Київський національний торговельно-економічний університет

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«WEB-застосування з підтримки діяльності служби евакуації на автомобільних шляхах»

Студента 4 курсу, 13 групи
спеціальності
122 «Комп'ютерні науки»
спеціалізації
«Комп'ютерні науки»

Дзюба Ілля Володимирович

_____
підпис студента

Науковий керівник
кандидат педагогічних наук,
доцент

_____
підпис керівника

Дивак Володимир Валерійович

Гарант освітньої програми
доктор фізико-математичних наук,
професор

_____
підпис керівника

Демідов Павло Георгійович

Київ 2021

Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»
Спеціалізація «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«18» грудня 2020р.

Завдання
на випускну кваліфікаційну роботу студенту

Дзюбі Іллі Володимировичу (прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи (проєкту):
«WEB-застосування з підтримки діяльності служби евакуації на автомобільних шляхах»
Затверджена наказом ректора від *«15» грудня 2020 р. № 3780*

2. Строк здачі студентом закінченої роботи:_____

3. Цільова установка та вихідні дані до роботи:
Мета роботи: створення WEB-сайту для підтримки роботи сайту служби евакуації на автомобільних шляхах.
Об'єкт дослідження: WEB-сайт для служби евакуації.
Предмет дослідження: технологія створення WEB-сайту для автономної роботи служби евакуації на автомобільних шляхах.

4. Перелік графічного матеріалу:_____
_____
_____
_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Дивак В.В.		
2	Дивак В.В.		
3	Дивак В.В.		

6. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом):

ВСТУП

1. ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

1.2 Огляд наявних аналогів

1.3 Постановка задачі

1.4 Висновки до розділу

2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

2.2 Вихідні дані

2.3 Висновки до розділу

3. ПРОГРАМНЕ ТА ЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Засоби розробки

3.2 Вимоги до технічного забезпечення

3.3 Архітектура програмного забезпечення

4. ТЕХНОЛОГІЧНИЙ РОЗДІЛ

4.1 Керівництво користувача

4.2 Керівництво програміста

4.3 Тексти програми

4.4 Тестування та аналіз результатів роботи

4.5 Висновки до розділу

ВИСНОВКИ

СПИСОК ВИКОРАСТИНАХ ДЖЕРЕЛ

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*		
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*		
3	*Вступ*		
4	*Розділ 1. Загальне положення, опис предметного середовища, огляд наявних аналогів, постановка задачі та висновки до розділу*		
5	*Розділ 2. Інформаційне забезпечення, вхідні дані, вихідні дані та висновки до розділу.*		
6	*Розділ 3. Програмне та технічне забезпечення, засоби розробки, вимоги до технічного забезпечення за архітектура програмного забезпечення.*		
7	*Розділ 4. Технічний розділ, керівництво користувача, керівництво програміста, текст програми, тестування та аналіз результатів роботи та висновки до розділу.*		
8	*Висновки*		
9	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*		
10	*Попередній захист випускної кваліфікаційної роботи*		
11	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*		
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*		
13	*Публічний захист випускної кваліфікаційної роботи*		

8. Дата видачі завдання _____

9. Керівник випускної кваліфікаційної роботи

Дивак В.В._____

(прізвище, ініціали, підпис)

10. Гарант освітньої програми Демідов П.Г._____

(прізвище, ініціали, підпис)

11. Завдання прийняв до виконання студент-дипломник

Дзюба І.В._____

(прізвище, ініціали, підпис)

12. Відгук керівника випускної кваліфікаційної роботи

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Керівник випускної кваліфікаційної роботи

_____

(підпис, дата)

13. Висновок про випускну кваліфікаційну роботу

Випускна кваліфікаційна робота студента Дзюби І.В.

(прізвище, ініціали)

може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми_____Демідов П.Г.____

(підпис, прізвище, ініціали)

Завідувач кафедри _____Пурський О.І.____

(підпис, прізвище, ініціали)

«_____»_____2021 р.

WEB-ЗАСТОСУВАННЯ З ПІДТРИМКИ ДІЯЛЬНОСТІ СЛУЖБИ ЕВАКУАЦІЇ НА АВТОМОБІЛЬНИХ ШЛЯХАХ

ЗМІСТ

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

ЕОМ – електронно-обчислювальна машина з необов'язковими додатковими пристроями та системними елементами.

ОС – операційна система, це базовий комплекс програм, що виконує управління апаратною складовою комп'ютера або віртуальної машини.

ПЕОМ – персональна ЕОМ.

ПЗ – програмне забезпечення; сукупність програм системи обробки інформації і програмних документів, необхідних для експлуатації цих програм.

ПК – персональний комп’ютер; електронна обчислювальна машина, що призначена для зберігання і переробки інформації.

ВСТУП

Об’єктом дослідження є WEB-сервіс з виклику евакуаційної служби «*SOSтачка*», що призначене для допомоги водіям у разі ДТП або несправності машини, щоб евакуювати їх.

Предметом дослідження є комплекс заходів по створенню та просуванні програмного забезпечення.

Результатом дослідження є розроблене програмне забезпечення «*SOSтачка*» на мові *HTML, CSS* для ОС *Windows*.

Аргументом для створення програмного забезпечення являється важливість допомоги водіям у складному становищі, аваріях або при поломці. У наш дуже часто трапляються аварії і потрібна допомога евакуатора, являється необхідною і водій ніяк не впорається без допомоги. Виходячи з цього можна назвати її найнеобхіднішою у наш час на усіх дорогах. Також слід зауважити, що майже всі служби евакуаторів викликаються за допомогою мобільного телефона, так як це швидко та зручно.

Не менш важливою причиною створення програми є те, що є багато аналогів, але з дуже неприємним та складним інтерфейсом. У сферах автотранспорту, швидкий виклик евакуатору у наш час необхідний.

Розробка програмного забезпечення для евакуації машин є важливим тому, що таких програм існує безліч але зручних небагато.

У наш час досить розповсюджене використання мобільних додатків. Це зумовлене зручністю їх застосування. Але це потребує постійного доступу до мережі. Тому важливістю створення програмного забезпечення для ПК є можливість користування програмою без під’єднання до Інтернету.

Хоч прикладні програми є не досить популярними нині у використані, їх великою перевагою можна вважати можливість використовувати програму в оффлайн режимі.

Програма «*SOSтачка!*» призначена для виклику евакуатора за допомогою лише мобільного номера та наявності інтернету. Цей додаток спроможний повністю заміни усі аналоги.

1. ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1. Опис предметного середовища

1.1.1. Опис процесу діяльності

У наш час на технічному ринку з’являється безліч програм для виклику евакуатора. Наприклад, *«Евакуатор плюс», «Єкспресс Евакуатор»,* всі вони є мобільними додатками. Також, існує безліч онлайн-сервісів для виклику евакуатора, що потребують постійного доступу в Інтернет. А для користувача важлива доступність та легкість виклику евакуатору. Зручним способом є використання мобільного додатку, що надає можливість користуватися програмою будь-де. Але це потребує постійного доступу до Інтернету. Це зумовлює важливість створення прикладної програми, робота якої не вимагає доступу до мережі Інтернет.

Доцільно буде створювати таку програму для роботи під управлінням ОС *Windows*. Програма буде доступна на ОС *Windows*. Сучасні ОС сімейства *Windows* – це графічні, інтерактивні, багатозадачні ОС корпорації *Microsoft*. Сімейство ОС Windows складається з декількох груп:

Windows 9x. Група ОС для 16 і 32 – розрядних процесорів. Вироблялися з 1995 по 2000 рік. В даний час ОС цієї групи є застарілими;

Windows NT. Це група сучасних ОС. Всі ОС цієї групи бувають 32 і 64-розрядними і працюють відповідно на 32 і 64-розрядних процесорах. Саме до цієї групи відносяться популярні системи *Windows XP, Windows 7, Windows 8*. Є ОС, призначені для управління серверними комп'ютерами ;

Windows для смартфонів. До цієї групи відносяться ОС *Windows CE, Windows mobile, Windows Phone, Windows 10 Mobile*. Системи цієї групи можна придбати виключно в складі готових смартфонів.;

Windows Embedded. Група вбудованих ОС реального часу застосовуються для різних спеціалізованих пристроїв. Наприклад, для інформаційних і платіжних терміналів, систем відеоспостереження на дорогах.

