**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**МІЖНАРОДНИЙ КЛАСИЧНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ПИЛИПА ОРЛИКА**

Економічно-технологічний факультет

Кафедра інженерних технологій

**КВАЛІФІКАЦІЙНА РОБОТА**

**другого (магістерського рівня) вищої освіти**

**на тему:**

# «Система підтримки прийняття рішень щодо вибору ВНЗ абітурієнтами»

зі спеціальності 123

«КОМП’ЮТЕРНА ІНЖЕНЕРІЯ»

Виконавець:

Мельникович Д.В.

Науковий керівник:

к.т.н., доц. Гайша О.О.

Миколаїв – 2025

ЗМІСТ

# ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

| | |
|---|---|
| ASP | Active Server Pages |
| CSS | Cascading Style Sheet |
| CSV | Comma Separated Values |
| GUI | Graphics User Interface |
| HTML | HyperText Markup Language |
| JSP | Java Server Pages |
| PC | Personal Computer |
| PHP | Personal Home Page, **P**HP **H**ypertext **P**reprocessor |
| SQL | Structured Queries Language |
| WWW | World Wide Web |
| XML | eXtensible Markup Language |
| БД | База даних |
| ВНЗ | Вищий навчальний заклад |
| ДПА | Державна підсумкова атестація |
| ЗМІ | Засоби масової інформації |
| ЗНО | Зовнішнє незалежне оцінювання |
| ІТ | Інформаційні технології |
| ОО | Об’єктно-орієнтований (-на, -не) |
| ООП | Об’єктно-орієнтоване програмування |
| ПЗ | Програмне забезпечення |
| ПК | Персональний комп’ютер |
| СУБД | Система управління базами даних |

# ВСТУП

Наявність вищої освіти у сучасному світі є однією із обов'язкових умов реалізації успішної кар'єри, особистого розвитку та за традиціями, що склалися у незалежній Україні протягом 1990-2000-х взагалі є майже обов'язковим елементом життя молодих громадян держави. Останні 5 років дещо змінили ситуацію, але здобуття вищої освіти все ще залишається важливим пріоритетом для середньостатистичної особи в Україні.

Вже під час навчання у випускному класі сучасні правила вступу вже вимагають від учня визначитися із обраним напрямком та записатися на відповідні предмети для здачі зовнішнього незалежного оцінювання (далі – ЗНО). Хоча в реальності велика частка молодих людей визначається із бажаним напрямком навчання ще до випускного класу (у 10 класі, чи, навіть, раніше). У будь-якому випадку, очевидно, що процес вибору напрямку, або навіть і конкретного ВНЗ, колись (рано, чи пізно) має відбуватися, а у цьому процесі учневі, та його рідним зацікавленим особам потрібно користуватися певною вхідною інформацією (хоча б переліком спеціальностей ВНЗ, необхідними дисциплінами, тощо), яка в цілому є розрізненою, розміщеною на різних сайтах окремих ВНЗ, у засобах масової інформації (далі – ЗМІ), і т.п. Відповідно, актуальною задачею є створення єдиного централізованого засобу, або, по-іншому, інформаційної системи, що могла би концентровано надавати усю необхідну вступникові інформацію.

Цей програмний засіб, може бути реалізований у вигляді веб-додатку, причому такий спосіб організації несе цілу низку переваг у порівнянні, наприклад, із настільним додатком, чи, тим більше, простою текстовою інструкцією (набором текстових документів). Таким чином, створення інформаційної системи з підбору вищого навчального закладу для абітурієнтів у вигляді веб-додатку є надзвичайно актуальною задачею, що має бути вирішена сучасними засобами веб-розробки та запропонована кінцевим споживачам для практичного використання.

Слід відмітити, що існують певні інформаційні продукти, що призначені для допомоги учневі у виборі конкретного ВНЗ для навчання. Однак, вони або мають простий (застарілий) характер, як наприклад, газета «Куда пойти учиться?» (Україна), або не повністю використовують інтерактивні та обчислювальні (зокрема, для підтримки процесу прийняття рішення про вступ до того, чи іншого ВНЗ) можливості сучасних інформаційних технологій (далі – ІТ), як, наприклад, Інтернет-ресурс схожої направленості http://osvita.ua.

Таким чином, дане дослідження є своєчасним та актуальним, а його проведення на даний час – необхідне та обґрунтоване.

Метою роботи є створення системи, що допомагатиме людині у процесі вибору майбутнього місця навчання.

Для цього слід вирішити наступні задачі проектування:

- розробити структуру бази даних для зберігання необхідної інформації для роботи системи;

- спроектувати інтерфейс користувача системи на основі застосування сучасних технологій (зокрема, динамічного завантаження інформації з серверу AJAX);

- виконати розробку алгоритмів роботи інформаційної ситсеми та їх реалізацію у програмних кодах до отримання кінцевого програмного продукту;

- провести аналіз та тестування роботи системи, зробити висновки по роботі.

Об’єктом дослідження є процес вибору майбутньої спеціальності та конкретного вищого навчального закладу абітурієнтом.

Предметом дослідження є інформаційна система, за допомогою якої абітурієнт може отримувати необхідну для такого вибору інформацію.

У якості методів дослідження можна назвати загальнонаукові методи аналізу і синтезу, та технології програмування: розподіленого, веб, баз даних, і т.п.

Практичне значення роботи полягає у створенні робото спроможного програмного продукту, використання якого дозволяє досягати поставленої мети.

# 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Особливості сучасного процесу вступу громадян до ВНЗ в Україні.

Вступ до вищого навчального закладу в усі часи був надзвичайно відповідальною процедурою, яка для успішного завершення вимагала мобілізації усіх наявних в учня та його рідних сил. Нерідко траплялися ситуації, коли сторона абітурієнта була на 100 % впевненою у високій якості вступної роботи (що намагалася довести і на апеляціях, і листами у Міністерство освіти і науки, тощо), але результату у вигляді вступу не було. Для уникнення таких проблемних ситуацій Міністерство освіти і науки України вело планомірну роботу по впровадженню контролю вступних знань абітурієнтів за межами ВНЗ, що фінально вилилося у механізм зовнішнього незалежного оцінювання, коли усі вступні випробування проводять не співробітники ВНЗ, а точніше, люди, взагалі не пов’язані з вузами.

Система, коли вступні іспити приймають не співробітники ВНЗ порівняно добре себе зарекомендувала, хоча, як це часто буває і з самими найкращими нововведеннями, принесла і деякі негативні тенденції. Зокрема, ЗНО стало обмежувальним фактором не тільки для осіб, що прагнуть навчатися за державний кошт, а й для тих, хто хотів вступати на контрактну форму навчання (що не зовсім зрозуміло, адже, на думку деяких юристів, обмежує в правах громадян України, які не можуть придбати існуючу на ринку платну освітню послугу). Однак, в цілому, за більшістю оцінок вступників, ЗНО є позитивним явищем на стику вищої та середньої освіти в Україні.

Відповідно, важливою обставиною тепер є перелік предметів, з яких абітурієнтові слід здавати ЗНО, щоби вступити на бажану спеціальність. Слід відмітити, що перелік цих предметів до того ж може ще і мінятися із року в рік, а записуватися на ці екзамени слід задовго до їх проведення (за декілька місяців). Ті учні, що пропустили термін реєстрації не можуть брати участь у ЗНО, що ставить вимогу дуже ретельного відбору предметів для

проходження тестування та завчасної на них реєстрації. Відповідно, для вибору цих предметів і стане в нагоді інформаційна система з підбору вищого навчального закладу для абітурієнтів.

## 1.2 Уточнена постановка задачі проектування.

Зважаючи на вищенаведене, можна сформулювати наступну уточнену задачу дипломного проектування: створити інформаційну систему з підбору вищого навчального закладу для абітурієнтів у вигляді розподіленого кросплатформенного веб-додатку на основі сучасних інструментальних засобів веб-розробки.

Робота має велике практичне значення, оскільки, за умови належного наповнення бази даних, буде корисною і може реально використовуватися абітурієнтами в процесі здійснення вибору вищого навчального закладу.

На основі цих відомостей можна переходити до виконання проектних рішень по розробці інформаційної системи з підбору вищого навчального закладу для абітурієнтів.

# 2. ПРОЕКТНІ РІШЕННЯ

## 2.1. Обгрунтування вибору засобів розробки веб-додатків.

Сучасні засоби для розробки Інтернет-ресурсів є настільки обширними, що розібрати їх усі в рамках дипломної роботи абсолютно не уявляється можливим. Тому в процесі вибору будемо орієнтуватися як на об’єктивні фактори (такі, як, наприклад, підтримка сучасних технологій програмування на зразок об’єктно-орієнтованого – ООП), так і суб’єктивні, але такі, що набули широкого впливу (як-то схильність окремих осіб до певних мов, чи, навіть, стилів програмування, що у результаті спричинює популярність відповідних засобів розробки на масовому рівні).

Таким чином, розглянемо найпоширеніші засоби розробки, що є популярними на даний момент у відповідній галузі.

В цілому увесь процес веб-розробки традиційно ділять на дві компоненти: front-end та back-end – рис. 2.1.

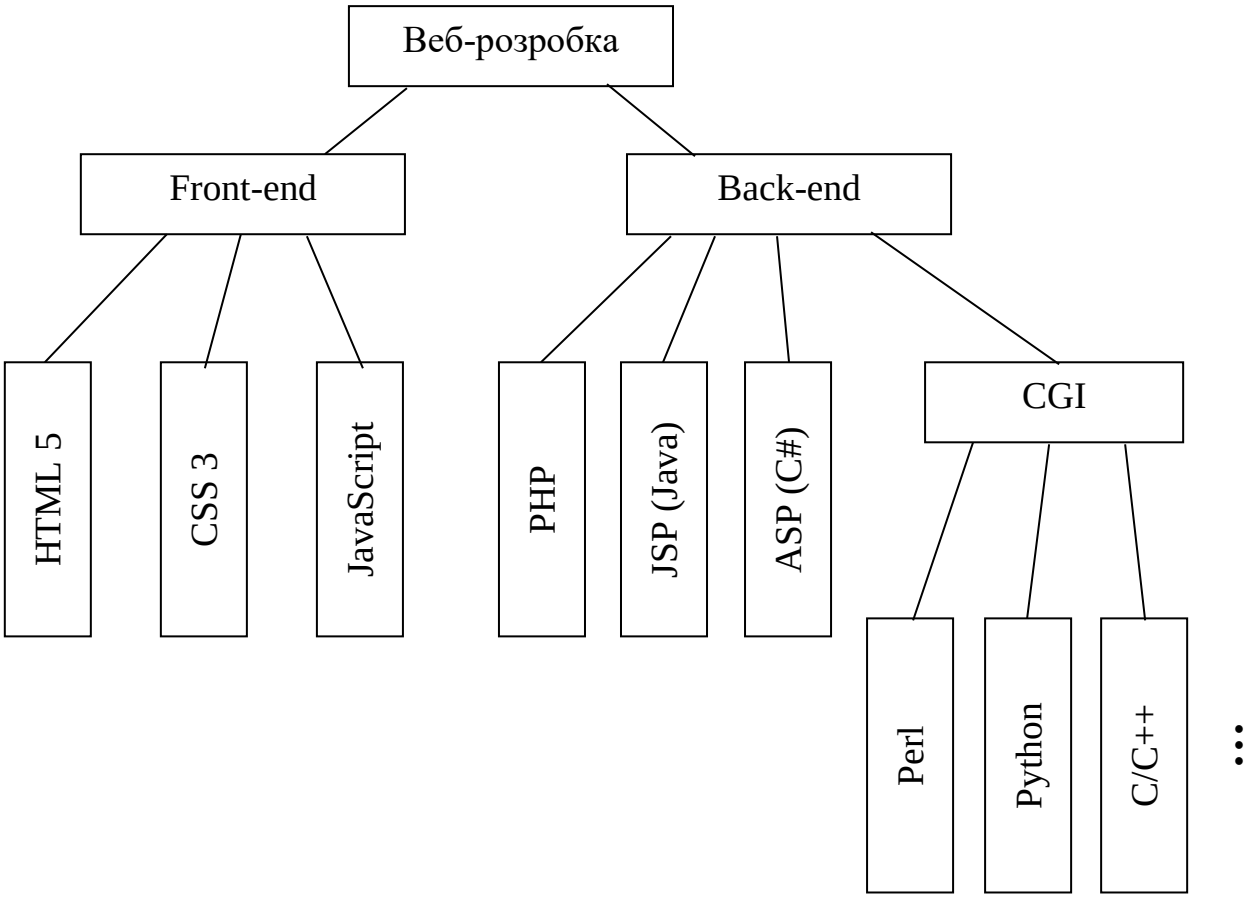



Рис. 2.1. Найпопулярніші засоби розробки сучасних Інтернет-ресурсів.

Спрощено кажучи, під «передньою» частиною – Front-end – у програмуванні мають на увазі те, що бачить користувач. Для настільних систем під цим мають на увазі інтерфейс, а під back-end'ом мають на увазі логіку роботи програми, яка теоретично (зокрема, так часто роблять у Linux-системах) взагалі може бути реалізована окремою програмою із текстовим консольним інтерфейсом. Задачею фронт-енду при цьому є зручний для широкого кола користувачів збір усіх вхідних даних, необхідних для виконання змістовних операцій консольною утилітою back-end'ом. У багатьох же випадках фронт та бек поєднані разом в рамках одного програмного забезпечення. Такою є ситуація для настільних систем.

Якщо конкретніше говорити про веб-програмування, то тут ситуація є максимально поляризованою (ще більше, ніж у описаному вище випадку написання фронт-енду одним програмістом для бек-енд-утиліти, створеної іншим розробником), оскільки користувач працює і «бачить» систему через браузер на одній віддаленій машині (на комп'ютері, що називається

клієнтом), а логіка додатку та практично уся супутня інформація розміщується на зовсім іншому комп’ютері-сервері.

