнаним захисним пристроєм, який захищатиме від осліплення та відбитого відблиску, а також пристроєм для регулювання його яскравості.

Інформація, яку одержує користувач, генерується на екрані, а комфортність її сприймання залежить від чіткості символів. При обговоренні проблеми дискомфорту або негативних наслідків для здоров'я та ефективності роботи на ВДТ слід враховувати ряд параметрів. Ці параметри поділені на три групи:

* параметри, пов'язані з мигтінням;
* пов'язані з структурою символів що представляються на екрані;
* пов'язані з яскравістю символів що представляються на екрані.

Тривалість горизонтальної розгортки дорівнює 54 мкс (лінійна частота 15,6 кГц) і відповідає 230 – 300 рядкам на екрані (з урахуванням часу зворотного ходу променя і т. п.). При символьній матриці 7 на 9 пікселей та 80 символах у рядку час пробігу становить близько 0,1 мкс/піксель Необхідна для модуляції променя інформаційна частота становить біля 10 МГц та вище.

За даними ВООЗ, при натурних обстеженнях мигтіння спостерігалося на 12% ВДТ, 68% користувачів скаржилися на мигтіння. Сприймання мигтіння залежить не тільки від частоти регенерації екрана, але й від цілого ряду інших параметрів, таких, як полярність екрана, його яскравість, освітленість приміщення, ступінь осциляції, контраст, а також від використання центрального або периферійного зору та від індивідуальної чутливості.

## 5.4 Використання принтерів

Під час роботи з робочим місцем ректора для аналізу контингенту абітурієнтів ректору або члену приймальної комісії необхідно буде використовувати принтери для друкування звітів, тому необхідно дотримуватися норм експлуатації цих пристроїв.

Широко застосовуються лазерні, струминні та інші принтери. Категорія сторінкових принтерів представлена насамперед лазерними принтерами. В них для створення на світлочутливому барабані, що друкує «зарядовий малюнок», котрий відповідає документу (зображенню), використовується лазерний промінь. Світлодіодні принтери, які працюють з «світлодіодною гребінкою», вважаються альтернативою лазерним.

За даними Міністерства охорони здоров’я, під час експлуатації розмножувальних машин у повітря робочої зони виділяються озон, діоксид селену; робота розмножувальних машин супроводжується виникненням електростатичного поля, шуму, порушенням температурного режиму, швидкості руху та відповідної вологості повітря, наявністю інфрачервоного випромінювання.

Оксид вуглецю легко сполучається із залізом гемоглобіну, утворюючи карбоксигемоглобін, який призводить до порушення транспортування кисню. Основними ознаками хронічної інтоксикації, яка розвивається за тривалої дії невеликих концентрацій оксиду вуглецю, є функціональні розлади центральної нервової системи у вигляді астенії або вегетативної дисфункції з ангіодистонічним синдромом, що проявляється головними болями, серцебиттям і ниючими болями в ділянці серця, підвищеною дратівливістю, швидкою втомлюваністю, порушенням сну, погіршенням пам’яті, періодичним підвищенням артеріального тиску.

Виділення шкідливих речовин залежить від типу апарату і тонера. У паспортах ксерокопіювальних приладів не завжди зазначається матеріал фотопровідного проміжного носія, а склад конкретних тонерів становить промислову таємницю фірм-виробників, тож інформація з цього приводу є обмеженою. Все це створює певні труднощі в проведенні контролю за станом повітря робочої зони.

Лазерні та світлодіодні принтери належать до електрофотографічних пристроїв. Принцип їх дії заснований на використанні електростатичних зарядів, що призводить до виділення озону.

Підвищена концентрація озону може бути шкідливою для здоров'я. З цієї причини він був внесений у список речовин, максимальні значення концентрацій яких на робочих місцях обмежені та строго визначені. Він утворюється із звичайного кисню під впливом ультрафіолетових променів, електричних розрядів, дуже високих температур та подальшого швидкого охолодження, а отже, при роботі лазерних та світлодіодних принтерів.