Пакет *Microsoft Windows* включає стандартні додатки, такі як браузер (*Internet Explorer*), поштовий клієнт (*Outlook Express*), програвач (*Windows Media Player*). За допомогою технологій *COM* і *OLE* їхні компоненти можуть вбудовуватися в інші застосунки, у тому числі й сторонніх виробників. Ці продукти позиціонуються як безкоштовні і можуть бути вільно завантажені з офіційного сайту *Microsoft*, проте для установки деяких з них необхідно мати робочу ліцензію *Windows*. Запуск цих програм під іншими операційними системами можливий за допомогою емуляторів середовища *Windows*. Існують також версії деяких з них, спеціально розроблені для інших операційних систем. Ці версії поступаються оригінальним версіям набором функцій і можливостей, а також частотою оновлення.

Навколо факту включення таких стандартних продуктів в ОС *Windows* розгортається багато суперечок і дискусій, оскільки це створює серйозні перепони для розповсюдження конкуруючих продуктів. Так, часто оскаржують лідерство і ставлять під сумнів якість браузера *Internet Explorer*, пояснюючи його популярність входженням в пакет *Windows* і поганою обізнаністю користувачів про наявність альтернатив.

1.1.2. Опис функціональної моделі

На початковому етапі створення програмного продукту створюється функціональна модель програми. Для того аби мати уявлення про майбутнє програмне забезпечення, яке буде створене.

У наш час для розробки моделі ПЗ використовується мова *UML* (англ. *Unified Modeling Language*, уніфікована мова моделювання).[8]

UML є графічною мовою і містить в собі шість основних моделей: модель прецедентів, модель аналізу, модель проектування, модель розгортання, модель реалізації та модель тестування. Основні дії при визначенні вимог направлені на побудову моделі прецедентів, яка відображує функціональні вимоги до системи, що моделюється. Модель прецедентів зображується у

вигляді діаграми, на якій зображено відношення між акторами та прецедентами в системі.

Основними елементами проектованої системи є актори і варіанти використання. Актори взаємодіють з системою за допомогою варіантів використання, які в свою чергу показують, що система надає актору.

Під час створення діаграми варіантів використання для програми *«SOSтачка»* були описаний один актор (користувач) та його варіанти використання (його дії).

Варіанти використання:

– перегляд типів евакуаторів;
– перегляд інформації про службу;
– виклик евакуатору різними варіантами;

Діаграма прецедентів зображена на рисунку 1.1.

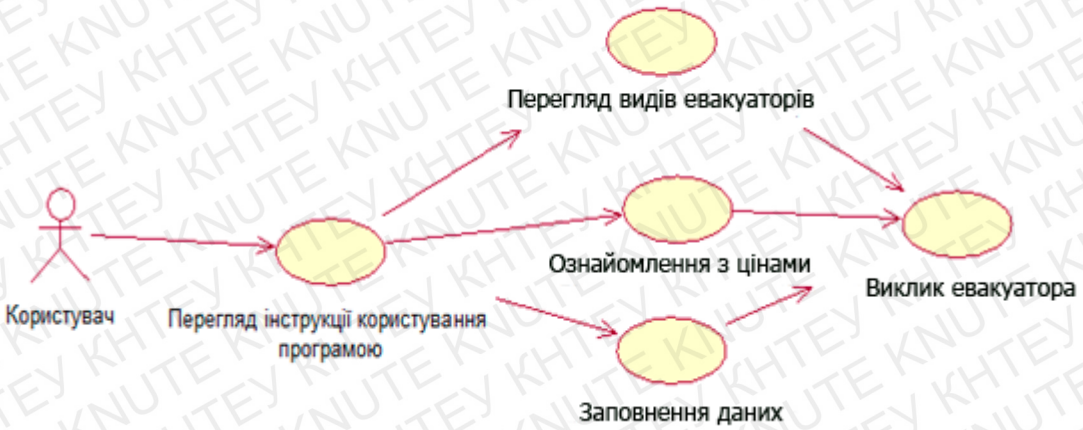

Рис 1.1. Діаграма прецедентів

1.2. Огляд наявних аналогів

1.2.1. Онлайн сервіс «EXPRESS T».

Онлайн сервіс «*EXPRESS T*» являє собою сайт для виклику евакуатора. Сервіс є доволі простим у використані з невиразним інтерфейсом, зображеним на рисунку 1.2.

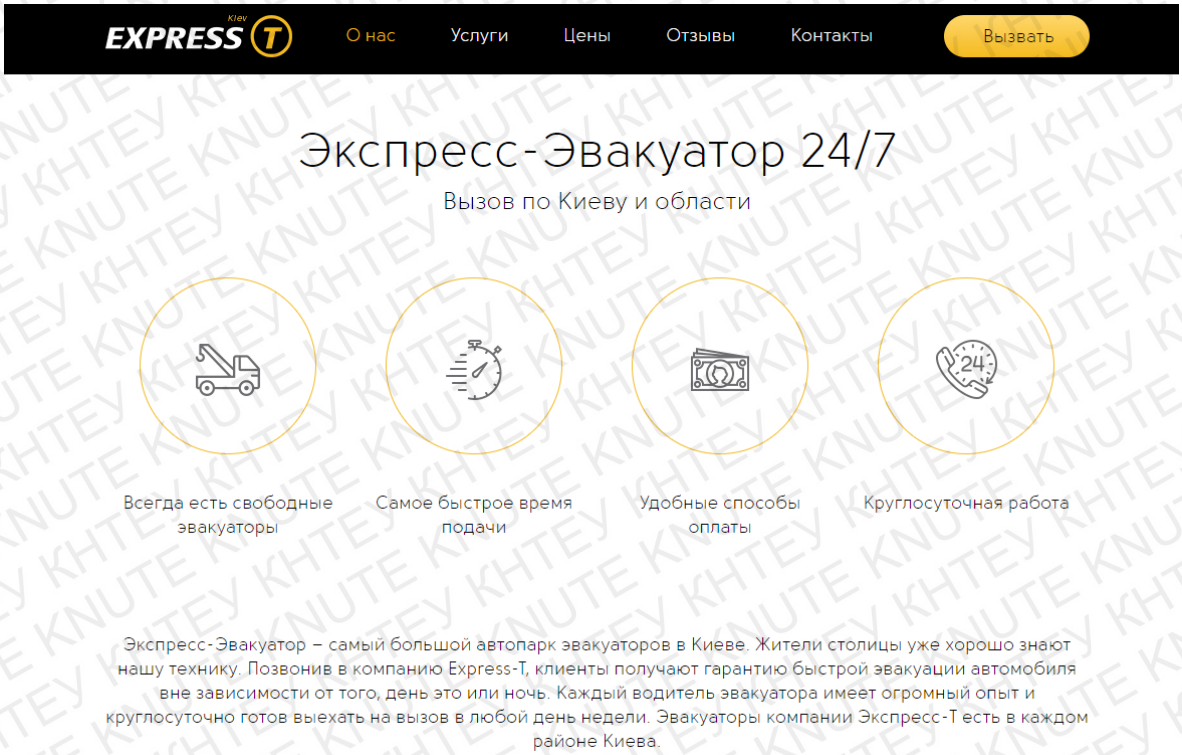

Рис 1.2. Інтерфейс сайту «EXPRESS T»

До переваг сервісу можна віднести зручність у наданні інформаціїї миші, також сервіс збагачений великою кількістю матеріалів для перегляду користувачами.

Недоліком сервісу є неможливість користування без під'єднання до мережі, а також досить несучасний інтерфейс.

1.2.2. Онлайн сервіс «*Евакуатор плюс*»

Наступний сайт - *Евакуатор плюс.* Особливістю сервісу є можливість обрати виклик евакуатору у будь який куток країни. Інтерфейс сайту не складний та примітивний, зображеним на рисунку 1.3.