Таким чином, усі програмні коди та висхідні тексти, що виконуються у браузері, тобто на стороні клієнта, відносяться до front-end частини. З іншого боку, все, що виконується на сервері, відноситься до back-end складової.

### 2.1.1 Вибір засобів фронт-енд розробки

Якщо глибоко не вдаватися у детальний аналіз, то коротко можна сказати, що набір інструментів для front-end розробки значно менший, ніж для беку. Серед них можна виділити:

- HTML – п’ятої версії (HTML5), найсучасніший стандарт мови гіпертекстової розмітки (HyperText Markup Language), де традиційно під гіпертекстом мають на увазі текст, оснащений гіперпосиланнями на інші частини документів та, можливо, супроводжуваний картинками, звуками, відео, і т.п.;

- CSS – третьої версії (CSS3) – правила запису стилів різноманітних елементів веб-сторінки, або, якщо говорити точно – каскадні таблиці стилів;

- JavaScript – найпотужніший інструмент front-end розробки, що дозволяє вивести html-сторінки на новий рівень практично повноцінних додатків, на зразок настільних, що активно реагують на дії користувача, передають і отримують інформацію, від нього.

Ці технології є безальтернативними, тому розглянемо їх докладніше.

1) Hyper Text Markup Language (HTML) - мова розмітки гіпертексту - призначена для написання усіх документів в мережі World Wide Web - WWW.

HTML документ - це текстовий файл, який має спеціальні мітки, що називаються тегами, які при затребуванні клієнтом передаються із комп’ютера-серверу на браузер клієнта і використовуються ним для відображення вмісту файлу па екрані комп'ютера.

За допомогою цих міток можна виділяти заголовки документа, змінювати колір, розмір і написання літер, вставляти графічні зображення і таблиці. Але основною перевагою гіпертексту перед звичайним текстом є можливість додавання до вмісту документа гіперпосилань - спеціальних конструкцій мови HTML, які дозволяють клацанням миші перейти до перегляду іншого документа. Таким чином «гіпертекст» означає «надтекст» - розвинення ідеї текстового документу, але із більш широкими можливостями.

Сама по собі мова HTML є різновидом більш загальної мови XML (eXtensible Markup Language), причому з одного боку звужує її властивості, а з іншого – навпаки деталізує. Так, у мові XML допустимим є використання будь-яких тегів, тобто з довільними іменами, в той час, як назви тегів мови HTML є цілком фіксованими, а інші слова окрім стандартних, використовувати не можна. З іншого боку, той вміст веб-сторінки, що розміщений біля конкретних тегів (точніше – між парою однакових тегів), набуває визначених стандартом властивостей (на відміну від тексту, що розміщений поруч із тегами XML, які самі по собі практично нічого не означають, а використовуються лише при наявності певних домовленостей про відображення й обробку того, чи іншого тегу) відображується браузером відповідно до суті цих тегів: якщо тегом є <B>, то текст зображуватиметься жирним, <i> - курсивом, і т.д.

HTML-документ має дві складові: власне текстова інформація, тобто дані, що складають вміст документа, і теги - спеціальні конструкції мови HTML, які використовуються для розмітки документа і керують його відображенням. Теги мови HTML визначають, в якому вигляді буде представлений текст, які його компоненти будуть виконувати роль гіпертекстових посилань, які графічні або мультимедійні об’єкти повинні бути включені в документ.

Графічна та звукова інформація, що включається в HTML-документ, зберігається в окремих файлах. Програми перегляду HTML-документів

називають браузерами, і вони інтерпретують теги розмітки і оперують текстом і графікою, розміщуючи їх на екрані відповідним чином. Для файлів, що містять HTML-документи використовується розширення .htm або .html.

У переважній більшості випадків теги використовуються парами. Пара складається з тега, що відкриває <ім'я_тега> і тега, що закриває </ім'я_тега>. Дія будь-якого парного тега починається з того місця, де зустрівся відкриваючий тег, і закінчується при зустрічі відповідного закриваючого тега. Часто пару, що складається з відкриваючого і закриваючого тегів, називають контейнером, а частину тексту між відкриваючим і закриваючим тегом - елементом.

Послідовність символів, яка утворює текст, може складатися з пробілів, табуляцій, символів переходу на новий рядок, символів повернення каретки, букв кириличного та латинського написань, знаків пунктуації, цифр, і спеціальних символів (наприклад #, +, $, @), за винятком наступних чотирьох символів, що мають в HTML спеціальний сенс: < (менше), > (більше), & (амперсанд) і «(лапки)». Якщо необхідно включити в текст будь-який із цих спецсимволів, то слід закодувати його особливою послідовністю символів:

```
<     -     &lt
>     -     &gt
&     -     &amp
"     -     &quot
```

Найпершим із тегів HTML розміщується однойменний тег <html>. Він завжди відкриває документ, так само, як тег </html> повинен неодмінно стояти в останньому його рядку. Ці теги позначають, що рядки, які між них знаходяться, представляють собою єдиний гіпертекстовий документ. Без цих тегів браузер або інша програма перегляду не в змозі ідентифікувати формат документа і правильно його інтерпретувати.

Далі, якщо говорити укрупнено, HTML-документ має дві частини: заголовок (head) і тіло (body), розташованих в наступному порядку:

```
<Html>
<Head> Заголовок документа </ head>
```

```
<Body> Тіло документа </ body>
</ Html>
```

Найчастіше в заголовок документа включають парний тег <title> ... </title>, що визначає назву документа. Багато програм перегляду використовують його як заголовок вікна, в якому виводиться документ. Програми, що індексують документи в мережі Інтернет, використовують назву для ідентифікації сторінки. Гарна назва повинна бути досить довгою для того, щоб можна було коректно вказати відповідну сторінку, і в той же час воно має міститися в заголовку вікна. Назва документа вписується між відкриваючим і закриваючим тегами title.

Тіло документа є обов'язковим елементом, так як в ньому розташовується весь матеріал веб-сторінки. Тіло документа розміщується між тегами <body> і </body>. Все, що розміщено між цими тегами, інтерпретується браузером відповідно до правил мови HTML дозволяють коректно відображати сторінку на екрані монітора.

Текст в HTML розділяється на абзаци за допомогою тега . Він розміщується на початку кожного абзацу, і програма перегляду, зустрічаючи його, відокремлює абзаци один від одного порожнім рядком. Використання закриваючого тега  є необов'язковим.

Якщо потрібно «розірвати» текст, перенісши його залишок на новий рядок, при цьому, не виділяючи нового абзацу, використовується тег розриву рядка <BR>. Він змушує програму перегляду виводити символи, що стоять після нього з нового рядка. На відміну від тега абзацу, тег <BR> не додає порожній рядок. У цього тега немає парного закриваючого тега.

Мова HTML підтримує логічне н фізичне форматування вмісту документа. Логічне форматування вказує на призначення даного фрагмента тексту, а фізичне форматування задає його зовнішній вигляд.

При використанні логічного форматування тексту браузером виділяються різні частини тексту відповідно до структури документа. Щоб відобразити назву, використовується один з тегів заголовка. Заголовки в

типовому документі поділяються за рівнями. Мова HTML дозволяє задати шість рівнів заголовків: h1 (заголовок першого рівня), h2, h3, h4, h5 і h6. Тема першого рівня має зазвичай більший розмір і насиченість в порівнянні з заголовком другого рівня. Приклад використання тегів заголовків:

```
<Hl> l. Назва розділу </hl>
<H2> l.l. Назва підрозділу </h2>
```

Теги фізичного форматування безпосередньо задають вид тексту на екрані браузера, наприклад пара   виділяє текст напівжирним шрифтом, <u> </u> задає підкреслення тексту, <font> </font> керує шрифтом тексту. Саме ці елементи вважаються застарілими, і замість них у специфікації HTML5 рекомендується користуватися стилями, що розглянемо нижче.

Тег <img> вставляє зображення в документ, причому так, якби воно було просто одним великим символом. Закриваючий тег для зображення не потрібний. Приклад застосування тега:

```
<Img src = "picture.gif">
```

Для створення гіпертекстового посилання використовується пара тегів <a> ... </a>. Фрагмент тексту, зображення або будь-який інший об'єкт, розташований між цими тегами, відображається у вікні браузера як гіпертекстове посилання. Активація такого об’єкта за допомогою щигля мишею призводить до завантаження у вікно браузера нового документа або до відображення іншої частини поточної Web-сторінки. Гіпертекстове посилання формується за допомогою формули:

```
<A href = "document.html"> посилання на документ </a>
```

Href тут є обов'язковим атрибутом, значення якого і є URL-адресою запитуваного ресурсу. Лапки в завданні значення атрибута href не обов'язкові (як і при завданні значень і для інших атрибутів будь-яких тегів, але використання лапок – подвійних, чи одинарних, є вкрай бажаним). Якщо задається посилання на документ на іншому сервері, то вид гіперпосилання такий:

```
<A href = "http://www.school.dn.ua/11.jpg">Фотографія 11-А
</ a>
```

Підсумовуючи можливості мови HTML, можна сказати, що за допомогою різних тегів можна малювати таблиці, форматувати текст, вставляти в документ зображення, відео-, звукові файли та інше. В процесі свого розвитку все більша увага приділялася тегу <style> та пов’язаними з ним каскадними таблицями стилів.

Каскадні таблиці стилів (Cascading Style Sheets, CSS) - це мова, яка містить набір властивостей для визначення зовнішнього вигляду документа. Специфікація CSS (CSS3 – на даний момент) визначає властивості і описову мову для встановлення зв'язку властивостей з елементами в документі. Розуміння таблиць стилів необхідно для додавання динамічного стилю сторінки. Під динамічним стилем (dynamic style) тут маємо на увазі модифікацією таблиці стилів, пов'язану з документом за допомогою сценарію.

На вузлі консорціуму W3C (www.w3.org) можна знайти останню інформацію про нововведення і елементах, підтримуваних таблицями стилів.

Таблиці стилів представляють собою абстракцію, в якій стиль документа визначається окремо від змісту або структури. Існує три методи додавання таблиць стилів в документ, доступних для Web-майстра - в цілому, з підвищенням рівня складності розширюються надані можливості з одночасним збільшенням ступеня абстракції. Перший метод полягає в використанні таблиці внутрішніх стилів (inline style sheet). Внутрішні стилі (inline styles) визначаються безпосередньо в елементі. Другий метод полягає в використанні таблиці глобальних стилів (global style sheet) для визначення стилю на початку документа. Третій, найбільш абстрактний і потужний метод полягає в використанні таблиці пов'язаних стилів (linked style sheet) для визначення стилю окремо в іншому документі.

Внутрішні стилі мало відрізняються від традиційного HTML. При використанні внутрішніх стилів зовнішній вигляд документа важко змінити.

Перевага даного методу полягає в скороченому обсязі розмітки і в тому, що HTML може бути більш повноцінно використаний для подання вмісту презентації. Використання таблиці глобальних стилів дозволяє більш ефективно відокремити представлення від вмісту, а також дає можливість швидкої і незалежної зміни стилю і обробки документа. Використання таблиці пов'язаних стилів дає більше переваг, представляючи вміст у вигляді набору сторінок або визначаючи цілий Web-вузол за допомогою одного файлу.

Термін cascading (каскадні) в назві CSS вказує на можливість злиття різних таблиць стилів для створення єдиного визначення стилю для елемента або для цілого документа. Це дозволяє проводити передбачуване злиття таблиці стилів Web-вузла з таблицею стилів документа і навіть з внутрішнім стилем.

Отже, внутрішній стиль (inline style) є по суті таблицею стилів для одиночного екземпляру елемента і його визначено безпосередньо у тегу елемента. Таблиця внутрішніх стилів визначається з використанням атрибута STYLE, а дані для атрибута визначаються за допомогою мови таблиці стилів. Нижче наведено код HTML, який збільшує шрифт відображення вмісту параграфа і вирівнює параграф по центру на жовтому фоні:

```
     <P STYLE = "font-size: 120%; text-align: center; background: yellow">
     Створює жовтий вирівняний по центру параграф з великим розміром шрифту.
     </ P>
```

Внутрішні стилі допомагають при вивченні мови внутрішніх стилів або за необхідності швидко змінити одиночний екземпляр елемента. Однак, внутрішні стилі не відповідають ідеології структурованого документа і погано працюють при необхідності зміни зовнішнього вигляду ряду елементів в документі, коли презентація та вміст розділені в повному обсязі. Для відділення стилю документа від його структури таблиця стилів повинна бути визначена в заголовку документа або як окремий файл, який пов'язаний з документом.

Наступною, більш ідеологічно вірною можливістю, є використання окремого тегу <STYLE> для завдання таблиць глобальних стилів, який зазвичай розміщується в заголовку документа. Розміщення усіх стилів документа в одному місці спрощує зміну режиму відтворення документа. Наведений нижче приклад таблиці стилів визначає зовнішній вигляд усіх параграфів в документі. Для зміни режиму відтворення параграфів потрібно змінити тільки елемент STYLE. Якби використовувалися внутрішні стилі, то довелося б міняти кожен параграф в документі окремо.

```
      <HTML>
        <HEAD>
           <STYLE TYPE = "text / css">
              P {font-size: 120%; text-align: center; background:
yellow}
           </ STYLE>
        </ HEAD>
        <BODY>
           <P> Все параграфи тепер більше і вирівняні по центру
на жовтому
              тлі. </ P>
        </ BODY>
     </ HTML>
```

Для зв'язку стилю з певним елементом використовується селектор (selector). У наведеному вище прикладі був створений простий селектор, який пов'язаний зі стилем у всіх параграфах. Можуть бути також визначені більш функціональні контекстуальні селектори, що описуються нижче у цьому розділі.