Для виключення виділення принтером озону треба застосовувати озонові фільтри. Необхідно пам'ятати, що озоновий фільтр перестає діяти, якщо принтер завантажений мало або простоює без роботи. Отже. варто міняти його через проміжки часу, рекомендовані виготовлювачем.

Промені у лазерних принтерах невидимі. Щоб захистити користувачів від шкідливого для здоров'я випромінювання, лазерний блок герметизуєтъся. Тепер застосовуються лазери з невисокою інтенсивністю випромінювання. Якщо ж з метою підвищення продуктивності у принтері буде встановлений більш потужний лазер, то необхідно екранувати траєкторію променя.

Лазерні та світлодіодні принтери характеризуються підвищеним тепловиділенням, тому в них монтуються вентилятори. Вони піднімають у повітря пил, що складається з паперових волокон, які відірвались у процесі друку. При використанні спеціального паперу з пониженим виділенням пилових часток можна довести викиди пилу до мінімуму. Тому для лазерних та світлодіодних принтерів придатні тільки сорти паперу з дуже короткими волокнами, невисоким вмістом смоли та пилу.

Отже, при роботі з копіювальною технікою треба застосовувати додаткові заходи для оздоровлення повітряного середовища: встановлювати копіювальну техніку в окремих приміщеннях; контролювати вміст шкідливих речовин у повітрі; застосовувати вентиляцію, кондиціонери, нейтралізатори статичної електрики, заземлення (занулення) корпусів устаткування.

## 5.5 Дії при аварійних ситуаціях

Після закінчення роботи з робочим місцем ректора для аналізу контингенту абітурієнтів рктору або члену приймальної комісії необхідно:

* при раптовому припиненні подавання електроенергії вимкнути послідовно периферійні пристрої, ВТП, процесор і витягнути штепсель вилки з розетки;
* при виявленні ознак горіння (дим, запах горіння), вимкнути апаратуру, знайти джерело займання і вжити заходів для його ліквідації, повідомити керівництво чи системного адміністратора;
* в разі виникнення пожежі негайно повідомити пожежну частину, вжити необхідних заходів для евакуації людей згідно з планом евакуації і приступити до гасіння первинними засобами пожежогасіння.

Пожежі становлять особливу небезпеку для життя людини, і можуть призвести до великих матеріальних втрат. Джерелами загоряння можуть виявитися електронні схеми ЕОМ, елементи, що перегрілися, які здатні викликати загоряння пальних матеріалів.

Для всіх споруджень та приміщень, у яких експлуатуються робоче місце рктора для аналізу контингенту абітурієнтів на ЕОМ, повинна бути визначена категорія по вибухонебезпечній і пожежній безпеці відповідно до [15], і клас зони згідно ППЕ [16]. Відповідні позначення повинні бути нанесені на вхідні двері приміщення.

Будинки та ті їх частини, у яких розміщуються ЕОМ, повинні мати ступінь вогнестійкості не нижче II. Приміщення для обслуговування, ремонту і налагодження ЕОМ повинні відноситися:

* по пожежонебезпеки до категорії В – пожежонебезпечні приміщення, де розташовуються тверді пальні речовини (ТПР);
* по класу приміщення до категорії П ІІ [17].

Приміщення з ЕОМ повинні бути оснащені:

* системою автоматичної пожежної сигналізації відповідно до вимог переліку однотипних за значенням об'єктів, що підлягають устаткуванню автоматичними установками пожежогасіння і пожежної сигналізації;
* димовими пожежними оповіщувачами;
* переносними вуглекислотними вогнегасниками з розрахунку 2 шт., на кожні 20 м2 площі приміщення з обліком гранично припустимих концентрацій вогнегасної рідини відповідно до вимог правил пожежної безпеки в Україні.

Стіни приміщень з ЕОМ виготовляються з не пальних матеріалів. Підходи до засобів пожежогасіння повинні бути вільними.

При роботі робочим місцем ректора на ЕОМ можуть трапитися нещасні випадки, зв'язані з ураженням електричним струмом, викликаним дотиком до оголених місць струмоведучих частин устаткування, або частин, що знаходяться під напругою.