Рис 1.3. Інтерфейс сайту Евакуатор плюс

1.2.3. Онлайн сервіс «*auto-sos*»

Сервіс для виклику евакуатора є досить багатофункціональним, але оснащений досить примітивним інтерфейсом, зображеним на рисунку 1.4.

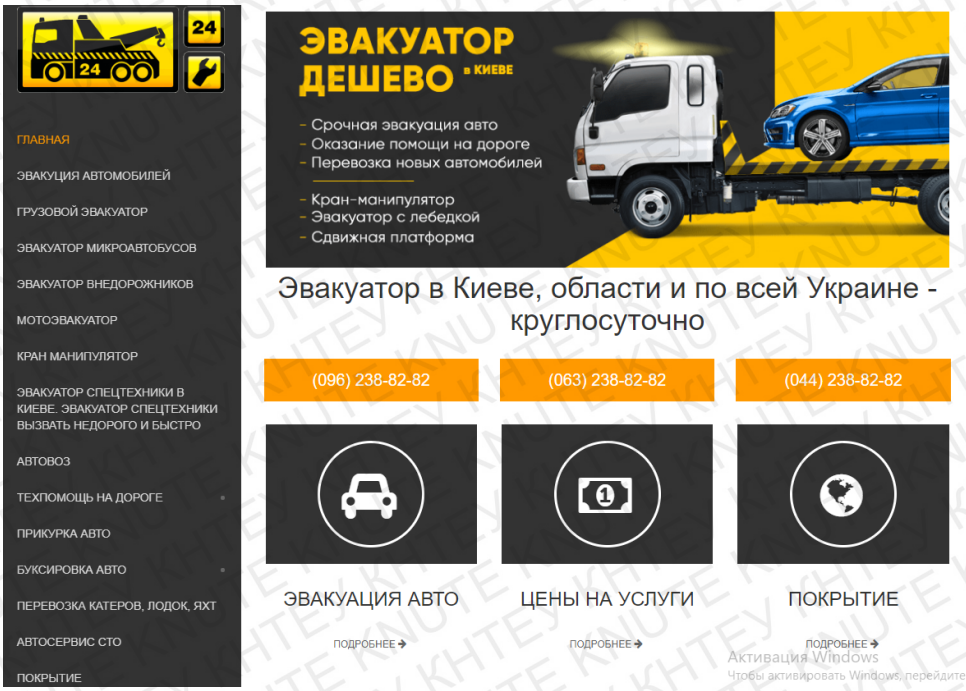

Рис 1.4. Інтерфейс сайту auto-sos

Перевагами сервісу є:

– можливість транспортування морського транспорту;

– наявність СТО;

– наявність прикурки авто;

– наявність трала;

Недоліками, як і в попередніх сервісах, являються:

– примітивний інтерфейс;

– відсутність можливості оффлайн.

1.3. Постановка задачі

1.3.1. Призначення розробки

Метою написання дипломного проекту є створення програмного продукту для виклику евакуатора.

Проектування і розробка програмного продукту включає:

– затвердження первинного технічного завдання розробки ПЗ;

– визначення структурної схеми ПЗ – розташування меню, розділів, контенту і панелі налаштувань;

– дизайн - створення графічних елементів макету, стилів і елементів навігації;

– розробка програмного коду, модулів, бази даних і інших елементів ПЗ, необхідних в проекті;

– тестування програмного продукту.

1.3.2. Цілі та задачі розробки

Основними цілями створення системи є:

– збільшення кількості користувачів;

– стабілізація на ринку комп’ютерних програм, що не потребують доступу в Інтернет.

Для досягнення поставлених цілей мають бути реалізовані такі задачі:

– вивчення потреб користувачів та формування опису вимог до програми;

– визначення архітектури проекту;

– розробка макету графічного інтерфейсу продукту;

– створення програмного продукту;

– встановлення додаткових можливостей користувача.

1.4. Висновки до розділу

Можна зробити висновок, що обрана тема для розробки програми є дуже актуальною та перспективною на наш час. В даному розділі була поставлена мета, цілі та задачі створення системи.

Була розроблена діаграма варіантів використання та визначено основні елементарні складові системи, їх функцію та сценарії взаємодії. Проаналізовано наявні аналоги на ринку та порівняно їх основні переваги і недоліки аналіз, що потрібно для унікальності и зручності продукту.

Таким чином, були охарактеризовані загальні положення при розробці програмного продукту для ОС *Windows* на мові *HTML*. [9]

2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Вхідні дані

Вхідними даними є інформація, яку надає користувач, використовуючи елементи інтерфейсу, які потім зберігаються в базі даних та передаються програмі на подальшу обробку.

Вхідні дані:

– номер телефону або e-mail;

– ім’я;

2.2. Вихідні дані

Вихідними даними є оброблена програмою інформація, яка надходить до оператора. До неї належать:

– номер телефону, за яким оператор виходить на зв'язок до клієнта;

2.3. Висновки до розділу

В даному розділі описано важливість інформаційного забезпечення, вхідні дані для кожного з розділів та їх вміст, вихідні дані програмного продукту.

3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Засоби розробки

3.1.1. Середовище розробки

HTML — стандартна мова розмітки для створення веб-сторінок і веб-додатків . З Cascading Style Sheets і JavaScript, вона утворює тріаду основних технологій для World Wide Web.[1]

Веб-браузери отримують HTML-документи з веб-сервера і передають документи в веб-сторінки. *HTML* описує структуру веб-сторінки семантично і спочатку включені сигнали для зовнішнього вигляду документа.

До складу також включені пристосовані під особливості платформи Windows розширені інструменти:

– перевірки сумісності з минулими випусками;

– виявлення проблем з продуктивністю;

– моніторингу споживання пам'яті;

– оцінки зручності використання.

3.1.2. Опис мови

Гіпертекст – це текст, який містить зв'язки з іншими документами («гіперзв'язки» чи «гіперпосилання»); читач має змогу перейти до пов'язаних документів безпосередньо з вихідного (первинного) тексту, активізувавши посилання. Такі посилання створюються за допомогою спеціальних кодів, що задають форматування тексту. Ці коди визначені у мові розмітки HTML.

HTML (англ. *HyperText Markup Language* — мова розмітки гіпертекстових документів) — основана на SGML – текстова мова розмітки, призначена для розмітки документів, які містять текст, зображення, гіперпосилання, тощо. Процес розміщення в текст кодів HTML називають *розміткою.*

Документи створені мовою *HTML* називають *HTML*-документи. Вони нагадують звичайні текстові файли за винятком того, що деякі символи в них (так звані теги (*tag*)) інтерпретуються як мітки.

Тег (від англ. *tag* - "ярлик, етикетка, бірка; розмічувати, маркувати") - це символ розмітки мови HTML, який подається в кутових дужках <>. Тег ще називають *дескриптором*. Використання дескрипторів дає змогу форматувати *HTML*-документи.

За допомогою дескрипторів у *HTML*-документі можна форматувати такі елементи: параграфи, розділи, абзаци, списки, малюнки, таблиці, колонтитули, індекси, зміст тощо. Кожен елемент має свою унікальну назву, яка записується латинськими літерами і не чутлива до їх регістру. В загальному вигляді елемент має три складові:

- теги (початковий та кінцевий);
- атрибути;
- зміст (контент).

Тег — це назва елементу, записана у кутових дужках (< >). Атрибути задають технічну інформацію про елемент. Зміст елементу — це вся необхідна текстова та графічна інформація документу, яка буде відтворюватися браузером на екрані.

Багато хто вважає, що елементи — це і є теги (наприклад, "тег p"). Проте варто пам'ятати, що елемент — це здебільшого дві складові (теги і зміст), а тег — це складова елемента. Наприклад, елемент *head* завжди представлений в документі, навіть якщо обидва тега, *<head>* та *</head>*, відсутні в розмітці.

Перегляд *HTML*-документу здійснюється за допомогою веб-оглядача (наприклад, *Internet Explorer, Google Chrome, Opera* та ін.).