Ще одним із трьох способів завдання стилів є введення таблиці пов'язаних стилів (linked style sheet), яка знаходиться у зовнішньому файлі. Перевага використання таблиці пов'язаних стилів полягає в тому, що всі правила і стилі можуть бути визначені і вміщені в одному файлі, який може бути спільно використаний багатьма сторінками або навіть цілим Web-вузлом. При використанні таблиці пов'язаних стилів обробка всіх параграфів цілого Web-вузла може бути змінена в одному документі. Таблиця пов'язаних стилів може також підвищити продуктивність, оскільки вона кеширується локально на диску клієнта, окремо від документа, так що кожен документ має

менший розмір, а інформацію про стилі буде потрібно завантажити тільки один раз.

Для визначення таблиці пов'язаних стилів у заголовку документа розміщується тег <LINK>:

```
<HTML>
  <HEAD>
     <LINK REL = "stylesheet" TYPE = "text / css" HREF =
"fancy.css">
  </ HEAD>
  <BODY>
     <P>  This  document  uses  the  styles  specified  in
fancy.css. </ P>
  </ BODY>
</ HTML>
```

Атрибут REL визначає, що пов'язаний файл є таблицею стилів, а атрибут TYPE визначає тип MIME таблиці стилів. Атрибут HREF є покажчиком URL, що вказує на зовнішню таблицю стилів. Таблиця пов'язаних стилів повинна містити тільки контекстуальні правила і визначення стилю і не може включати код HTML.

Для створення таблиці стилів всередині документа використовується той же синтаксис, який використовувався при створенні таблиці пов'язаних стилів. Мова CSS складається з селекторів і правил подання (presentation rules). Селектори (selectors) визначають елементи, які пов'язані з певним правилом, а правила подання встановлюють методи обробки даних елементів.

CSS містить два типи селекторів: прості і контекстуальні. Простий селектор пов'язує елемент на основі його атрибутів або типу незалежно від контекстуального положення усередині інших елементів. Контекстуальні елементи є більш потужними - вони можуть зв'язати правило з контейнерами певного елемента, наприклад, усі теги <EM> всередині тегів <P>.

У базовій формі простий селектор може бути створений для зв'язку певного елемента, класу елементів або ідентифікатора (ID) з певним стилем. Наведений нижче код демонструє ряд простих селекторів і їх правил подання:

```
      <STYLE TYPE = "text / css">
        / * Change all H1s to red. * /
        H1 {color: red}
        / * Встановлює напівжирний шрифт всіх елементів з тегом
CLASS = "Special" boldface. * /
        .special {font-weight: bold}
        / * Розміщує елемент з ідентифікатором ID = special на
жовтому тлі.*/
        #special {background: yellow}
        / * Встановлює висновок елементів H1 з тегом CLASS =
"cool" з великим інтервалом.* /
        H1.cool {letter-spacing: 2px}
     </ STYLE>
```

Селектори можуть бути перераховані через кому, що дозволяє одночасно описати кілька селекторів:

```
/ * Встановлює для всіх заголовків однакові правила. * /
H1, H2, H3, H4, H5, H6 {color: red; background: yellow}
```

Контекстуальні селектори визначають ієрархію контейнерів, з якою повинен бути пов'язаний стиль. Ієрархія контейнерів визначається порядком елементів в списку через кому. Наприклад, наведений нижче оператор визначає правило для всіх елементів EM, що знаходятцься в елементі P:

```
P EM {color: blue}
```

Кожен селектор може посилатися на теги CLASS, ID або тип елемента. Нижче наведена більш складна версія контекстуального селектора:

```
      / * Будь-який елемент з CLASS = "cool", який знаходиться
всередині елемента LI з CLASS = "special" і далі перебуває
всередині елемента UL, буде використовувати даний стиль. * /
     UL LI.special .cool {font-weight: bolder; font-size: 120%}
```

Всі елементи контекстуального селектора є нечутливими до регістру - наприклад, .cool це те ж саме, що і .cOoL.

Псевдоклас (pseudo-class) складається з елементів одного типу, які задовольняють певному контекстуальному критерію. Наприклад, переглянуті елементи Anchor (посилання) представляють собою псевдоклас visited. Активні і не переглянуті посилання являють собою псевдокласи active і link, відповідно. Псевдоклас в таблиці стилів відділяється двокрапкою:

```
A: link {color: green}
: Link {color: green}
```

У другому прикладі опущено ім'я елемента (A), так як тільки посилання мають псевдоклас link. Псевдоклас може бути використаний так само, як клас або покажчик ID і є нечутливим до регістру.

На одні й ті ж самі елементи можуть посилатися кілька селекторів. CSS визначає послідовність каскадування (cascading order), яка використовується для вирішення проблем перекривання областей дії селекторів і правил. Послідовність каскадування об'єднує всі правила, які застосовуються до елементу, шляхом сортування на основі їх визначень. Наприклад, елемент Strong, що знаходиться в елементі H1, може мати правила подання, визначені селектором H1, селектором STRONG і контекстуальним селектором для елементів Strong всередині елементів H1. Параметр каскадування в CSS визначає порядок об'єднання даних трьох правил. Загалом, правило для більш конкретного контекстуального селектора відкидає правило для менш конкретного селектора, а правила, подані нижче у початковій таблиці стилів або документі, мають більш високий пріоритет.

Докладно розглянувши каскадні таблиці стилів слід також подивитися основні особливості останнього дуже потужного засобу front-end-розробки, а саме – мови програмування JavaScript. Це спеціалізована мова програмування, яка традиційно використовується для управління об’єктною моделлю документа у браузері під час перегляду сторінок мережі Інтернет (існують продукти, що перетворюють JavaScript на мову загального призначення, команди якої виконуються на сервері, наприклад, Node.JS; але такий підхід сильно протирічить початковій концепції використання цієї мови програмування, і, за умови наявності значної кількості традиційних засобів back-end-розробки перетворення на серверну мову ще й JavaScript значна кількість програмістів вважає абсолютно недоцільною, тому про цей варіант використання говорити більше не будемо).

Особливістю JavaScript є те, що його висхідні програмні коди інтерпретуються, що є досить гнучким, але повільним рішенням.

Оператори у цій, в цілому С-подібній мові, розділяються крапкою з комою. Мова чутлива до регістру, що часто випускають з виду початківці, ймовірно тому, що HTML, застосовуваний зазвичай з JavaScript спільно, не залежить від регістру (імена тегів і атрибутів HTML можна писати як малими, так і великими літерами).

Однорядковий коментар виділяється символом //, а багаторядковий - парою символів / * і * /, в чому JavaScript знову повторює С.

До даних застосовується слабкий (динамічний) контроль типів. В операторах з різнотипними даними останні автоматично приводяться до необхідного типу. Типи даних можуть бути примітивними і складеними. Примітивні типи містять прості однорідні значення, такі дані можна передавати функціям як параметри за значенням, а не за посиланням. Складені типи містять різнорідні дані (в тому числі і складені), їх передають у функції тільки за посиланням.

Мова JavaScript об’єктно-орієнтована, проте заснована на прототипах, а не на класах. Є чотири типи об’єктів: вбудовані об’єкти, об’єкти браузера, об’єкти документа і об’єкти користувача (програміста).

Введення-виведення в основному обмежене взаємодією з документами і користувачами. За умовчанням передбачається, що доступ до локальної файлової системи заборонений. Однак браузери можуть надавати спеціальні об’єкти, за допомогою яких забезпечується робота з файловою системою користувача, хоча і з видачею попереджень про небезпеку виконання файлових операцій.

Сценарії JavaScript активно взаємодіють з об’єктами, вбудованими в Web-сторінку. Для цього вони, власне, і створюються. Але перш, ніж ця взаємодія стане можливою, слід впровадити код сценарію в текст HTML-документа. Існує кілька способів зв'язати HTML-документ з конкретним сценарієм (скриптом), але зазвичай їх просто розміщують всередині контейнерного тега <SCRIPT>, тобто між дескрипторами <script> і </script>.

Контейнер <SCRIPT> в цьому випадку буде перебувати безпосередньо в HTML-документі, причому у довільному його місці. Програмний код пишуть прямо в HTML-документі або в спеціальних текстових файлах, які можна викликати з головного HTML-документа. Для початку розглянемо перший варіант. Перш за все браузер знаходить тег <script> в тілі веб-документа, і весь наступний текст намагається обробити як скриптовий код. І так до тих пір, поки не зустріне закриваючий тег </script>. Після цього всі наступні символи будуть вважатися HTML-текстом. Будь-який HTML-документ може містити довільне число «скриптових включень», але кожне має відкриватися і завершуватися відповідним тегом. Від їх розташування у тілі HTML-документа іноді може залежати функціонування всієї Web-сторінки, але про це буде сказано пізніше.

Контейнерний тег <script> може містити атрибут SRC, який вказує ім'я або URL-адресу текстового файлу, що містить код сценарію. Цей атрибут необхідний в тому випадку, якщо сценарій розташований не безпосередньо в HTML-документі, а в окремому файлі. Розширення файлу зі сценарієм може бути яким завгодно, але зазвичай використовують js:

```
<SCRIPT SRC = «myscripts.js»> </ SCRIPT>.
```

Якщо сценарій розташовується в окремому файлі, то в ньому, зрозуміло, теги <SCRIPT> і </ SCRIPT> не пишуть. Сценарій, завантажений з зовнішнього файлу, можна уявити собі просто як його вставку в HTML-документ.

У браузерах, що потенційно підтримують сценарії, цю функцію користувач може відключити (зокрема, із міркувань підвищеної безпеки). Крім того, існують браузери, які принципово не підтримують сценарії. У даній ситуації бажано хоча б вивести повідомлення про те, що на сторінці був сценарій, але в даному конкретному випадку він не виконується. З цією метою в HTML-документі використовують контейнерний тег <noscript>, в якому розміщують текст, який з'явиться користувачеві за умови, що сценарій з тієї, чи іншої причини не виконуватиметься. Всі браузери, які підтримують

сценарії, проігнорують вміст тегів <noscript> крім тих випадків, коли підтримка сценаріїв відключена. Браузери, що принципово не підтримують сценарії, навпаки, вміст тега <script> опустять (особливо, якщо він вкладений у теги <!-- -->, які є HTML-коментарем), а тега <noscript> - відобразять. Приклад наведено нижче:

```
<HTML> <HEAD>
<TITLE> Приклад застосування тега NOSCRIPT </ TITLE>
</ HEAD>
<SCRIPT TYPE = "text / JavaScript»>
<! -
alert ( «Підтримка JavaScript включена»); // ->
</ SCRIPT>
<NOSCRIPT>
<H2> Ваш браузер не підтримує JavaScript або його підтримка відключена </ H2>
</ NOSCRIPT>
</ HTML>
```

Тут був використана функція alert (повідомлення) для виведення повідомлень в маленькому діалоговому вікні.

Як уже зазначалося, в окремих файлах зазвичай розміщують бібліотеки функцій (визначення функцій), а також сценарії, які використовуються в декількох HTML-документах одного або декількох сайтів. Сценарій можна також писати у вигляді рядка операторів, між якими ставиться крапка з комою. Такий рядок, взятий в лапки, служить у якості значення атрибута-події, наприклад:

```
<IMG SRC = "mypicture.gif” ONCLICK = “alert ('Привіт!')”.
```

Виклики функцій і їх визначення можуть слідувати в довільному порядку, але тільки якщо вони розташовані в одному і тому ж контейнері <script>. Якщо вони розміщуються у різних контейнерах <script>, то необхідно, щоб визначення функції передувало її виклику. Аналогічно, якщо визначення функцій знаходяться в окремому файлі, то його необхідно завантажити в HTML-документ раніше викликів цих функцій.

При спробі завантажити і виконати неправильний варіант HTML-коду з'явиться діалогове вікно з повідомленням «Помилка: Передбачається наявність об’єкта». Браузер інтерпретує теги HTML послідовно. Так,

зустрівши вираз виклику функції myfunc() в одному контейнері <script>, браузер ще не має в пам'яті визначення цього об'єкта (функції), розташованого в іншому контейнері <script>. Якщо ж визначення функції і її виклик знаходяться в одному і тому ж контейнері <script>, то його вміст спочатку завантажується в пам'ять, а потім аналізується. Коли інтерпретатор зустрічає виклик функції, то він шукає її визначення в пам'яті і в разі успіху виконує код цієї функції.

Якщо необхідно, щоб сценарій виявився в браузері перш, ніж буде завантажено елементи HTML-документа, то його слід розташувати у верхній частині HTML-коду, а ще краще в контейнері <head> в заголовку документа.

### 2.1.2. Засоби Back-end розробки.

Як зазначалося вище, під Back-end'ом мають на увазі програмні компоненти, що працюють на сервері, і вибір засобів для реалізації цих компонентів є вкрай широким – рис. 2.1, справа.

Умовно усі засоби для серверного веб-програмування можна поділити на два класи: окремі програми, що працюють відповідно до технології CGI та мови, висхідні коди яких вбудовуються прямо у веб-сторінку та проганяються препроцесором веб-серверу перед видачею у браузер клієнта. Другий підхід є більш зручним, про що свідчить все більше занепадання CGI-засобів. У якості альтернативи для них виступають такі серйозні продукти, як:

- серверні сторінки Java - JSP, що активно просуваються любителями цієї відкритої, сучасної мови програмування (загального призначення);

- активні серверні сторінки ASP від корпорації Microsoft (відповідно ступінь підтримки цього рішення надзвичайно високий);

- використання мови PHP, яку і обирає більшість розробників серверних додатків через цілий комплекс позитивних рис цього продукту, які розглянемо докладніше.