Вплив струму на людину проявляється по-різному, в залежності від його величини:

* струм до 0,6 мА не відчувається людиною;
* струм завбільшки 6 мА призводить до скорочення м'язів тієї частини, тіла, що піддалася його впливу;
* струм 50 мА вражає органи дихання та серцево-судинну систему;
* при досягненні струму порога 100мА настає фібриляція серця і, потім, його зупинка;
* при впливі на тіло людини струму в 3 – 4 А виникає обвуглювання ділянок тіла.

За способом захисту людини від ураження електричним струмом відео термінали, ЕОМ, периферійні пристосування ЕОМ і оснащення для обслуговування, ремонту і налагодження ЕОМ повинні відповідати I класу захисту. Вимоги електричної і механічної безпеки і методи іспитів чи повинні бути заземлені.

Лінія електромережі для живлення ЕОМ, периферійних устроїв ЕОМ і устаткування для обслуговування, ремонту і налагодження ЕОМ виконується як окрема групова, трьох провідна мережа, шляхом прокладки фазового, нульового робочого і нульового захисного провідників.

Підключення на розподільному щиті до одного контактного затиску нульового робочого і нульового захисного провідників заборонено.

Площа перетину нульового робочого і нульового захисного провідника в груповій трьох провідній мережі повинна бути не менш площі перетину фазового провідника.

У приміщенні, де одночасно експлуатується чи обслуговується більш п'яти персональних ЕОМ, на видному доступному місці встановлюється аварійний вимикач, за допомогою якого можливо зробити знеструмлення приміщення (за винятком освітлення).

Найбільш поширеними причинами електротравматизму є:

* + - * поява напруги там, де її в нормальних умовах бути не повинно (на корпусах устаткування, на металевих конструкціях споруд і т.д.), найчастіше це відбувається внаслідок пошкодження ізоляції;
      * можливість дотику до неізольованих струмоведучих частин за відсутності відповідних огороджень;
      * вплив електричної дуги, що виникає між токоведущей частиною і людиною в мережах напругою вище 1000 В, якщо людина виявиться в безпосередній близькості від струмоведучих частин;
      * інші причини: неузгоджені і помилкові дії персоналу, подача напруги на установку, де працюють люди, залишення установки під напругою без нагляду, допуск до робіт на відключеному електрообладнанні без перевірки відсутності напруги і т.д.

Ураження електричним струмом виникає в результаті:

* безпосереднього дотику до відкритих токоведучіх частин і проводів;
* дотику до металевих частин обладнання, яке випадково може опинитися під напругою;
* дотику до токоведучіх частин, ізоляція яких пошкоджена;
* дотику до токоведучіх частин за допомогою предметів з низьким опором ізоляції;
* помилковою подачею напруги під час ремонтів або оглядів;
* відсутність або пошкодження захисного заземлення;
* дії електричного струму через дугу;
* впливу крокової напруги.

Практика показує, що правильна організація робіт, сумлінне виконання правил безпечної експлуатації електроустановок споживачів, дисципліна і відповідальність керівників, і працюють – надійна основа безпечної роботи.

ВИСНОВКИ

Розробка даного дипломного проекту була зумовлена потребою приймальної комісії у автоматичному генеруванні звітів, які на протязі довгих років складалися вручну: член приймальної комісії тричі на день завантажував дані з ЄДЕБО у форматі .xls та за допомогою сортування та формул складав звіт. З’ясувавши потреби приймальної комісії під час кількох зустрічей був отриманий макет звіту та вимоги до базового функціоналу.

З технічної точки зору було обрано технології, що входять до одного «стеку технологій», тобто, між ними велика сумісність. Це допомогло концентруватися не на проблемах, що можуть з’явитися під час з’єднання різних технологій, а на виконанні поставленої задачі, розробці доцільної архітектури, досягненні кращого результату та максимально швидкому виконанні завдання.