Мова *HTML* набула на сьогодні певних розширень. На її основі було створено нові мови розмітки. Наприклад, поєднання статичної мови розмітки *HTML*, вбудованої скриптової мови *JavaScript* (сценарії виконуються на стороні клієнта), *CSS* (каскадних таблиць стилів) і *DOM* (об'єктної моделі

документа) утворило нову концепцію створення веб-сайту, що розглядає *HTML*-документ як об'єктну структуру та дістала назву *DHTML (Dynamic HTML)*. Ця концепція може бути використана для створення додатків в браузері: наприклад для навігації або для додання інтерактивності формам. Також *DHTML* може бути використаний для динамічного перетягування елементів по екрану і може служити як інструмент для створення заснованих на браузері відео-ігор.

XHTML (англ. *Extensible Hypertext Markup Language* — розширювана мова розмітки гіпертексту) — мова розмітки, що базується на основі *XML* та повторює і розширює можливості мови *HTML*. Головна відмінність *XHTML* від *HTML* полягає в опрацюванні документа. Документи *XHTML* опрацьовуються своїм модулем, під час виконання якого помилки, що допущені розробником не виправляються.

У 1980 році фізик Тім Бернерс-Лі запропонував систему *ENQUIRE*, яка мала полегшити сумісне використання документів.

У 1989 році Бернерс-Лі запропонував впровадити на базі Internet гіпертекстову систему документів.

Вже наприкінці 1990 року він розробив мову *HTML* і створив веб-оглядач та серверне програмне забезпечення для запропонованої ним системи гіпертекстових документів.

Наприкінці 1991 року Тім Бернерс-Лі опублікував в мережі Інтернет перший загальнодоступний опис мови *HTML*, відомий як документ «*HTML* теги» (*HTML Tags*). В ньому були описані 20 елементів найпершої схеми розмітки HTML-документів. 13 з яких до цих пір використовуються у мові HTML.

Бернерс-Лі розглядав *HTML* як похідну мову від *SGML*, і в середині 1993 року її офіційно визнали такою, опублікувавши першу специфікацію HTML: "*Hypertext Markup Language (HTML)*", авторами якої були Тім Бернерс-Лі та Ден Конолі.

В 1993 році Дейв Раджетт запропонував стандартизувати розмітку для визначення таблиць та інтерактивних форм.

На початку 1994 року створено робочу групу *HTML (HTML Working Group)*, яка займалась проблемами стандартизації мови розмітки *HTML*.

В 1995 році робоча група *HTML* завершила роботу над першою специфікацією «*HTML* 2.0», що мала бути використана як базовий стандарт для подальших вдосконалень *HTML*. Версія 2.0 окреслювала чіткі відмінності між новим виданням специфікації та попередніми проектами.

З 1996 року специфікації *HTML* затверджувались консорціумом *W3C*, враховуючи доповнення до розмітки, що впроваджувалися компаніями-розробниками веб-оглядачів.

У 2000 році *HTML* стала міжнародним стандартом (*ISO/IEC* 15445:2000).

Остання специфікація *HTML*, опублікована *W3C* наприкінці 1999 року, має назву «*HTML* 4.01 Recommendation».

Версії мови HTML

Тім Бернерс-Лі представив *HTML* в дослідницькому центрі *CERN* в Женеві в 1989 році.

HTML (без номера версії, 3 листопада 1992): найперша версія, орієнтована лише на текст.

HTML (без номера версії, 30 квітня 1993): до тексту додаються атрибути, які визначають курсивне або жирне написання літер, та зображення.

HTML+ (листопад 1993): заплановані доповнення, які потрапили до наступних версій, але ніколи не були відокремлені як *HTML+*.

HTML 2.0 (листопад 1995): визначена стандартом *RFC* 1866 версія з підтримкою форм. Статус цього стандарту вже «історичний», також визнані застарілими попередні версії.

HTML 3.0: версія, яка не зазнала поширення, оскільки разом із випуском браузера *Netscape Navigator* версії 3, цей стандарт вже був застарілим.

HTML 3.2 (14 січня 1997): в цій версії були додані численні можливості, такі як таблиці, обтікання текстом зображень, інтеграція аплетів.

HTML 4.0 (18 грудня 1997): в цій версії були додані таблиці стилів, скрипти та фрейми. Також, відбулось розділення на *Strict* (суворе дотримання стандартів), *Frameset* (з підтримкою фреймів), *Transitional* (перехідний). 24 квітня 1998 було випущено виправлену версію цього стандарту.

HTML 4.01 (24 грудня 1999): заміна версії *HTML* 4.0, містить численні дрібні виправлення.

HTML 5 (5 квітня 2008): *HTML* 5 має новий словник побудований на основі HTML 4.01 та XHTML 1.0. Також перероблена і розширена пов'язана з HTML специфікація DOM.

XHTML 1.0 (26 січня 2000): висловлення стандарту *HTML* 4.01 засобами *XML*. 1 серпня 2002 було випущено оновлену редакцію стандарту.

XHTML 1.1 (31 травня 2001): після того, як *XHTML* буде розділено на модулі, стандарт *XHTML* 1.1 визначатиме сувору версію, в якій не буде запроваджених *HTML* 4 можливостей *Frameset* та *Transitional*.

XHTML 2.0 (в розробці): ця версія вже не базується на *HTML* 4.01 і додає деякі нові теги. Буде завершено розділення між представленням та вмістом.

Розмітка документів – це опис різних фрагментів документа та їх взаємного розташування. Розмітка здійснюється за допомогою дескрипторів (тегів) мови *HTML*.

Переглянути розмітку будь-якого *HTML*-документа можна, якщо відкрити його у простому текстовому редакторі, типу Блокнот. Або, якщо документ відкрити у веб-оглядачі, наприклад, *Internet Explorer*, обрати команду Вид-В виде *HTML*. Якщо веб-оглядач Опера, то достатньо клацнути правою кнопкою миші на документі та обрати команду «Исходный код».

Для вивчення розмітки мови *HTML* потрібні дві речі:

– будь-який веб-оглядач, тобто, програма, придатна для перегляду *HTML*-файлів;

– будь-який редактор текстових файлів, наприклад Блокнот (*Notepad*).

Свої перші *HTML*-файли можна розробляти на локальному диску. При цьому комп`ютер може і не мати доступу до мережі Інтернет. Під час роботи, *html*-документ може бути одночасно відкритий і в Блокноті, і у веб-оглядачі. Збергіаючи зміни в Блокноті, потрібно натиснути кнопку оновлення сторінки, наприклад F5 чи *Ctrl+R*, щоб побачити ці зміни в *HTML*-документі.

Як почати:

– Відкрити програму Блокнот.
– Відкрити вікно збереження документа.
– Вибрати тип файла «Все файлы».
– Ввести ім’я файла із зазначенням його розширення – *html* (наприклад, *text.html, example.html, memory.html*).
– Ввести структуру документа та зберегти.
– Закрити документ у програмі Блокнот.
– Відкрити документ у веб-оглядачі.

Наступного разу, коли потрібно здійснити оновлення *HTML*-документу, цей файл потрібно відкривати за допомогою програми Блокнот.

Загальна структура

HTML-документ складається з трьох основних елементів:

Декларація типу документу – (англ. *Document type declaration, Doctype*), на початку документа, в якій визначається тип документа (*DTD*).

Шапка документа (знаходиться в межах елементу *head*), в якій записано загальні технічні відомості або додаткові відомості про документ, які не відтворюються безпосередньо у веб-оглядачі; наприклад, теги *title* визначають заголовок документа.

Тіло документа (може знаходитися в елементах *body*), в якому містяться основні відомості документа, його зміст.

Для того, щоб веб-оглядач розпізнавав ці елементи, назва кожного елементу має обмежуватись кутовими дужками <>. Елемент, обмежений кутовими дужками і є дескриптором.