На сьогоднішній день PHP є найбільш поширеною мовою веб-програмування. Переважна більшість сайтів і веб-сервісів в Інтернеті написано за допомогою PHP. За деякими оцінками PHP застосовується більш ніж на 80% сайтів, серед яких такі сервіси, як facebook.com, vk.com, baidu.com і інші. І така популярність не видається дивною. Простота мови дозволяє швидко і легко створювати сайти і портали різної складності.

PHP був створений в 1994 році датським програмістом Расмусом Лердорфом і спочатку являв собою набір скриптів на іншій мові програмування Perl. Пізніше цей набір скриптів був переписаний в інтерпретатор на мові C. Із самого виникнення PHP представляв зручний набір інструментів для спрощеного створення веб-сайтів і веб-додатків.

Розглянемо, які ж переваги надає PHP:

- для всіх найбільш поширених операційних систем (Windows, MacOS, Linux) є свої версії пакетів розробки на PHP, а це означає, що можна створювати веб-сайти на будь-якій з цих операційних систем;

- PHP може працювати в зв'язці з різними веб-серверами: Apache, Nginx, IIS, тощо;

- простота і легкість освоєння є особливістю цієї мови. Як правило, маючи навіть невеликий досвід в програмуванні на PHP, можна створювати простенькі веб-сайти;

- PHP схожий на мову C, тому, знаючи C, або одну з мов з C-подібним синтаксисом, буде простіше опановувати PHP;

- PHP підтримує роботу з великою кількістю систем баз даних (MySQL, MS SQL Server, Oracle, Postgres, MongoDB, та інші);

- поширеність хостингових послуг і їх дешевизна. Так як, як правило, хостингові компанії розміщують веб-сайти на PHP на веб-серверах Apache або Nginx, які працюють на одній з операційних систем сімейства Linux. І веб-сервери, і операційні системи на базі Linux безкоштовні, що знижує загальну вартість використання хостингу;

- постійний розвиток, адже PHP продовжує розвиватися, виходять все новіші версії, які несуть нові функції, адаптуючи мову програмування до нових викликів. І, як правило, перехід на нову версію не викликає ніяких труднощів;

- існує надзвичайно багато якісних Інтернет-ресурсів з підтримки процесу програмування на PHP, де можна задати власне питання і найскоріше отримати кваліфіковану відповідь.

Зважаючи на усі позитивні риси, приймаємо PHP у якості основного засобу Back-end розробки у даному проекті.

## 2.2. Проектування бази даних розроблюваного Інтернет-ресурсу.

Розробка бази даних є дуже важливим питанням при створенні будь-якого веб-додатку, що використовує серверні дані (тобто практично будь-якого веб-додатку в цілому). Оскільки у якості серверної мови програмування було обрано PHP, то традиційно з ним використовують продукт MySQL, що є системою управління базами даних із численною кількістю переваг, у порівнянні з конкурентами:

- порівняно легкий продукт, що займає малу кількість оперативної пам’яті сервера;

- порівняно функціональний продукт, що включає практично усі основні функції по обробці даних та створенню ефективних БД;

- це програмне забезпечення є вільним для використання (на відміну від деяких корпоративних продуктів, ліцензії на які можуть коштувати тисячі умовних одиниць, тобто сотні тисяч грн.);

- існує значна кількість уже розробленого програмного забезпечення для роботи з цією СУБД;

- існує чимало Інтернет-ресурсів, на яких здійснюється технічна підтримка програмістів, що здійснюють розробку для MySQL.

Зважаючи на усі ці особливості, СУБД MySQL береться за основу для збереження даних, що використовуватиме веб-додаток, що розробляється.

Проаналізуємо таблиці, що слід ввести у проектованій БД:

а) specialties, де зберігатиметься інформація про спеціальності, на які можна здійснювати вступ у даному році.

Ця таблиця міститиме поля:

- SpecialtyID – ідентифікатор спеціальності, який буде виступати первинним ключем цієї таблиці;
- SpecialtyName – назва спеціальності;
- SpecialtyZNODisciplines – дисципліни, за якими слід здавати ЗНО, для вступу на дану спеціальність;
- SpecialtyNumber – номер спеціальності;
- SpecialtyTextAbout – короткий текстовий опис спеціальності.

Типи кожного поля можна подивитися на рис. 2.2, де показана структура цієї таблиці.

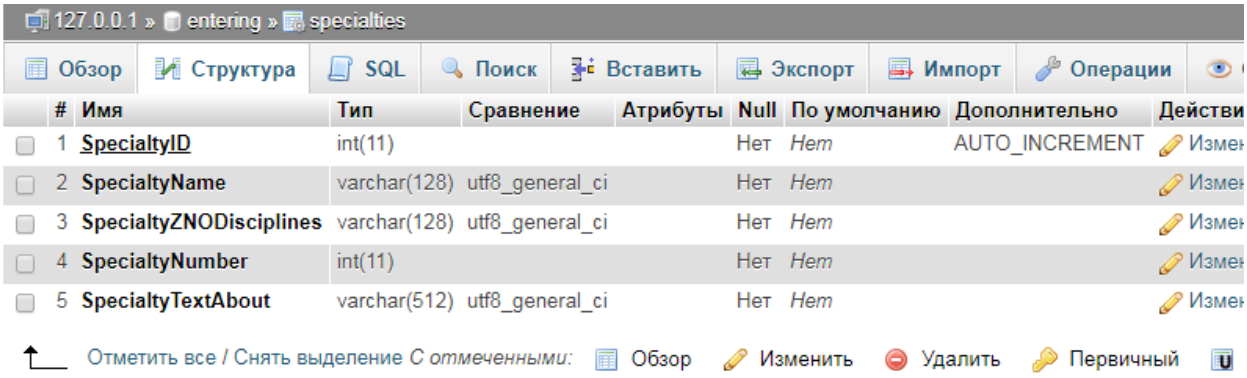



Рис. 2.2. Структура таблиці specialties.

б) таблиця universities, де буде зберігатися інформація про усі університети країни. Таблиця міститиме наступні поля:

- UniversityID – ідентифікатор університету; це число буде виступати первинним ключем цієї таблиці;
- UniversityName – назва університету;
- UniversityDescription – короткий текстовий опис університету;

- UniversityNumberOfStudents – кількість студентів, що навчаються у даному університеті;

- UniversityPicture – посилання на файл з зображенням у каталозі Images, що відповідає даному університетові.

Типи кожного поля можна подивитися на рис. 2.3, де показана структура цієї таблиці.

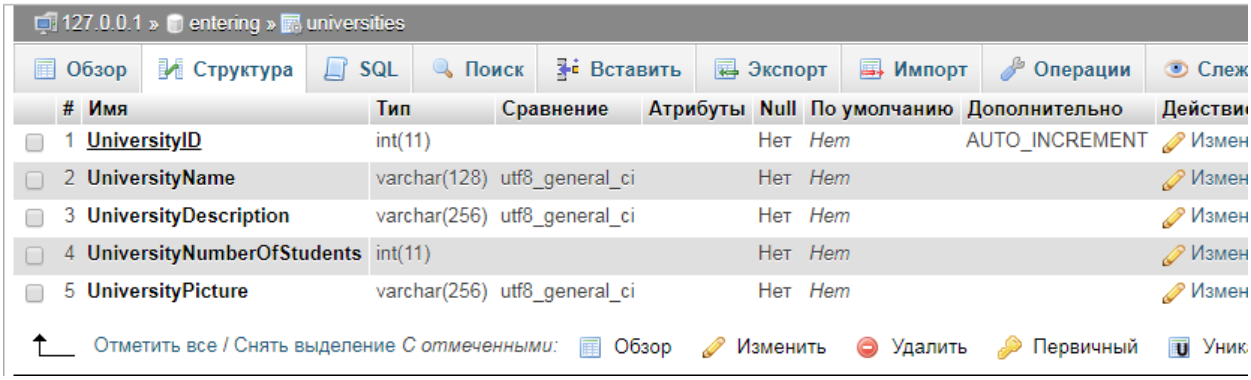



Рис. 2.3. Структура таблиці universities.

в) таблиця specialtiesinunuiversities, де буде зберігатися інформація про особливості спеціальності $a_i$ в університеті $b_i$. Ця окрема таблиця несе допоміжний характер і необхідна для організації зв'язку між спеціальностями та університетами типу «багато-до-багатьох». Вона міститиме наступні поля:

- SIUID – ідентифікатор запису, що відповідає спеціальності $a_i$ в університеті $b_i$, це число буде виступати первинним ключем цієї таблиці;

- SIUSpecialtyID – ідентифікатор спеціальності $a_i$ з таблиці specialties, зовнішній ключ;

- SIUUniversityID – ідентифікатор університету $b_i$ з таблиці universities, до якого планується вступ, зовнішній ключ;

- SIUPrice – вартість навчання за даною спеціальністю у даному ВНЗ.

Дана таблиця представлена на рис. 2.4.

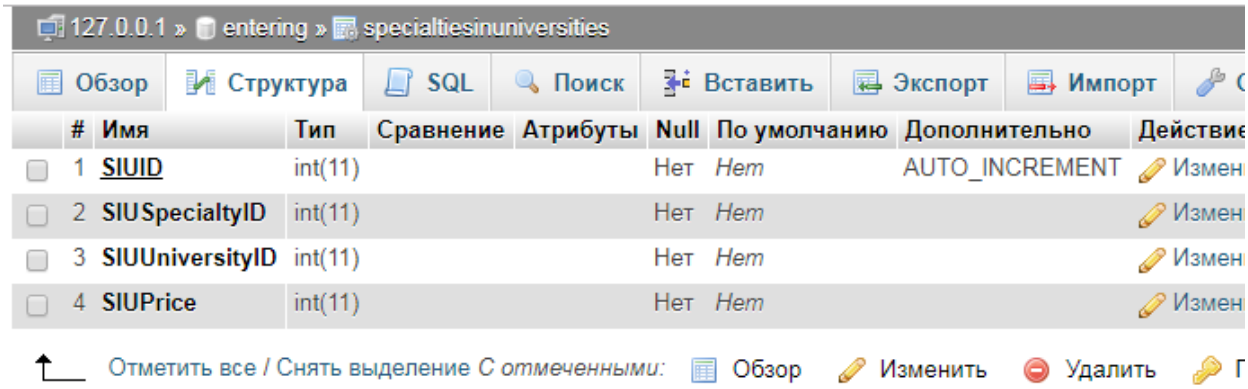



Рис. 2.4. Структура таблиці specialtiesinuniversities.

г) таблиця divisions, де буде зберігатися інформація про усі структурні підрозділи різних університетів, пов’язані із процесом навчанням (тобто такі, до яких можуть відноситися ті, або інші спеціальності та студенти). Структура таблиці буде наступною:

- DivisionID – ідентифікатор структурного підрозділу, це буде первинний ключ усієї таблиці;
- DivisionName – ім’я відокремленого структурного підрозділу (оскільки усі відокремлені структурні підрозділи усіх університетів мають унікальні назви, то тут реалізується зв’язок «один-до-багатьох»; навіть якщо деякі назви співпадають в даний момент, вони можуть почати різнитися у майбутньому, тому тут не можна організовувати зв’язок «багато-до-багатьох»);
- DivisionType – ідентифікатор типу підрозділу, зовнішній ключ, обирається із таблиці divisiontypes, що описана нижче;
- DivisionContacts – контакти підрозділу (особливо актуально, якщо факультет, чи інститут територіально розміщені в окремому приміщенні), тобто інформація, як його можна знайти (телефон, адреса, посилання на сайт/сторінку, і т.д.);
- DivisionUniversity – визначає університет, до якого належить даний підрозділ, зовнішній ключ.

Типи кожного поля можна подивитися на рис. 2.5, де показана структура цієї таблиці.

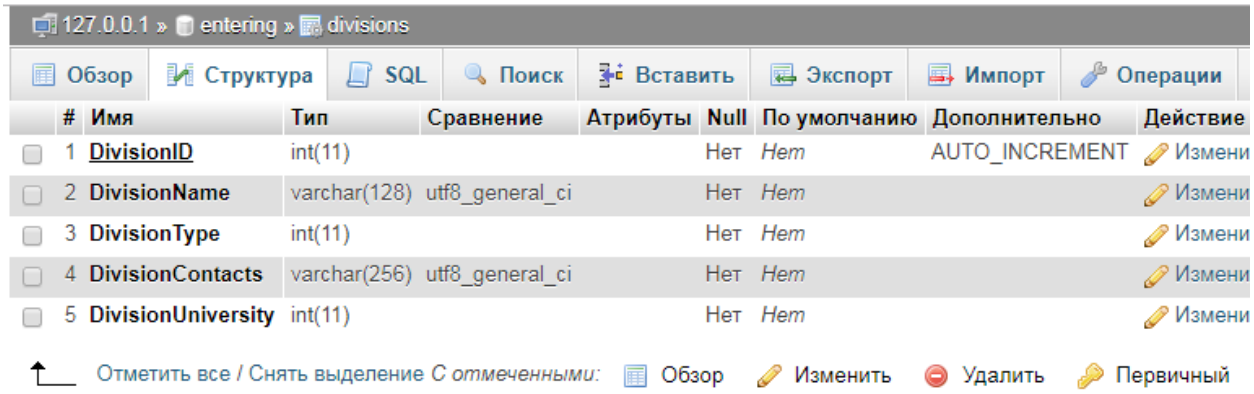



Рис. 2.5. Структура таблицы divisions.

д) таблиця divisiontypes, що має чисто допоміжне технічне значення, необхідна для того, щоби уникнути очевидного дублювання інформації. Містить поля:

- DivisionTypeID – ідентифікатор типу підрозділу, первинний ключ цієї таблиці;

- DivisionTypeName – назва типу підрозділу (наприклад, інститут, факультет, кафедра, тощо).