Під час проектування проекту були повторені та уточнені знання з архітектури. Під час розробки проекту були повторені знання з роботи з  
web-додатками та вивчені нові методи та можливості роботи JavaScript та навички верстання сайтів.

Даний проект представляє цінність для університету тим, що автоматизує монотонну та однакову роботу, а значить члени приймальної комісії зможуть спокійніше працювати, зосередитися на інших, важливіших справах, ніж витрата часу на постійне сортування та складання звітів. До того ж, ректор може отримати та оновити інформацію будь коли без потреби звернення до третьої особи.

Архітектура та тема даної роботи залишають простір для творчості та збагачення функціоналу. До проекту можна легко додати модуль обробки даних чи аналітичну частину, яка займатиметься аналізом даних минулих років та будуватиме на цій основі прогноз на наступну вступну кампанію. Або можна складати статистику рівня освіченості студентів чи скласти діаграми щодо походження студентів. Також, проект може бути покращений з точки зору GUI, оскільки проектувався він для забезпечення базового функціоналу.

На завершення можна сказати, що проект відповідає потребам замовника, технологічно виконаний згідно потреб (навмисне не ускладнювався, проте має архітектуру для зручного додавання функціоналу) та має привабливий зовнішній вигляд, який відповідає потребам цільової аудиторії.

ЛІТЕРАТУРА

1. 0095.120606.ДП.2016.001 «Розробка робочого місця декана для аналізу контингенту абітурієнта за даними ЄДЕБО. Пояснювальна записка.» – 84 с.
2. Інструкція WEB-клієнту ЄДЕБО [Електронний ресурс] // Режим доступа до сайту: http://www.inforesurs.gov.ua/instr-edebo.html
3. Автономна система АВЕРС «Приймальна комісія» [Електронний ресурс] // Режим доступа до сайту: http://vuz.osvita.net/ua/asu-vnz/as-pryimalna-komisiia/
4. Magellan: модуль «Приемная комиссия» » [Електронний ресурс] // Режим доступа до сайту: http://www.quorus.ru/pages/modul-priemnaya-komissiya
5. 0095.120612.ДП.2015.001 «Розробка підсистеми інформаційного забезпечення з аналізу контингенту абітурієнтів університету за даними ЄДБО» – 98 с.
6. Основи MVC. З чого розпочати? [Електронний ресурс] // Режим доступа до сайту: <http://ruseller.com/net.tutsplus.com/tutorials/other/mvc-for-noobs/>
7. Архитектура REST [Електронний ресурс] // Режим доступа до сайту: https://habrahabr.ru/post/38730/
8. Що таке AJAX, його переваги та недоліки [Електронний ресурс] // Режим доступа до сайту: <http://promo.ingate.ru/seo-wikipedia/ajax/>
9. Багаторівнева архітектура в ASP.NET MVC [Електронний ресурс] // Режим доступа до сайту: http://metanit.com/sharp/mvc5/23.5.php
10. Рихтер Дж., CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Дж. Рихтер – М.: ДМК Пресс, 2014. – 824 с.
11. Современный учебник JavaScript [Електронний ресурс] // Режим доступа до сайту: https://learn.javascript.ru/
12. Об объектном подходе к программированию [Електронний ресурс] // Режим доступа до сайту: http://h-l-l.ru/publ/41-1-0-79
13. ДСТ 12.0.003-74. «ССБТ. Небезпечні і шкідливі виробничі фактори. Класифікація»
14. ДСН 3.3.6-037-99. «Державні санітарні норми виробничого шуму, ультразвуку та інфразвуку»
15. ДСН 3.3.6-042-99. «Державні санітарні норми мікроклімату виробничих приміщень»
16. ДСанПіН 2.2.2/2.4. 1340-03 «Гігієнічні вимоги до персональних електронно- обчислювальних машин та організації роботи»
17. ДСанПіН 3.3.2.007-98 «Державними санітарними правилами і нормами роботи з візуальними дисплейними терміналами електронно-обчислювальних машин»
18. НПАОП 0.00-4.12-05 «Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці»

ДОДАТКИ