Дескриптори можуть бути двох видів: початковий та кінцевий. Наприклад, для позначення типу документа, одного дескриптора *<html>* не достатньо, потрібен ще й кінцевий – *</html>*, який позначається так само, але перед ключовим словом додається знак «слеш».

Отже, для визначення типу документу потрібно вказати два дескриптори:

<html></html>

Деякі дескриптори, наприклад *
*, не містять змісту, тому не мають кінцевого дескриптора. Розмітка може не мати початкового та кінцевого дескриптора (наприклад, дескриптор *<head>*), проте він завжди буде представлений в документі.

Всередині дескрипторів можна додавати інші дескриптори. Дескриптор *<html></html>* є основним та містить в собі всі інші дескриптори. Тому на початку *HTML*-документа завжди стоїть початковий дескриптор *<html>*, а в кінці всього документу кінцевий – *</html>*.

Розглянемо використання дескрипторів на прикладі загальної структури *HTML* документа:

```
<html>
  <head>
    <title>Мій перший HTML-документ</title>
  </head>
  <body>
  Вчимось розмічувати HTML-документ!
  </body>
</html>
```

Елементи являють собою базові компоненти розмітки мови *HTML*. Кожен елемент має три складові: теги (початковий і кінцевий), атрибути та зміст (контент). Існують певні настанови щодо кожного дескриптора, атрибута та змісту елемента, які треба виконувати для того, щоб *HTML*-документ правильно відображався у веб-оглядачі.

Атрибути записуються в початковому дескрипторі одразу після його назви, зміст записується між початковим та кінцевим дескрипторами. Наприклад:

<ПОЧАТКОВИЙ ДЕСКРИПТОР АТРИБУТ="значення">зміст елементу</КІНЦЕВИЙ ДЕСКРИПТОР>

Більшість з атрибутів являє собою пару «назва-значення», розділених між собою знаком «рівно», та записаних у початковому тегу одразу після назви елемента. Значення атрибуту може бути вказано в лапках (подвійними або одиничними), також, якщо значення атрибуту складається з певних символів, його можна не виділяти лапками зліва. Проте, не виділення значення атрибутів в лапки вважається небезпечним кодом. На відміну від атрибутів виду «назва-значення», є певні атрибути, що впливають на елемент, якщо з'явилась тільки назва атрибуту (без значення) в початковому дескрипторі (наприклад, атрибут *border* дескриптора *<table>).*

Більшість елементів можуть мати будь-який з загальних атрибутів:

Атрибут *id* впроваджує унікальний ідентифікатор елементу по всьому документу. Доданий до *URL* документу, він впроваджує глобальний унікальний ідентифікатор елементу.

Це може використовуватися:

- таблицями стилів для впровадження презентаційних властивостей;
- браузерами для фокусування уваги на певному елементі;
- скриптами для виконання дій над елементом.

Елементи структурної розмітки застосовуються задля опису семантики тексту, іншими словами ці елементи описують призначення тексту свого контенту. Вони не зазначають ніякого спеціального (візуального) відтворення тексту, проте більшість браузерів мають стандартні стилі форматування для кожного елемента. Для подальшого стилізування тексту рекомендується використовувати Каскадні таблиці стилів (*CSS*).

Елементи візуальної розмітки застосовуються задля опису візуальних ефектів тексту, не зазначаючи при цьому функції тексту свого контенту.

Остання чинна специфікація *HTML* 4.01 визначає більшість з цих елементів такими, що не рекомендується застосовувати у розмітці.

Елементи розмітки гіпертексту застосовуються задля з'єднання частин документу з іншими документами.

Базові типи даних

Оскільки *HTML* є похідною мовою від *SGML*, усі типи даних *HTML* ґрунтуються на базових типах даних *SGML* (наприклад, *PCDATA, CDATA, NAME, ID, NUMBER*).

Кожен елемент має дві властивості — атрибути і вміст, які мають певні значення. Всі можливі значення цих двох властивостей прописуються відповідно до визначених у *DTD* типів даних. Нижче наведено кілька типів даних *HTML*:

- *Color* — колір *sRGB*, записаний у шістнадцятковому вигляді, або одне з шістнадцяти службових слів;
- *ContentType* — тип умісту/носія;
- *Charset* — таблиця кодування символів;
- *Character* — мнемоніка або окремий символ із *UCS*;
- *Length* — *nn* розмір в пікселях, *nn*% — у відсотках;
- *URI* — Універсальний ідентифікатор ресурсу;
- *Datetime* — дата та час;
- *Script* — скрипт;
- *StyleSheet* — дані таблиць стилів;
- *Text* — текстові рядки.

Існують такі випадки, коли в документі потрібно використати якийсь символ, якого немає в обраній для документу кодовій таблиці. Для таких випадків можливо замінити символ на еквівалентне йому *SGML*-посилання на символ (мнемоніку).

Розрізняють мнемоніки двох видів:

- Цифрові мнемоніки (десяткові або 16-кові)
- Визначають кодову позицію символу із таблиці кодів *UCS*.
- Мнемоніки із певних сполучень символів

Такі мнемоніки використовують псевдоніми замість кодів символів. Проте в *HTML* не визначені псевдоніми для кожного символа із *UCS*.

HTML 4.01 підтримує три різні набори мнемонік:

- Мнемоніки для символів *ISO 8859-1* (*Latin-1*)
- Символи, математичні символи та грецькі літери
- Мнемоніки для символів розмітки та інтернаціоналізації

Валідація.

Так само як і кожна мова, будь-яка комп'ютерна мова має свою власну граматику, словник і синтаксис. І кожен документ, написаний цією мовою, має дотримуватися цих правил. *HTML* використовує машинно-зчитуючу граматику, яка називається *DTD*, механізм, успадкований від *SGML*.

Проте, так само як і тексти природної мови можуть містити граматичні помилки, документи, що використовують мови розмітки можуть не дотримуватись визначеної граматики. Процес перевірки документа на дотримання визначених мовою правил називають валідацією, а інструмент, який здійснює перевірку — валідатором. Документ, що пройшов цей процес без помилок називають валідним.

Згідно з цією концепцією, «валідація *HTML* розмітки» визначається як процес перевірки веб-документа за правилами граматики (визначеними в *DTD*), на які він посилається із елементу *doctype*.

Один із важливих принципів програмування: «Будьте консервативні в тому, що ви робите; будьте ліберальним в тому, що ви приймаєте».

Браузери дотримуються другої частини цього принципу: вони приймають веб-документи такими, які вони є, та намагаються відтворити їх на екрані, навіть якщо вони не використовують стандартний *HTML*. Зазвичай це означає, що браузер спробує «здогадатися» про те, що автор документу мав на увазі. Проблема полягає в тому, що різні браузери (або навіть різні версії одного браузера) зроблять різні припущення щодо одних і тих же не стандартних конструкцій, і навіть гірше: якщо *HTML*-код дуже відрізняється

від стандарту, браузер безнадійно заплутається і безладно відтворить сторінку на екрані, або навіть аварійно закриється.

Саме тому, дотримуватися першої частини принципу належить авторам документа, шляхом перевірки своїх документів на дотримання стандарту. Найкращий інструмент для цього — валідатор *HTML* розмітки.

Редактор *HTML* або *HTML*-редактор – програмне забезпечення, що дає змогу створювати і змінювати *HTML*-сторінки. Незважаючи на те, що *HTML*-код може бути написаний в простому текстовому редакторі (наприклад, Блокнот), спеціальні редактори для написання коду *HTML* мають широкий набір інструментарію, який дає змогу розробнику швидше і ефективніше створити *HTML*-документ. За функціональністю їх умовно можна поділити на дві категорії:

- Редактор, що показує тільки вихідний код.
- Редактор показує готову сторінку в режимі *WYSIWYG* (що бачиш, те й отримаєш).

Багато *WYSIWYG*-редакторів дають змогу одночасно працювати і з кодом сторінки.

Крім цього, існують різні системи управління вмістом, онлайн-редактори і конструктори для створення готових сторінок.