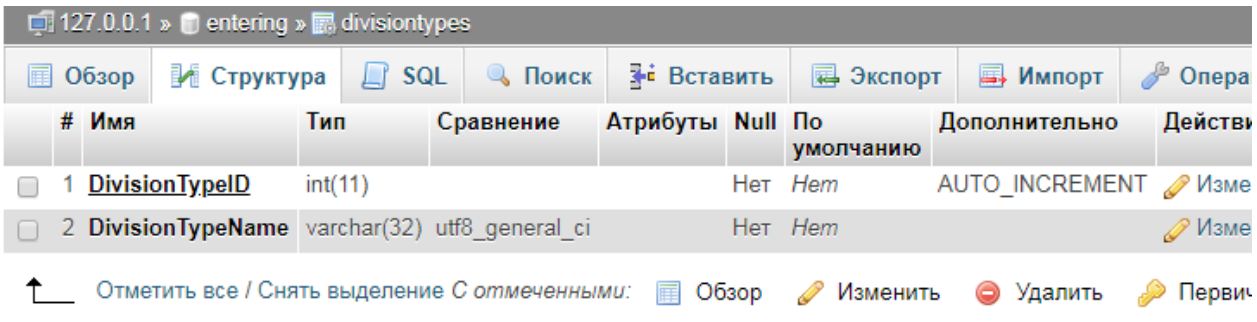



Рис. 2.6. Структура таблицы divisiontypes.

За наведеними відомостями будуємо таблиці, встановлюємо зв’язки між ними, та реалізовуємо БД entering – рис. 2.7.

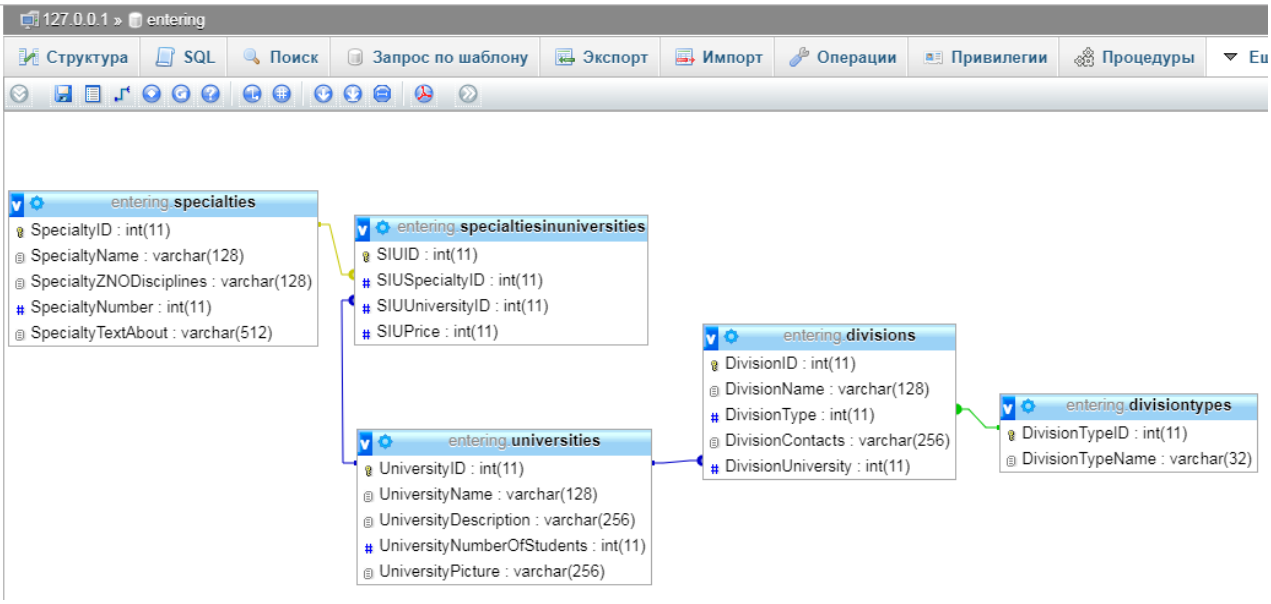



Рис. 2.7. Схема даних спроектованої БД entering.

## 2.3. Зовнішній вигляд сторінок сайту (інтерфейс користувача системи).

Виконавши реалізацію БД системи, можемо переходити до проектування інтерфейсу користувача. Це питання тісно пов’язане із комунікаційними технологіями, що застосовані у веб-додатку для здійснення зв’язку клієнтської та серверної частин.

Оскільки часті перезавантаження сторінки після кожної дії користувача не є ознакою гарного інтерфейсу, в роботі будемо орієнтуватися на технологію AJAX, яка дозволяє здійснювати активний обмін інформацією (як завгодно інтенсивний, включаючи звернення до баз даних, що і потрібно для цілей даної роботи), без перезавантаження всієї сторінки.

AJAX (англ. Asynchronous Javascript and XML) - спосіб побудови призначених для користувача web-додатків за допомогою фонового обміну інформацією браузера з сервером. Термін AJAX ввів Джессі Джеймс Гаррет у 2005 році. Першими додатками, що використовували дану технологію, стали сервіс карт Google Maps і поштовий клієнт Gmail.

Використання AJAX для розкрутки сайту дозволяє поліпшити його юзабіліті (додатки стають більш зручними і швидкими для відвідувачів), покращуються функціональність і зовнішній вигляд сторінок.

Коротко розглянемо принцип роботи цієї технології.

AJAX базується на технології звернення до сервера без перезавантаження сторінки (XTMLHttpRequest, створення дочірніх фреймів або тега script) або використанні DHTML, що дозволяє динамічно змінювати вміст. Формат передачі даних - XML або JSON. AJAX можна реалізувати в різних мовах програмування: PHP, Ruby on Rails, ASP.NET і інших. У коді web-сторінок широко використовується JavaScript для прозорого обміну даними клієнта з сервером. Користувачі взаємодіють зі стандартними HTML елементами, динамічна поведінка яких описується на JavaScript.

Для просування сайту застосування такої технології, як AJAX, має ряд суттєвих переваг:

- економія трафіку користувача (замість оновлення всієї сторінки, завантажується її невелика частина, що змінилася);

- зниження навантаження на сервер. Наприклад, на сторінці особистих повідомлень форуму при виділенні користувачем прочитаних листів сервер вносить зміни в БД і відправляє скрипту клієнта відповідь про виконання операції без повторного створення сторінки і її передачі;

- прискорення реагування інтерфейсу на команди користувача.

В той же час AJAX має і свої недоліки, серед яких можна назвати наступні:

- не завжди можлива інтеграція зі стандартним набором інструментів браузера, а саме, так як Інтернет-переглядачі не реєструють в історії AJAX-переходи по сторінках, не можна скористатися кнопкою «Назад». У деяких випадках немає можливості додати в закладки потрібний матеріал;

- контент, що завантажується динамічно, не доступний пошуковим системам, тому необхідно забезпечити альтернативний доступ до вмісту ресурсу;

- здійснюється неправильний облік статистики переміщення користувача по сайту;

- відбувається ускладнення програмного контролю цілісності типів і форматів, так як процеси форматування даних частково переносяться на сторону клієнта;

- в браузері користувача повинен бути включений JavaScript (не дуже суттєве на сьогоднішній день уточнення, оскільки з кожним роком частка користувачів із увімкненим JavaScript наближається до 100 %).

Альтернативою AJAX виступають Java-аплети, JavaFX, технології ActionScript 3, Flash Remoting, Adobe Flex, складові технологічну основу Rich Internet Applications від Macromedia, і Silverlight від корпорації Microsoft.

Таким чином, враховуючи усі особливості технології, будемо використовувати AJAX для організації роботи інформаційної системи з підбору вищого навчального закладу для абітурієнтів, в результаті чого увесь додаток матиме лише одну робочу сторінку – рис. 2.8.

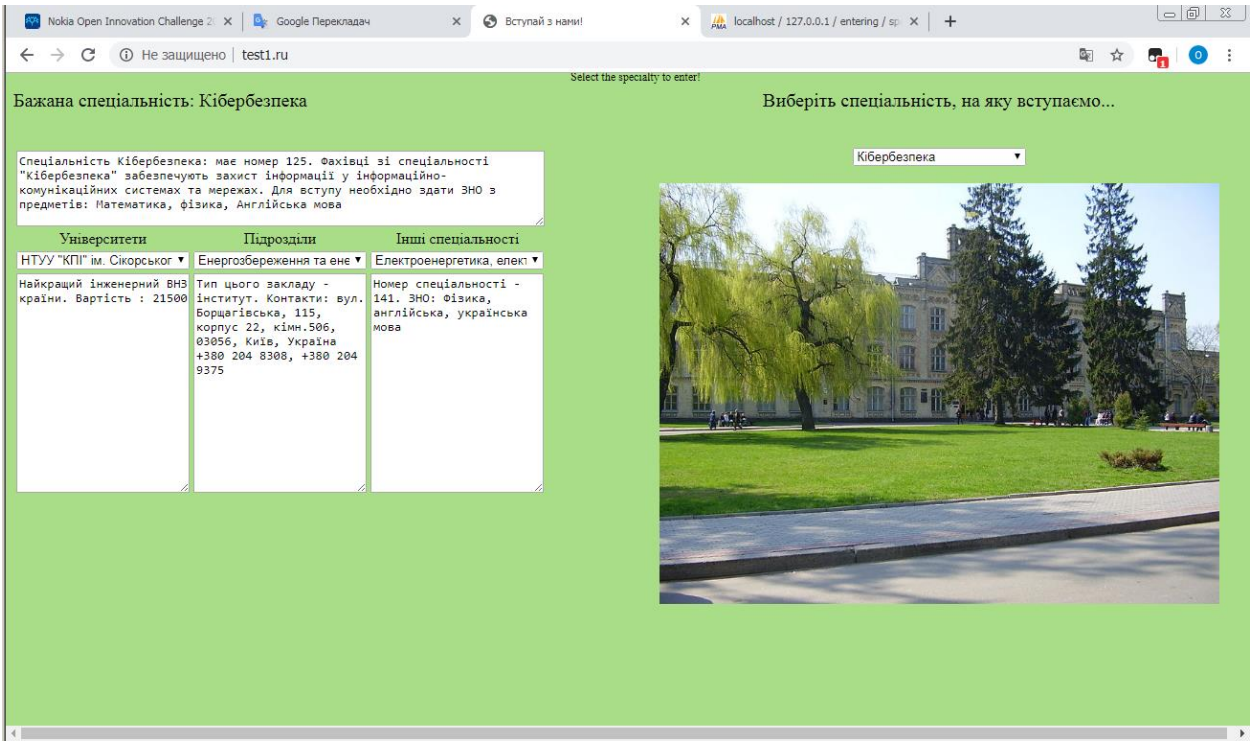



Рис. 2.8. Зовнішній вигляд розробленого веб-додатка.

Змінюючи різноманітні елементи управління, розміщені на цій сторінці, користувач буде ініціювати AJAX-запити, що швидко доставлятимуть назад усю необхідну інформацію без перезавантаження цілих сторінок. Тим самим досягається цілісність сприйняття системи, що реально видається для користувача одним додатком (на зразок Desktop-додатків для PC). Відсутність перезавантаження сторінок робить роботу в системі легкою та простою.

В той же час, використання технології AJAX вимагає залучення значної кількості сценаріїв, що обробляють запит (зазвичай POST), та надсилають назад певним чином сформовані дані. Для простих задач формат цих даних може визначатися програмістом, а при необхідності передачі великих обсягів даних слід використовувати формати представлення даних XML або JSON (останній став дозволеним для використання у стандарті HTML5). Підкреслимо, що для цілей даної роботи по мережі будуть передаватися порівняно малі обсяги інформації (орієнтовно кажучи, до десятків рядків SQL-запиту), тому доцільним є використання власного формату відповіді (розглянуто у наступному розділі).

# 3. ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ ВЕБ-ДОДАТКУ З ПІДТРИМКИ ПРОЦЕСУ ВИБОРУ ВНЗ

## 3.1. Структура програмної реалізації розроблюваного ресурсу.

Зважаючи на незвичайне рішення про повне використання AJAX (та, як наслідок, існування тільки однієї сторінки, що бачить користувач), може представляти інтерес питання організації всієї системи, з точки зору окремих файлів, яке наведемо на рис. 3.1.

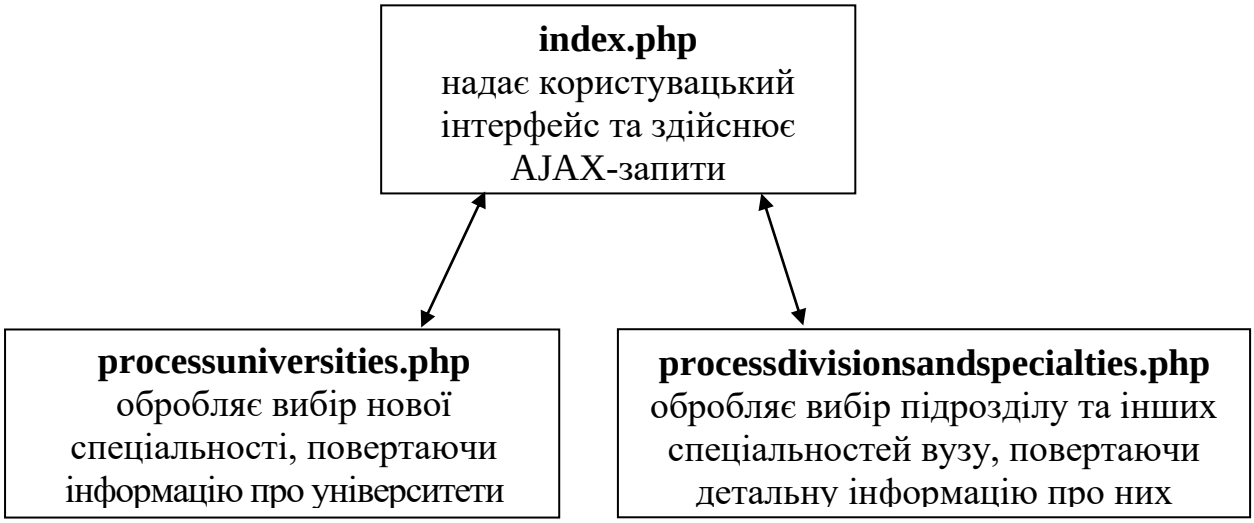



Рис. 3.1. Файлова структура проекту інформаційної системи з підбору вищого навчального закладу для абітурієнтів.