PHP — мова, у код якої можна вбудовувати безпосередньо *html*-код сторінок, які, у свою чергу, коректно оброблюватимуться *PHP*-інтерпретатором. Обробник PHP просто починає виконувати код після відкриваючого тегу *(<?php)* і продовжує виконання до того моменту, поки не зустріне закриваючий тег.

Велика різноманітність функцій *PHP* дає можливість уникати написання багаторядкових функцій, призначених для користувача, як це відбувається в *C* або *Pascal*.

Наявність інтерфейсів до багатьох баз даних

- у *PHP* вбудовані бібліотеки для роботи з *MySQL, PostgreSQL, SQLite, mSQL, Oracle, dbm, Hyperware, Informix, InterBase, Sybase*.
- завдяки стандарту відкритого інтерфейсу зв'язку з базами даних (*англ. Open Database Connectivity Standard, ODBC*) можна підключатися до всіх баз даних, до яких існує драйвер.

Мова PHP здаватиметься знайомою програмістам, що працюють в різних областях. Багато конструкцій мови запозичені з *C, Perl*. Код *PHP* дуже схожий на той, який зустрічається в типових програмах мовами *C* або *Pascal*. Це помітно знижує початкові зусилля при вивченні *PHP*. *PHP* — мова, що поєднує переваги *Perl* та *C* і спеціально спрямована на роботу в Інтернеті, мова з універсальним і зрозумілим синтаксисом. І хоча *PHP* є досить молодою мовою, вона здобула таку популярність серед *web*-програмістів, що в наш час є найпопулярнішою мовою для створення веб-застосунків (скриптів).

Стратегія *Open Source*, і розповсюдження початкових текстів програм в масах, безсумнівно справили сприятливий вплив на багато проектів, в першу чергу — *Linux* хоч і успіх проекту *Apache* сильно підкріпив позиції прихильників *Open Source*. Сказане відноситься і до історії створення *PHP*, оскільки підтримка користувачів зі всього світу виявилася дуже важливим чинником в розвитку проекту *PHP*. Ухвалення стратегії *Open Source* і безкоштовне розповсюдження початкових текстів *PHP* надало неоціненну послугу користувачам. Окрім цього, користувачі *PHP* в усьому світі є свого роду колективною службою підтримки, і в популярних електронних конференціях можна знайти відповіді, навіть на найскладніші питання.

Ефективність є дуже важливим чинником у програмуванні для середовищ розрахованих на багато користувачів, до яких належить і *web*. Важливою перевагою *PHP* є те, що ця мова належить до інтерпретованих. Це дозволяє обробляти сценарії з достатньо високою швидкістю. За деякими

оцінками, більшість *PHP*-сценаріїв (особливо не дуже великих розмірів) обробляються швидше за аналогічні їм програми, написані на *Perl*. Проте хоч би що робили розробники *PHP*, виконавчі файли, отримані за допомогою компіляції, працюватимуть значно швидше — в десятки, а іноді і в сотні разів. Але продуктивність *PHP* достатня для створення цілком серйозних веб-застосунків.

Історія *PHP* починається з 1995 року, коли Расмус Лердорф (англ. *Rasmus Lerdorf*) створив простий застосунок мовою *Perl*, що аналізував відвідування користувачами його резюме на веб-сайті. Потім, коли цим застосунком вже користувалися кілька осіб, а число охочих одержати його постійно збільшувалося, Лердорф назвав своє творіння Інструменти для особистої домашньої сторінки англ. *Personal Home Page Tools* версія 1 і виставив для вільного завантаження. З цієї миті почався небувалий зліт популярності *PHP*.

Як це завжди буває, терміново було потрібне доопрацювання і нові доповнення. Для їхньої реалізації Расмус створює нову версію пакету, тепер уже написану на C. Отриманий таким чином інструмент набуває робочої назви *PHP/FI* (укр. Персональна Домашня сторінка / Інтерпретатор Форм, англ. *Personal Home Page / Forms Interpreter*), надалі він також буде відомий під назвою *PHP 2*. Ця версія вже більшою мірою схожа на сьогоднішній *PHP*. Вона мала синтаксис і спосіб іменування змінних в стилі мови *Perl*, можливість вбудовування *PHP* операторів в *html*-код сторінки, автоматичну інтерпретацію форм, інтеграцію з базами даних. При цьому все працювало досить швидко, оскільки *PHP* прикомпілювалася до веб-сервера *Apache*. До 1997 року *PHP* використовувався вже на 50,000 доменах (не більше 1 % всіх веб-серверів).

Того ж 1997 року до проекту *PHP* підключилися Зев Сураскі (англ. *Zeev Suraski)* і Енді Гутманс (англ. Andi Gutmans). Ці студенти Техніону, одного з найкращих ізраїльських університетів, намагалися використовувати *PHP/FI*

для одного з комерційних університетських проектів. При цьому їм довелося зіткнутися з багатьма труднощами і обмеженнями цієї технології. Вивчаючи початковий код *PHP 2*, Зев і Енді дійшли висновку про необхідність доопрацювання, а точніше істотної переробки *PHP*, особливо в плані синтаксису мови. Протягом декількох місяців вони блискуче впоралися з цим завданням.

Закінчивши роботу, Зев і Енді домовились з Расмусом про співпрацю в галузі розвитку та вдосконалення мови. З цієї миті з'являється *PHP Group* — група однодумців, що працюють над розвитком технології *PHP*. Одержаний продукт з'явився на світ 1998 року під назвою *PHP 3*.

При цьому головною особливість *PHP 3* була можливість розширення ядра, що привернуло до роботи над *PHP* безліч сторонніх розробників, що створюють спеціалізовані модулі. Їх наявність дала *PHP* можливість працювати з величезною кількістю баз даних, протоколів, підтримувати велике число *API*. До кінця 1998 року кількість користувачів *PHP* перевищила 100 тисяч, а *PHP* був уже встановлений на понад 10 % серверах Інтернету. Водночас значному поширенню даної мови сприяли публікації в електронній пресі та видання посібників із *PHP*.

Відразу ж після виходу *PHP 3*, Енді Гутманс і Зев Сураскі почали переробку ядра *PHP*. В першу чергу належало розв'язати проблему підвищення продуктивності. Новий продукт, названий *Zend Engine* (від імен творців: *Zeev i Andi*), успішно справлявся з поставленим завданням і був реалізований 1999 року. Основними реалізованими ідеями є можливість компіляції сценарію у виконуваний модуль, за рахунок чого продуктивність можна було підняти на порядок.

Зеєв Сураскі та Ані Гутманс переробили аналізатор в 1997 році і сформували базу *PHP 3*, змінивши назву мови на рекурсивний акронім *PHP: Hypertext Preprocessor*. Після цього почалося публічне тестування PHP 3, а офіційний запуск відбувся в червні 1998 року. Після цього Сураскі та Гутманс

розпочали нове перекодування ядра *PHP*, видавши *Zend Engine* у 1999 році. Вони також заснували *Zend Technologies* в Рамат-Гані, Ізраїль.

22 травня 2000 р. Був випущений *PHP 4* з підтримкою *Zend Engine 1.0*. Станом на серпень 2008 року ця філія досягла версії 4.4.9. *PHP 4* більше не розробляється, оновлення безпеки також більше не були випущені.

14 липня 2004 р. Був випущений *PHP 5* з новим двигуном *Zend Engine II*. *PHP 5* включав нові функції, такі як покращена підтримка об'єктно-орієнтованого програмування, розширення *PHP Data Objects (PDO)* (який містить легкий і послідовний інтерфейс для доступу до баз даних) та численні поліпшення продуктивності. У 2008 році *PHP 5* став єдиною стабільною версією, що розроблялася. Пізній статичний зв'язок відсутній у *PHP* і був доданий у версії 5.3.

Багато високопрофесійних проектів з відкритим кодом з 5 лютого 2008 року перестали підтримувати *PHP 4* в новому коді, оскільки це ініціатива *GoPHP5*, що надається консорціумом розробників *PHP*, що сприяє переходу від *PHP 4* до *PHP 5*.

З часом перекладачі *PHP* стали доступними для більшості існуючих 32-розрядних та 64-розрядних операційних систем, або будували їх з вихідного коду *PHP*, або використовували попередньо побудовані двонарні файли. Для версій *PHP* версії 5.3 та 5.4 єдиними доступними двосторонніми дистрибутивами *Microsoft Windows* були 32-розрядні версії x86, що вимагають 32-розрядний режим сумісності *Windows* при використанні інформаційних служб Інтернету (*IIS*) на 64-розрядній платформі *Windows* . *PHP* версії 5.5 зробив збірки 64-розрядних x86-64 доступними для *Microsoft Windows*.

PHP отримав змішані відгуки через відсутність власної підтримки *Unicode* на рівні основної мови. У 2005 році був започаткований проект, очолюваний Андрієм Змієвським, для залучення рідної підтримки *Unicode* на *PHP*, шляхом вбудовування бібліотеки «Міжнародні компоненти для

Unicode» (*ICU*) та вбудованих текстових рядків як *UTF-16*. Оскільки це призведе до серйозних змін як до внутрішньої частини мови, так і до коду користувача, планувалося випустити його як версію 6.0 мови разом з іншими основними функціями, які розвиваються.

Проте дефіцит розробників, які зрозуміли необхідні зміни та проблеми продуктивності, що виникають внаслідок перетворення на *UTF-16* та з нього, що рідко використовується в веб-контексті, призвело до затримок проекту. Як результат, випуск *PHP 5.3* був створений у 2009 році, при цьому багато не-*Unicode*-функцій було відновлено з *PHP 6*, зокрема простору імен. У березні 2010 року проект у своїй нинішній формі був офіційно відкинутий, і був підготовлений випуск *PHP 5.4*, що містить більшість опублікованих функцій, що не входять до *Unicode*, з *PHP 6*, такі як риси та переприв'язка до закриття. Початкові сподівання полягали в тому, що для інтеграції з *Unicode* був би сформований новий план, але з 2014 року ніхто не був прийнятий.

Протягом 2014 та 2015 років було розроблено нову основну версію *PHP*, яку було пронумеровано *PHP 7*. Нумерація цієї версії викликала дебати. Хоча реліз *PHP 6 з Unicode* ніколи не був випущений, декілька назв статей і книг посилалися на назву *PHP 6*, що могло викликати плутанину, якби новий реліз повторно використовував цю назву. Після голосування було обрано назву *PHP 7*.

PHP 4, що працює на цьому ядрі, вийшов 2000 року. На додаток до збільшення продуктивності, *PHP 4* мав нові можливості щодо підтримки сесій, буферизацію виводу, безпечні способи обробки інформації, що вводиться користувачем, і нові мовні конструкції. З виходом 4 версії *PHP* став використовуватися вже на більш ніж 20 % доменів Інтернету.

Протягом 2000—2004 років продовжувалися активні роботи з покращення четвертої версії, але майже відразу *PHP Group* приступила до продумування можливостей нової версії. В першу чергу було вирішено підсилити об'єктні можливості мови, що дозволяло використовувати його для

реалізації масштабних проектів. Роботи із створення п'ятої версії велися тривалий час, в них брало участь рекордна кількість фахівців, зокрема Стерлінг Хьюз (англ. *Sterling Hughes*) і Маркус Бергера (англ. *Marcus Boerger*).

У липні 2004 року виходить офіційний реліз *PHP 5*. В першу чергу, як і планувалося, було перероблено весь механізм роботи з об'єктами. І якщо в попередніх версіях об'єктно-орієнтоване програмування на PHP було можливе в мінімальному ступені, а тому і використовувалося на практиці не часто, то *PHP 5* володіє прекрасним потенціалом реалізації об'єктного програмування. Окрім цього, *PHP* збагатився рядом цінних розширень для роботи з *XML*, різними джерелами даних, генерації графіки і інше.

Серед інших украй корисних доповнень в *PHP 5* слід зазначити нову схему обробки винятків. Конструкція *try/catch/throw* дозволяє весь код обробки помилок локалізувати в одному місці сценарію.

Всі основні бібліотеки для роботи з *XML*, запозичені в *PHP 4*, були піддані серйозній переробці. Такі популярні розширення, як *SAX, DOM і XSLT*, тепер використовують інструмент *libxml2*, що робить їх ще ефективнішими.

У *PHP 5* також включені два нові модулі для роботи з протоколами — *SimpleXML і SOAP*. *SimpleXML* дозволяє значно спростити роботу з *XML*-даними, представляючи вміст *XML*-документа у вигляді *PHP*-об'єкта. Розширення *SOAP* дозволяє будувати на *PHP* сценарії, що обмінюються інформацією з іншими застосунками за допомогою *XML*-повідомлень поверх існуючих веб-протоколів, наприклад *HTTP*. Модуль для роботи з *SOAP* для *PHP 5* надає розробникам засіб для достатньо швидкого створення ефективних *SOAP*-клієнтів і *SOAP*-серверів.

Новий модуль *PHP 5 MySQLi (MySQL Improved)* призначений для роботи з *MySQL*-сервером версій 4.1.2 і вище, реалізовуючи не тільки

процедурний, але і об'єктно-орієнтований інтерфейс до *MySQL*. Додаткові можливості цього модуля включають — SSL, контроль транзакцій, підтримка реплікації та інші.

Шоста версія *PHP* розроблялася з жовтня 2006 року. Було зроблено безліч нововведень, як, наприклад, виключення з ядра регулярних виразів POSIX і «довгих» суперглобальних масивів, видалення директив *safe_mode, magic_quotes_gpc і register_globals* з конфігураційного файлу *php.ini*. Одним з основних нововведень повинна була стати підтримка Юнікоду. Однак у березні 2010 року розробка *PHP6* була визнана безперспективною через складнощі з підтримкою Юнікоду. Вихідний код *PHP6* переміщений на гілку, а основною лінією розробки стала версія 5.4.

У 2014 році було проведено голосування, за результатами якого наступна версія отримала назву *PHP 7*. Вихід нової версії планувався в середині жовтня 2015 року. У березні 2015 року *Zend* представили інфографіку в якій описані основні нововведення *PHP 7*.

3 грудня 2015 року було оголошено про вихід *PHP* версії 7.0.0.

Нова версія ґрунтується на експериментальній гілці *PHP*, яка спочатку називалася *phpng* (*PHP Next Generation* — наступне покоління), і розроблялася з нахилом на збільшення продуктивності і зменшення споживання пам'яті. У новій версії додана можливість вказувати тип повертаються з функції даних, доданий контроль переданих типів для скалярних даних, а також нові оператори.

Недоліки *PHP*:

- Незручність дизайну мови
  - Змінні з символом *$*
  - Складні назви поширених функцій
  - Не підтримується *Unicode* у версіях до 6.0
- Непередбачуваність нових версій *PHP*.

3.2. Вимоги до технічного забезпечення

- тип комп’ютера : будь-який;
- монітор: розширення 10+" ;
- жорсткий диск: 30+ГБ;
- оперативна пам'ять: 1+Гб;
- тип ЦП: Intel Core 3+, AMD FX;
- відеокарта: будь-яка;
- відеопам’ять: 32+ Мб;
- клавіатура: стандартна;
- маніпулятор «миша»: стандартна;

3.3. Архітектура програмного забезпечення

3.3.1. Діаграма класів

Діаграма класів - статична структурна діаграма, що описує структуру системи, вона демонструє класи системи, їх атрибути, методи і залежності між класами.[6]

Існують різні точки зору на побудову діаграм класів в залежності від цілей їх застосування:

– концептуальна точка зору - діаграма класів описує модель предметної області, в ній присутні лише класи прикладних об'єктів;

– точка зору специфікації;

– точка зору реалізації - діаграма класів містить класи, що використовуються безпосередньо в програмному коді.

Діаграма компонентів - статична структурна діаграма, показує розбиття програмної системи на структурні компоненти та зв'язку між компонентами. Як фізичних компонент можуть виступати файли, бібліотеки, модулі, виконувані файли, пакети і т. п.