Ці два допоміжних файли processuniversities.php та processdivisionsandspecialties.php мають схожу структуру та працюють за однаковою схемою:

- підключення до бази даних;
- формування та виконання SQL-запиту (на базі інформації із POST-запиту, здійсненого інтерфейсною частиною index.php);
- формування та друк у відповідь текстового повідомлення, що включає в себе усі необхідні результати пошуку в базі даних, причому яке записано за певними правилами.

Інформація у текстовому повідомленні розділяється певними маркерами. Наприклад, якщо говорити про розбиття на найбільші блоки інформації, то для цього використано символ «*»: блок інформації про підрозділи університету відділяється від блоку інформації про інші спеціальності саме зірочкою (це повернення файлу processdivisionsandspecialties.php). Так само зіркою і блок загальної інформації про обрану спеціальність відділяється від блоку інформації про університети, в яких ця спеціальність доступна (це повернення файлу processuniversities.php).

Всередині великих блоків інформація, що відноситься до одного рядка запиту SQL, відділяється від іншого символом нового рядку “\n”. Всередині одного рядка окремі значення полів відділяються одне від одного крапкою з комою (аналогічно до формату CSV).

Дуже позитивним моментом є те, що обидві застосовані з різних боків мови програмування (PHP та JavaScript) мають широкі, зручні можливості по обробці рядків, тому поставлені задачі були легко вирішені в процесі написання дипломної роботи.

З точки зору структури проекту інтерес також представляє набір функцій (користувача), що застосовані в роботі, тому розглянемо їх перелік та призначення кожної:

- filluniversities(e) – функція, що обробляє першу відповідь на AJAX-запит, пов’язаний із отриманням інформації про спеціальність та переліком університетів, у яких дана спеціальність представлена;

- ChangedUniversitySelect() – обробник процесу змінення випадаючого списку з університетами, надсилає AJAX-запит для отримання інформації про підрозділи університету та інші спеціальності, що присутні для даного обраного університету;

- filldivisionsandotherspecialties(e) – функція-обробник AJAX-відповіді, яка розміщує усю отриману від PHP-скрипта інформацію про підрозділи ВНЗ

та про наявні у ньому інші спеціальності по необхідним змінним та елементам управління;

- ChangedDivisionSelect() – обробник змінення випадаючого списку підрозділів, що є доступними для даного обраного університету. Ця функція завантажує інформацію про факультети та інститути ВНЗ;

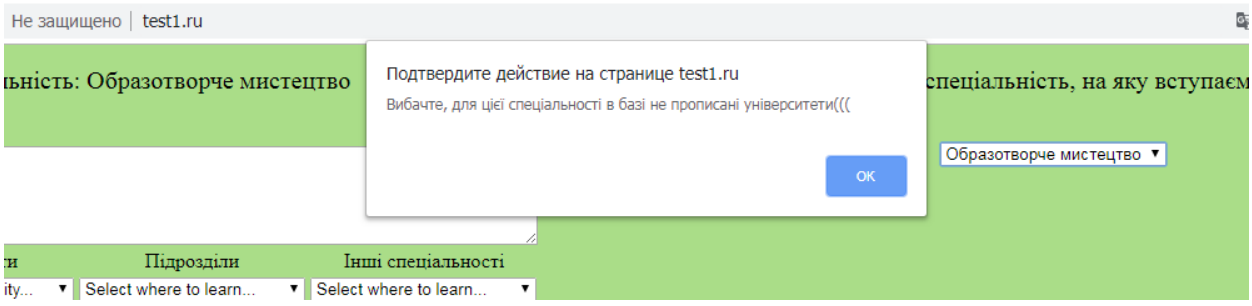
- ChangedOtherSpecialtySelect() – аналогічна функція, але для вибору та обробки даних про інші спеціальності вузу;

- ChangedSpecialtySelect() – функція обробник вибору основної спеціальності, на яку планується вступ. Ініціює AJAX-запит про університети, де ця спеціальність присутня.

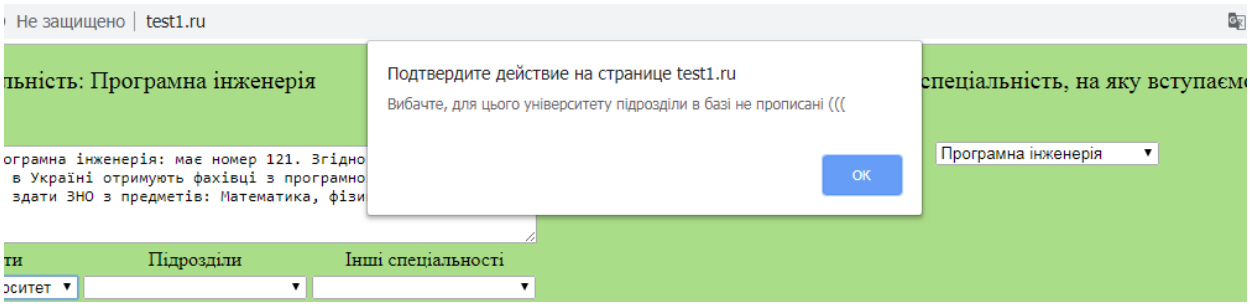
## 3.2. Особливості програмної реалізації алгоритмів та результати роботи проектованого сайту.

Безпосередньо запропоновані алгоритми уже описані у попередніх розділах даної роботи.

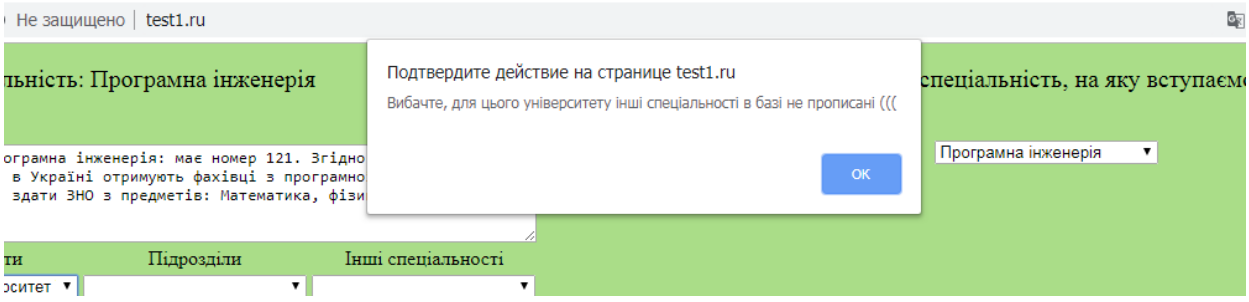
Головне вікно програми уже було наведено на рис. 2.8. Якщо у базі даних відсутня якась інформація, необхідна для заповнення списку університетів, або їхніх підрозділів, або інших спеціальностей (крім обраної), то користувачеві видається про це одне із попереджувальних повідомлень – рис. 3.2.




*а*)




*б*)




*в*)

Рис. 3.2. Попереджувальні повідомлення у випадку, якщо для обраних користувачем пунктів більш високого рівня відсутнє наповнення у базі даних: *а* – немає університетів для обраної спеціальності; *б* – немає підрозділів для обраного університету; *в* – немає інших спеціальностей для обраного університету.

В цілому, програмне забезпечення підтримки планування подорожей працює у штатному режимі, якщо користуватися наступною методикою роботи з ним:

- завантажити веб-додаток, який локалізований у файлі index.php;

- обрати у правій частині вікна браузеру бажану спеціальність, після чого у лівій верхній частині браузера має відобразитися назва обраної спеціальності та інформація про неї;

- у самому лівому списку, що випадає, після виконання зупинки, мають бути наявними усі університети, в які можна вступати за обраною спеціальністю, і серед яких слід клацнути мишею один із пунктів. Після цього у другий та третій випадаючі списки завантажаться переліки усіх підрозділів (факультетів, інститутів, і т.п.) обраного університету та усіх доступних у ньому інших спеціальностей (разом із їх коротким описом); якщо до університету прикріплено картинку, то вона завантажується у нижню частину правої сторони браузера;

- у другому списку можна продивитися усі підрозділи, наявні в обраному університеті: якщо клацнути один із них, то випадає докладна інформація про його тип, контактні відомості, тощо;

- у третьому списку можна продивитися інші спеціальності, наявні в обраному університеті: якщо клацнути одну із них, то випадає докладна інформація про цю спеціальність: шифр, предмети ЗНО, які треба здавати для вступу, тощо.

Після тривалої роботи з інформаційною системою з підбору вищого навчального закладу для абітурієнтів, встановлено, що система працює у штатному режимі без виникнення аварійних ситуацій, що в першу чергу обумовлене чіткою взаємодією серверних частин (PHP-файлів) та клієнтської частини index.php, а також системи управління базами даних.

# 4. ОХОРОНА ПРАЦІ

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних та лікувально-профілактичних заходів та засобів, що направлені на збереження життя та здоров’я (працездатності) людини в процесі трудової діяльності.

Оскільки диплом присвячено розробці «Інформаційної системи з підбору вищого навчального закладу для абітурієнтів» тому для розгляду питань охорони праці обрано робоче місце інженера-програміста, яке розташоване в кабінеті офісної будівлі.

## 4.1 Організація та управління охороною праці

Працівник інженер-програміст, на своєму робочому місці, виконує свою діяльність на фірмі, отже, розглянемо організацію та управління охороною праці взагалі, та у підрозділі 4.2 більш детально.

Згідно з п. 4.3.1. «Рекомендацій щодо побудови, впровадження та удосконалення системи управління охороною праці», фірма розробила Положення про службу охорони праці, що відповідає Типовому положенню про службу охорони праці (НПАОП 0.00-4.35-04), затвердженому наказом Держнаглядохоронпраці України від 15.11.2004 N 255 (z1526-04), зареєстрованому в Мін'юсті України 01.12.2004 за N 1526/10125.

У Положенні про СУОП регламентується порядок дій, компетенція відповідальних осіб при організації і проведенні навчання, своєчасна актуалізація навчальних програм та інструкцій. Виконуються обов'язкові вимоги до проведення навчання з питань охорони праці, що викладено в статті 18 Закону України "Про охорону праці", а також у Типовому положенні про порядок проведення навчання і перевірки знань з питань охорони праці (НПАОП 0.00-4.1205), затвердженому наказом Держнаглядохоронпраці України від 26.01.2005 N 15, зареєстрованому в

Мін'юсті України 15.02.2005 за N 231/10511. Порядок проведення і види інструктажів також викладено в зазначеному Типовому положенні.

Відповідно до Переліку робіт, де є потреба у професійному доборі (ДНАОП 0.03-8.06-94), та Переліку важких робіт зі шкідливими і небезпечними умовами праці, на яких забороняється застосування праці жінок (ДНАОП 0.03-8.08-93), на фірмі не проводиться професійний добір та немає заборони на працю жінок.

Питання щодо забезпечення працівників ЗІЗ регламентується Положенням про порядок забезпечення працівників спеціальним одягом, спеціальним взуттям та іншими засобами індивідуального захисту (НПАОП 0.00-4.26-96). Інструктаж працівників щодо використання ЗІЗ повинен бути викладений в інструкціях з охорони праці згідно з Положенням про розробку інструкцій з охорони праці (НПАОП 0.00-4.15-98).

Питання, що стосуються нещасних випадків та аварій, регламентуються Порядком розслідування та ведення обліку нещасних випадків, професійних захворювань і аварій на виробництві (НПАОП 0.00-6.02-04).

## 4.2. Аналіз умов на робочому місці інженера-програміста.

### 4.2.1 Аналіз освітлення.

Згідно постанови ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» найменша нормована освітленність складає 30 лк, найбільша – 5000 лк. Зорова робота інженера-програміста відноситься до робіт малої важкості, тому що виконується сидячи і не потребують фізичного напруження, отже наведемо нормовані значення рівня освітлення в табл. 4.1.

Таблиця 4.1 – Нормовані значення рівня освітлення.

| Характеристика зорової роботи | Найменший або еквівалентний розмір об'єкта розрізнення, мм | Розряд зорової роботи | Підрозряд зорової роботи | Контраст об'єкта з фоном | Характеристика фону | Штучне освітлення | | | | | Природне освітлення | | Суміщене освітлення | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Освітленість, лк | | | сукупність нормованих величин показника осліпленості і коефіцієнта пульсації | | КПО, $e_н$, % | | | |
| | | | | | | при системі комбінованого освітлення | | при системі загального освітлення | | | при верхньому або комбінованому освітленні | при боковому освітленні | при верхньому або комбінованому освітленні | при боковому освітленні |
| | | | | | | всього | ут. ч. від загального | | P | Кп, % | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Малої точності | Більше 1,0 до 5 | V | а | Малий | Темний | 400 | 200 | 300 | 40 | 20 | 3 | 1 | 1,8 | 0,6 |
| | | | б | Малий Середній | Середній Темний | — | — | 200 | 40 | 20 | | | | |
| | | | в | Малий Середній Великий | Світлий Середній Темний | — | — | 200 | 40 | 20 | | | | |
| | | | г | Середній Великий Великий | Світлий Світлий Середній | — | — | 200 | 40 | 20 | | | | |

На робочому місці присутнє природне і штучне освітлення. Вікно в приміщенні направлене на схід і має занавіски. Освітленість робочого місця становить 350 лк.