При створенні програми були розроблені класи, які відносяться до програмного забезпечення

4. ТЕХНОЛОГІЧНИЙ РОЗДІЛ

4.1. Керівництво користувача

В керівництві користувача описано найбільш загальні сценарії взаємодії користувача з програмою *«SOSтачка»*.

Для вивчення виклику евакуатора за допомогою програми необхідно виконати таку послідовність дій:

- відкривається головна форма
- на панелі меню обираймо вкладку «*Контакти*»;
- натиснення клавіші миші;
- відбувається перенесення до вкладки «*Контакти*»;
- натиснути на кнопку;
- відкривається вікно де користувач вводить *e-mail*;
- натиснути на кнопку кнопки «*Відправити*»;
- далі при натисканні кнопки на закритті вікна;

Вікна, які використовуються в цьому завданні, зображено на рисунку 4.1. Певні елементи інтерфейсу виділені для відображення ключових позицій.

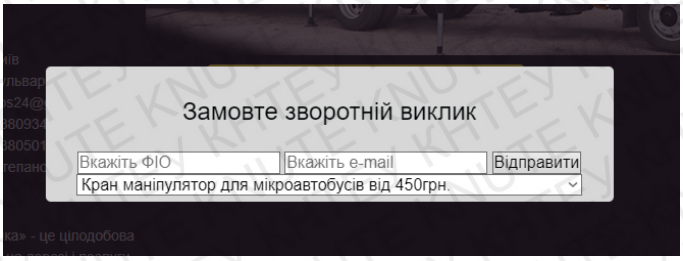

Рис. 4.1. Вікна, що використовуються виклику евакуатора

4.1.1. Призначення та запуск програми

Призначенням програми є допомога водіям під час виходу машини з ладу або ДТП, а саме виклику евакуатора. Для цього вони можуть використовувати програму *Web*застосування *«SOSтачка!»*, яка забезпечує зручний інтуїтивний інтерфейс та багатофункціональність.

Основними перевагами у використанні даного ПЗ є зручний інтерфейс, спокійні відтінки і кольори, та просте користування.

Для запуску програми необхідно завантажити його у файлі формату *apk* на комп'ютер з операційною системою *Windows* та встановити стандартним процесом інсталяції комп'ютерного додатку

4.1.2. Умови виконання програми

Щоб скористатись комп'ютерним додатком, необхідно мати:

– ноутбук або стаціонарний комп'ютер;

– операційну систему *Windows;*

– версію операційної системи не нижче 7,32 *bit.*

4.1.3. Повідомлення користувачу

При не введенні *e-mail*, форма залишається порожньою і не виконується виклик евакуатору, порожня форма, що залишається незмінна у користувача зображена на рисунку 4.2.

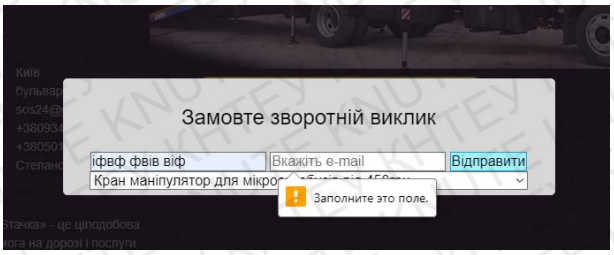

Рис. 4.2. Повідомлення користувачу

4.2. Керівництво програміста

Для створення проекту програмного забезпечення були вирішені такі задачі:

– розробка інтерфейсу програми;

– забезпечення базового функціоналу елементів інтерфейсу;

– кодування основної частини за побудованою схемою архітектури класів;

Структура проекту з переліком основних файлів програми вказана на рисунку 4.3.

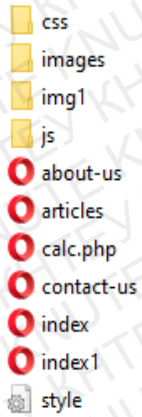

Рис 4.3. Файли програми

При розробці додатку було створено 15 файлів для створення інтерфейсу програми.

4.3. Текст програми

Проект, створений в *Sublime Text*, має велику кількість файлів. В ході дипломного проектування була здійснена робота з файлами для створення структури кожного вікна програми, де були вказані властивості різних елементів, полів, кнопок тощо.

До кожного вікна автоматично створюється програмний код, де було додатково опрацьовано події при натисненні та роботі з елементами вікон.

Текст основного коду програми представлений у Додатку А.

4.4. Тестування та аналіз результатів роботи

Для створення ПЗ було використано середовище розробки *Sublime Text*. [11]

При тестуванні ПЗ було виконано перегляд інтерфейсу та перевірка правильної роботи за логікою програми.

Тестування інтерфейсу програми:

– відсутність можливості зміни вікон користувачем;

– можливість переходу між вікнами програми швидко;

– доступність усіх кнопок;

– інтуїтивність інтерфейсу.

Інтерфейс програми є зручним у використанні, зрозумілим для користувача, а також оснащений сучасним дизайном.

4.5. Висновки до розділу

Під час роботи над розділом було продемонстровано весь основний функціонал програми.

Написано керівництво користувача, де була вказана необхідна інформація для правильної роботи з програмою, описано можливі варіанти дій, переходи між вікнами, елементи інтерфейсу. Вказано керівництво програміста з описом здійснених етапів розробки програмного продукту.

Представлено типи повідомлень, що з’являються у процесі роботи, за допомогою яких легко орієнтуватися у правильності виконаних дій. Розглянуто, яким чином проводилось тестування даного ресурсу.

ВИСНОВКИ

В ході дипломного проектування було проаналізовано предметну область, що стосується замовлення та надання послуг евакуаторного обладнання, та розглянуто можливості та методи автоматизації негайного або запланованого виклику евакуатора необхідного типу та вантажопідйомності, за наявністю доступу до мережі *Internet*. Також було зроблено огляд перспектив розвитку прикладних програм *Windows* та мови програмування *HTML, CSS*.

При створенні додатку було пройдено всі етапи розробки програмного забезпечення: описане технічне завдання; побудована архітектура додатку створено дизайн інтерфейсної частини додатку; розроблено програмні коди та описано процес роботи користувача з додатком.

Продемонстровано весь основний функціонал створеної програми. Представлено типи повідомлень, що з’являються у процесі функціонування програми. Проведено тестування та проаналізовано його результати.

Підсумком виконаного дипломного проекту на тему «Розробка WEB-застосування з підтримки діяльності служби евакуації на автомобільних шляхах.» є готовий програмний продукт, що може використовуватись як на ПК, так і на будь якому сучасному мобільному пристрої, що має будь який браузер та підключення до мережі *Internet.*

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Квинт И.: HTML, XHTML и CSS на 100%. - СПб.: Питер, 2010;
2. Седерхольм Дэн: CSS ручной работы. - СПб.: Питер, 2011;
3. Прохоренок Н. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / Н. Прохоренок. – СПб.: БХВ-Петербург, 2010. 900с.
4. Колисниченко Д.Н. PHP 5/6 и MySQL 6. Разработка web приложений / Д.Н. Колисниченко. – СПб.:БХВ-Петербург, 2010. 560с.
5. Советов Б. Я. Архитектура информационных систем / Б. Я. Советов, А. И. Водяхо, В. А. Дубенецкий, В. В. Цехановский. – М.: Академия, 2012. 228с.
6. Ковалюк Т.В. Основи програмування: Підручник/ Т.В.Ковалюк. - К.: Видавнича група BHV, 2005. -384 с. -ISBN 966-552-138-1;
7. Прохоренок Н. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / Н. Прохоренок. – СПб.: БХВ-Петербург, 2010. 900с;
8. Основи PHP [Електронний ресурс]http://kohanaframework.org;
9. Сучасний підручник JavaScript [Електронний ресурс] http://www.web-learn.ru;
10. Основи розробки сайтів [Електронний ресурс] http://www.tinymce.com;
11. Вільна енциклопедія [Електронний ресурс] https://uk.wikipedia.org/wiki/HTML;
12. Основи роботи з HTML [Електронний ресурс] http://htmlbook.ru/html.