4.2.2. Повітря робочої зони.

У повітрі на робочому місці інженера-програміста відсутні джерела шкідливих та небезпечних речовин, але побутовий пил, вуглекислий газ та мікроорганізми присутні, отже наведемо у табл. 4.2 ГДК для наведених речовин у повітрі робочої зони згідно з наказом «Про затвердження списків і введення в дію гігієнічних регламентів шкідливих речовин у повітрі робочої зони і атмосферному повітрі населених місць».

Табл. 4.2 - ГДК для шкідливих речовин робочої зони.

| Назва речовини | ГДК |
|---|---|
| Пил(тваринний/рослинний, 10%,20%) | 2, 4, 6 |
| Вуглекислий газ (CO2) | 0,5% об. або 9000 мг/м3 |
| Мікроорганізми-продуценти Streptomyces avermitilis | 5.10ст.2 куо/куб.м |

4.2.3. Мікрокліматичні умови праці.

Важкість праці Легка (1а) встановлює перелік його функціональних обов'язків, отже, згідно з ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» були визначені та наведені у табл. 4.3 мікрокліматичні параметри умов праці.

Табл. 4.3 – Мікрокліматичні параметри умов праці.

| Період року | Категорія робіт | Температура повітря | Відноста вологість | Швидкість руху, м/сек. |
|---|---|---|---|---|
| Холодний період року | Легка 1а | 22-24 | 60-40 | 0.1 |
| Теплий період року | Легка 1а | 23-25 | 60-40 | 0.1 |

Основні параметри мікроклімату – температура повітря, вологість, швидкість руху вітру та інтенсивність опромінювання. Далі перелічені отримані параметри:

- категорія робіт в приміщенні – легка 1а;
- швидкість руху повітря дорівнює 0.1 м\с;
- температура повітря дорівнює 23 град. С;
- відносна вологість дорівнює 53%.

4.2.4. Іонізуюче випромінювання.

Згідно з НПАОП 00.3-3.24-97 "Норми радіаційної безпеки України (НРБУ-97)" на робочому місці виконуються заходи по зменшенню можливого впливу іонізуючого випромінювання.

4.2.5. Електробезпека.

Згідно з ДНАОП 0.00-1.21-98 «Про затвердження Правил безпечної експлуатації електроустановок споживачів» електричні прилади в приміщенні заземлені та знаходяться в справному стані.

4.2.6. Пожежна безпека.

Категорія пожежної небезпеки будівлі - Д. Ступінь пожежостійкості – II ступінь. В приміщенні відкрите складування і відсутні легкозаймисті і горючі речовини.

4.3. Індивідуальне завдання. Розрахунок системи повітрообміну на робочому місці.

В табл. 4.4 вказані задані значення параметрів для розрахунків.

Табл. 4.4 – задані значення параметрів.

| Назва | Значення параметру |
|---|---|
| Технологічний процес | Травлення соляною кислотою |
| Шкідливі виділення при заданому техпроцесі | Хлороводень (соляна кислота) HCl<br>Теплота |
| Площа відкритої частини отвору F, $м^2$ | 0,25 |
| Швидкість руху повітря в робочій зоні приміщення $V_в$ , м/с | 0,5 |
| Температура повітря в робочій зоні приміщення $t_в$ , $^oC$ | 20 |
| Температура речовин і матеріалів t, $^oC$ | 30 |
| Висота робочого отвору h, м | 0,435 |
| | |
| Пружність водяної пари в повітрі, Р1 кПа | 1,9 |
| Пружність водяної пари, що насичує повітря при температурі поверхні рідини, що випаровується, Р2, кПа | 20 |
| Поверхня випаровування, F, $м^2$ | 0,7 |
| Швидкість руху повітря над джерелом випаровування, V, м/с | 0,7 |
| Фактор гравітаційної подвижності навколишнього середовища | 0,035 |
| Зміст вологи видаляємого повітря,$d_{уд}$, $гр/м^3$ | 19 |
| Зміст вологи приточного повітря, $d_{пр}$, $гр/м^3$ | 15 |
| | |
| Висота вікна завантаження печі, h0, м | 0,5 |
| Ширина вікна завантаження печі, b0, м | 0,9 |
| Густина повітря в приміщенні, $p_г$, $кг/м^3$ | 1,13 |
| Густина газу в печі, $p_в$, $кг/м^3$ | 1,225 |

Розрахунок об’єму повітря для асиміляції тепловиділень, виконується по формулі 4.1:

$$L1 = 1840 * \sqrt[3]{h * q * F^2}, \quad (4.1)$$

де $h$ – висота робочого отвору, м;

$V_в$ – швидкість руху повітря в робочій зоні приміщення, м/с;

$t$ – температура речовин і матеріалів, $^oC$;

$t_в$ – температура повітря в робочій зоні приміщення, $^oC$;

$q$ – тепловиділення при процесі,

$$q = (5{,}71 + 4{,}06 * V_в) * (t_в - t) = 77{,}4 \text{ Вт.}$$

$$L1 = 1840 * \sqrt[3]{0{,}435 * 77{,}4 * 0{,}25^2} = 2357{,}88 \frac{м^3}{год}$$

Розрахунок продуктивності прямокутних зонтів для видалення шкідливих речовин, виконується по формулі (4.2):

$$L_{тр} = 3600 * (L_r + L_в), \frac{м^3}{год}, \quad (4.2)$$

де $L_r$ – кількість газів, що вириваються з печі під зонт, $м^3/с$;

$L_в$ – кількість підсмоктуємого повітря під зонт, $м^3/с$.

$$L_{тр} = 3600 * (0{,}5664 + 0{,}6687) = 4446{,}36 \frac{м^3}{год},$$

Розрахунок кількості газів, що вириваються з печі під зонт, виконується по формулі (4.3):

$$L_r = 0{,}62 * b0 * h0 * \sqrt{\frac{19{,}6 * \Delta Р_{из}}{p_r}}, \frac{м^3}{с}, \quad (4.3)$$

де $b0$ - ширина вікна завантаження печі;

$h0$ - висота вікна завантаження печі;

$\Delta Р_{из}$ – надлишковий тиск газу на половині висоти вікна,

$= \frac{h0 * (p_r - p_в)}{2} = 0{,}238$;

$p_r$ - густина повітря в приміщенні.

$$L_r = 0{,}62 * 0{,}9 * 0{,}5 * \sqrt{\frac{19{,}6 * 0{,}238}{1{,}13}} = 0{,}5664 \frac{м^3}{с},$$

Розрахунок максимально допустимої концентрації шкідливої речовини в гирлі труби виконується за формулою (4.4):

$$L_в = b0 * (0{,}62 * h0 + 0{,}056) * \sqrt{\frac{19{,}6 * \Delta Риз}{p_r}}, \frac{м^3}{с}, \quad (4.4)$$

де $b0$ - ширина вікна завантаження печі;

$h0$ - висота вікна завантаження печі;

$\Delta Риз$ – надлишковий тиск газу на половині висоти вікна,

$= \frac{h0 * (p_r - p_в)}{2} = 0{,}238$;

$p_r$ - густина повітря в приміщенні.

$$L_в = 0{,}9 * (0{,}62 * 0{,}5 + 0{,}056) * \sqrt{\frac{19{,}6 * 0{,}238}{1{,}13}} = 0{,}6687 \frac{м^3}{с}.$$

Розрахунок кількості повітря, що необхідно для видалення надлишків вологи виконується за формулою (4.5):

$$L_{тр} = 3600 * \frac{Gв}{dуд - dпр}, \frac{м^3}{год}. \quad (4.5)$$

де $G_в$ – кількість вологи, що виділяється усіма джерелами, мг/м$^2$,

$d$уд – зміст вологи видаляємого повітря, гр/м$^3$;

$d$пр – зміст вологи приточного повітря, гр/м$^3$;

$$L_{тр} = 3600 * \frac{0{,}1652}{19 - 15} = 148{,}68 \frac{м^3}{год}.$$

Кількість вологи, що виділяється усіма джерелами розраховується по формулі (4.6):

$$G_в = 0{,}278 * F * (0{,}017 * V + a) * (P2 - P1), \quad (4.6)$$

де $F$ - поверхня випаровування, м$^2$;

$V$ - швидкість руху повітря над джерелом випаровування, м/с;

$a$ – Фактор гравітаційної подвижності навколишнього середовища;

$P1$ - пружність водяної пари в повітрі, кПа;

$P2$ - пружність водяної пари, що насичує повітря при температурі поверхні рідини, що випаровується, кПа.

$$G_в = 0{,}278 * 0{,}7 * (0{,}017 * 0{,}7 + ,035) * (20 - 1{,}9) = 0{,}1652$$

Вибираємо з рис. 4.1 потрібний вентилятор з параметрами, що підходять до заданого приміщення. Для вибору використовуємо параметри, що було отримано розрахунками:

- необхідна продуктивність *Q* (найбільша з отриманих розрахунками) = 4446,36 м$^3$/год.

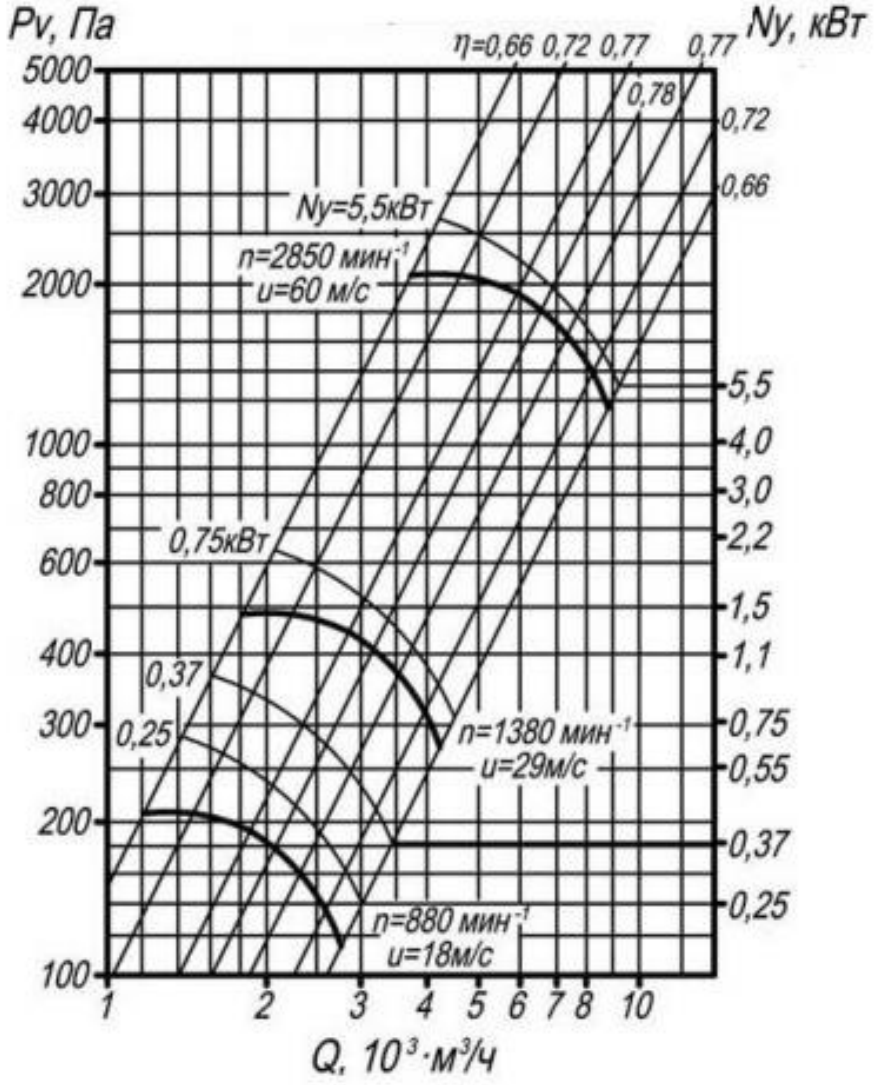



Рис. 4.1. Вибір параметрів вентилятора.

Вибраний вентилятор має наступні параметри:

- модель ВР 80-75 №4;
- потужність 0.37 кВт;
- швидкість обертів 18 м/с, 880 об/хв.;
- повний тиск 180 Па.

4.4. Заходи з охорони праці.

Оскільки індивідуальний розрахунок був спрямований контроль та вибір вентиляційної на робочому місці, то заходи з охорони праці будуть націлені на вибір вентиляційної установки з наступними параметрами (модель ВР 80-75 №4, потужність 0.37 кВт, швидкість обертів 18 м/с, 880 об/хв.) і на її обслуговування, а саме:

- підтримка системи в чистоті;
- підтримка механічних і електричних частин в робочому стані;
- дотримання правил безпеки з електричними частинами вентиляційної системи;
- генеральний огляд та очищення установки раз на рік.

4.5 Висновки по охороні праці.

У розділі розглянуто організацію та управління охороною праці (на підприємстві). Проаналізовано умови праці на робочому місці інженера-програміста. Запропоновано заходи з охорони праці, а саме вибір вентиляційної установки на робочому місці та догляд за нею.

# ВИСНОВКИ

1. Flanagan D. JavaScript: The Definitive Guide. 7th ed. Sebastopol: O'Reilly Media, 2020. 706 p.

2. Duckett J. JavaScript and JQuery: Interactive Front-End Web Development. Indianapolis: Wiley, 2014. 640 p.

3. Zakas N. Professional JavaScript for Web Developers. 4th ed. Indianapolis: Wrox, 2021. 960 p.

4. Freeman E., Robson E., Bates B., Sierra K. Head First HTML and CSS. 2nd ed. Sebastopol: O'Reilly Media, 2012. 768 p.

5. Meloni J. Sams Teach Yourself PHP, MySQL & JavaScript. 7th ed. Indianapolis: Pearson Education, 2017. 720 p.

6. Nixon R. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5. 6th ed. Sebastopol: O'Reilly Media, 2018. 829 p.

7. Powers D. PHP Solutions: Dynamic Web Design Made Easy. 5th ed. New York: Apress, 2023. 592 p.

8. Teixeira M. Pro PHP Programming. New York: Apress, 2014. 508 p.

9. Glassford J. PHP 8 Objects, Patterns, and Practice. 6th ed. New York: Apress, 2021. 624 p.

10. Resig J., Bibeault B., Kats E. Secrets of the JavaScript Ninja. 2nd ed. Shelter Island: Manning Publications, 2016. 464 p.

11. Crockford D. JavaScript: The Good Parts. Sebastopol: O'Reilly Media, 2008. 176 p.

12. McFarland D. CSS: The Missing Manual. 5th ed. Sebastopol: O'Reilly Media, 2017. 688 p.

13. Keith J., Andrew R. HTML5 for Web Designers. 2nd ed. New York: A Book Apart, 2016. 176 p.

14. Osmani A. Learning JavaScript Design Patterns. 2nd ed. Sebastopol: O'Reilly Media, 2017. 254 p.

15. Chaffer J., Swedberg K. Learning jQuery. 4th ed. Birmingham: Packt Publishing, 2013. 444 p.

16. Fowler S. Web Development with Node and Express: Leveraging the JavaScript Stack. Sebastopol: O'Reilly Media, 2019. 480 p.

17. Beighley L. Head First PHP & MySQL. Sebastopol: O'Reilly Media, 2008. 812 p.

18. Bovet D., Ciscon S. Understanding Linux Web Hosting. Boston: Addison-Wesley, 2016. 320 p.

19. Іванчук С., Петренко В. Основи Web-програмування: навчальний посібник. Львів: Видавництво Львівської політехніки, 2020. 210 с.

20. Ковальчук А. Сучасні технології Web-розробки: навчальний посібник. Київ: Кондор, 2019. 198 с.

# ДОДАТОК А. ВИСХІДНИЙ ТЕКСТ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ: ФАЙЛ INDEX.PHP.

```
<?php header('Content-Type: text/html; charset=UTF-8');
?>
<html>
<head>
<title>
Вступай з нами!
</title>
</head>
<body          bgcolor=#AADD88          style="background-image:
url(images/background.png)">
<script language="JavaScript">
var SpecialtySelected="";
var universityproperties=[];
var divisionproperties=[];
var specialtiesinuniversitiesproperties=[];
var universityimages=[];

<?php
$hostname="localhost"; $username="root"; $password="";
$dbName="entering";
$link=mysql_connect($hostname,$username,$password)   or   die("No
connect to DB!");
mysql_set_charset('utf8',$link);
mysql_select_db("$dbName",$link) or die("Cannot select DB!");
$query="select SpecialtyName from specialties;";
$result=mysql_query($query);
$num_rows=mysql_num_rows($result);
print "var text=new Array(";
for($i=0;$i<$num_rows;$i++)
{
     $row=mysql_fetch_array($result);
     print     "\"{$row['SpecialtyName']}\",";
}
print ");";
?>

function filluniversities(e)
{
     var blocks=e.target.responseText.split('*');
     document.getElementById("specialtyinfo").innerHTML=blocks[0]
;
     var strings=blocks[1].split('\n');
     var words;
     document.getElementById("universityinfo").innerText="";
     document.getElementById("divisioninfo").innerText="";
     document.getElementById("otherspecialtyinfo").innerText="";
     list=document.getElementById("divisionselect");
     while(list.length>0) list.remove(0);
     list=document.getElementById("otherspecialtyselect");
     while(list.length>0) list.remove(0);
```

```
     list=document.getElementById("universityselect");
     while(list.length>0) list.remove(0);
     universityimage.style.visibility="hidden";
     universityproperties=[];
     universityimages=[];
     if(strings.length>1)
          for(var i=0;i<strings.length-1;i++)
          {
               words=strings[i].split(';');
               var opt = document.createElement("option");
               opt.text = words[0];
               opt.value = i;
               list.appendChild(opt);
               universityproperties[i]=words[1]+".\n    Приблизна
кількість студентів : "+words[2];
               universityimages[i]=words[3];
          }
     else
          alert("Вибачте,  для  цієї  спеціальності  в  базі  не
прописані університети(((");
}

function ChangedUniversitySelect()
{
     var
index=document.getElementById("universityselect").selectedIndex;
     document.getElementById("universityinfo").innerText=universi
typroperties[index];
     var
universityimage=document.getElementById("universityimage");
     var imagestr;
     if(universityimages[index])
     {

     imagestr="images/"+universityimages[index];//alert(imagestr)
;
          universityimage.style.visibility="visible";
          universityimage.src=imagestr;
     }
     else
          universityimage.style.visibility="hidden";
     var data=new FormData();
     data.append('universityname',document.getElementById("univer
sityselect").options[document.getElementById("universityselect")
.selectedIndex].text);
     var req=new XMLHttpRequest();
     req.addEventListener('load',filldivisionsandotherspecialties
,false);
     req.open("POST","processdivisionsandspecialties.php",true);
     req.send(data);
}

function filldivisionsandotherspecialties(e)
```

```
{
     var blocks=e.target.responseText.split('*');
     var strings=blocks[0].split('\n');
     var words;
     var
divtypes=["","інститут","факультет","кафедра","навчально-
науковий центр"]
     document.getElementById("divisioninfo").innerText="";
     list=document.getElementById("divisionselect");
     while(list.length>0) list.remove(0);
     divisionproperties=[];
     if(strings.length>1)
     {
          for(var i=0;i<strings.length-1;i++)
          {
               words=strings[i].split(';');
               var opt = document.createElement("option");
               opt.text = words[0];
               opt.value = i;
               list.appendChild(opt);

     document.getElementById("divisioninfo").innerText="";
               divisionproperties[i]="Тип   цього   закладу   -
"+divtypes[words[1]]+".\n Контакти: "+words[2];
          }
          ChangedDivisionSelect();
     }
     else
          alert("Вибачте, для цього університету підрозділи в
базі не прописані (((");

     strings=blocks[1].split('\n');
     document.getElementById("otherspecialtyinfo").innerText="";
     list=document.getElementById("otherspecialtyselect");
     while(list.length>0) list.remove(0);
     otherspecialtyproperties=[];
     if(strings.length>1)
     {
          for(var i=0;i<strings.length-1;i++)
          {
               words=strings[i].split(';');
               var opt = document.createElement("option");
               opt.text = words[0];
               opt.value = i;
               list.appendChild(opt);

     document.getElementById("otherspecialtyinfo").innerText="";
               otherspecialtyproperties[i]="Номер спеціальності -
"+words[1]+".\n ЗНО: "+words[2];
          }
          ChangedOtherSpecialtySelect();
     }
     else
```

```
          alert("Вибачте,    для    цього    університету    інші
спеціальності в базі не прописані (((");
}

function ChangedDivisionSelect()
{
     var
index=document.getElementById("divisionselect").selectedIndex;
     document.getElementById("divisioninfo").innerText=divisionpr
operties[index];
}

function ChangedOtherSpecialtySelect()
{
     var
index=document.getElementById("otherspecialtyselect").selectedIn
dex;
     document.getElementById("otherspecialtyinfo").innerText=othe
rspecialtyproperties[index];
}

function ChangedSpecialtySelect()
{
     var
index=document.getElementById("specialtyselect").selectedIndex;
     SpecialtySelected=document.getElementById("specialtyselect")
.options[index].text;
     document.getElementById("specialtyfrom").innerText="Бажана
спеціальність: "+SpecialtySelected;
     var data=new FormData();
     data.append('specialty',SpecialtySelected);
     var req=new XMLHttpRequest();
     req.addEventListener('load',filluniversities,false);
     req.open("POST","processuniversities.php",true);
     req.send(data);
}
</script>
<div           style="position:absolute;width:100%;top:-3px;text-
align:center;font-size:12">
Select the specialty to enter!
</div>
<div            style="position:absolute;width:40%;top:-3px;text-
align:left;z-index:10">
<form>
<p        style="font-size:20"        id="specialtyfrom">Виберіть
спеціальність для вступу...</p>
<br>
<table width=100% border=0>
<tr><td colspan=3>
<textarea rows=5 cols=76 id="specialtyinfo">
</textarea>
</td></tr>
```

```
<tr>
<td align="center">Університети</td>
<td align="center">Підрозділи</td>
<td align="center">Інші спеціальності</td>
</tr>
<tr>
<td>
<select         style="width:181px"         id="universityselect"
onchange="ChangedUniversitySelect();">
<option>Select some university...
</option>
</select>
</td>
<td>
<select          style="width:181px"          id="divisionselect"
onchange="ChangedDivisionSelect();">
<option>Select where to learn...
</option>
</select>
</td>
<td>
<select       style="width:181px"       id="otherspecialtyselect"
onchange="ChangedOtherSpecialtySelect();">
<option>Select where to learn...
</option>
</select>
</td></tr><tr><td>
<textarea rows=15 cols=23 id="universityinfo">
</textarea></td><td>
<textarea rows=15 cols=23 id="divisioninfo">
</textarea></td><td>
<textarea rows=15 cols=23 id="otherspecialtyinfo">
</textarea></td></tr>
<tr><td colspan=3>
</td></tr>
</table>
</form>
</div>
<div    style="position:absolute;width:50%;top:-3px;right:0;text-
align:center;z-index:10">
<p        style="font-size:20"        id="specialtyfrom">Виберіть
спеціальність, на яку вступаємо...</p>
<br>
<select          style="width:181px"          id="specialtyselect"
onchange="ChangedSpecialtySelect();">
<option>Оберіть спеціальність...</option>
<script>
for(i=0;i<text.length;i++)
{
     document.write("<option
value=\""+text[i]+"\">"+text[i]+"</option>");
}
</script>
```

```
</select><br><br>
<img      style="visibility:hidden"      src="images/default.jpg"
width=90% id="universityimage">
</div>
</body>
</html>
```

# ДОДАТОК Б. ВИСХІДНИЙ ТЕКСТ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ: ФАЙЛ PROCESSUNIVERSITIES.PHP.

```
<?php
header('Content-Type: text/html; charset=UTF-8');
$hostname="localhost"; $username="root"; $password="";
$dbName="entering";
$link=mysql_connect($hostname,$username,$password)   or   die("No
connect to DB!");
mysql_set_charset('utf8',$link);
mysql_select_db("$dbName",$link) or die("Cannot select DB!");
$query="SELECT
SpecialtyName,SpecialtyNumber,SpecialtyZNODisciplines,SpecialtyT
extAbout            FROM            specialties            WHERE
SpecialtyName='".$_POST["specialty"]."';";
$result=mysql_query($query);
$num_rows=mysql_num_rows($result);
for($i=0;$i<$num_rows;$i++)
{
     $row=mysql_fetch_array($result);
     print  "Спеціальність  ".$row['SpecialtyName'].": має номер
".$row['SpecialtyNumber'].".  ".$row['SpecialtyTextAbout']."  Для
вступу      необхідно      здати      ЗНО      з      предметів:
".$row['SpecialtyZNODisciplines'];
}

print "*";
$query="SELECT
UniversityName,UniversityDescription,UniversityNumberOfStudents,
UniversityPicture   FROM   universities   WHERE   UniversityID   IN
(SELECT  SIUUniversityID  FROM  specialtiesinuniversities  WHERE
SIUSpecialtyID=(SELECT   SpecialtyID   FROM   specialties   WHERE
SpecialtyName='".$_POST["specialty"]."'));";
$result=mysql_query($query);
$num_rows=mysql_num_rows($result);
for($i=0;$i<$num_rows;$i++)
{
     $row=mysql_fetch_array($result);
     print
$row['UniversityName'].";".$row['UniversityDescription'].";".$ro
w['UniversityNumberOfStudents'].";".$row['UniversityPicture'].";
\n";
}
?>
```

# ДОДАТОК В. ВИСХІДНИЙ ТЕКСТ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ: ФАЙЛ PROCESSDIVISIONSANDSPECIALTIES.PHP.

```
<?php
header('Content-Type: text/html; charset=UTF-8');
$hostname="localhost"; $username="root"; $password="";
$dbName="entering";
$link=mysql_connect($hostname,$username,$password)   or   die("No
connect to DB!");
mysql_set_charset('utf8',$link);
mysql_select_db("$dbName",$link) or die("Cannot select DB!");
$query="SELECT  DivisionName,DivisionType,DivisionContacts  FROM
divisions WHERE DivisionUniversity IN (SELECT UniversityID FROM
universities                                                WHERE
UniversityName='".$_POST["universityname"]."');";
$result=mysql_query($query);
$num_rows=mysql_num_rows($result);
for($i=0;$i<$num_rows;$i++)
{
     $row=mysql_fetch_array($result);
     print
$row['DivisionName'].";".$row['DivisionType'].";".$row['Division
Contacts'].";\n";
}

print "*";
$query="SELECT
SpecialtyName,SpecialtyNumber,SpecialtyZNODisciplines        FROM
specialties  WHERE  SpecialtyID  IN  (SELECT  SIUSpecialtyID  FROM
specialtiesinuniversities      WHERE      SIUUniversityID=(SELECT
UniversityID          FROM          universities          WHERE
UniversityName='".$_POST["universityname"]."'));";
$result=mysql_query($query);
$num_rows=mysql_num_rows($result);
for($i=0;$i<$num_rows;$i++)
{
     $row=mysql_fetch_array($result);
     print
$row['SpecialtyName'].";".$row['SpecialtyNumber'].";".$row['Spec
ialtyZNODisciplines'].";\n";
}
?>
```