**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА**

**ФАКУЛЬТЕТ УПРАВЛІННЯ ФІНАНСАМИ ТА БІЗНЕСУ**

**Кафедра цифрової економіки та бізнес-аналітики**

**КУРСОВА РОБОТА**

**з навчальної дисципліни ,,** **Проектування та адміністрування БД і СД”**

на тему:

**“Інформаційна система залізничних маршрутів”**

**Галузь знань:** 05 «Соціальні та поведінкові науки»

**Спеціальність:** 051 «Економіка»

**Спеціалізація:** «Інформаційні технології в бізнесі»

**Освітній ступінь:**  бакалавр

|  |  |  |  |
| --- | --- | --- | --- |
| **Науковий керівник:**  к.ф.-м.н., доц. Депутат Б.Я.\_\_\_\_\_\_\_\_\_.  (науковий ступінь, посада, прізвище, ініціали)  \_\_\_\_\_\_\_\_\_\_\_\_\_“\_\_\_” травня 2021 р.  (підпис) |  |  | **Виконавець:**  студент(ка) групи УФЕ-31 с  Федоранич Д.В.  \_\_\_\_\_\_\_\_\_\_\_\_ “\_\_\_”травня 2021 р.  (підпис) |

**Загальна кількість балів** \_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис, ПІП членів комісії)

**ЛЬВІВ 2021**

ЗМІСТ

[ВСТУП 3](#_Toc75761822)

[РОЗДІЛ 1. ТЕХНОЛОГІЇ РОЗРОБКИ САЙТУ 4](#_Toc75761823)

[1.1. HTML 4](#_Toc75761824)

[РОЗДІЛ 2. РОЗРОБКА БАЗИ ДАНИХ 18](#_Toc75761825)

[РОЗДІЛ 3. СТВОРЕННЯ САЙТУ 27](#_Toc75761826)

[3.1.Структура сайту 27](#_Toc75761827)

[3.2.Шапка сайту 27](#_Toc75761828)

[3.3. Сторінка з контентом 28](#_Toc75761829)

[ВИСНОВКИ 30](#_Toc75761830)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 31](#_Toc75761831)

# ВСТУП

Актуальність теми дослідження. В даний час Інтернет стає все більш розвиненим середовищем для здійснення комунікацій. У зв'язку з глобальним розвитком мережі Інтернет, в програмуванні все більш різко почала виділятися окрема його галузь web-програмування.

Зараз, щоб привернути увагу клієнтів, покупців або партнерів, просто необхідно заявити про себе в інтернеті, шляхом створення WEB-сайту. Для цих цілей якраз і служить web-сайт, що містить основну інформацію про організацію, приватну особу, компанію, товари або послуги, прайс-листи, контактні дані. Сайти дозволяють зберігати, передавати, продавати різні типи інформації, не відходячи від екрану компьютера. Wide Web - глобальна комп'ютерна мережа, на сьогоднішній день містить мільйони сайтів, на яких розміщена будь-яка інформація. Люди отримують доступ до цієї інформації за допомогою використання технологій Internet. Для пошуку по інтернету використовують спеціальні програми - Web-браузери, які суттєво полегшують подорож по безкрайніх просторах інтернету.

Розробка сайту та управління ним на сьогоднішній день для більшості організацій стає нагальною потребою. На сьогодні, сайт виступає як засіб комунікації, залучення цільової аудиторії.

Інтернет в наші дні - це ціла індустрія, швидко проникає в усі сфери людської діяльності. Величезна кількість компаній у всьому світі бачать в Internet великий комерційний потенціал і можливість перекладу свого бізнесу на якісно новий рівень. Завдяки постійному вдосконаленню технологій, швидкість доступу росте, а ціна стає все більш прийнятною. Internet знаходить своє застосування в багатьох сферах, не обійшовши стороною і діяльність комерційних фірм.

Реклама в друкованих виданнях нездатна повною мірою висвітлити діяльність організації, інформація в рекламних буклетах і брошурах швидко втрачає актуальність. У цьому ключі розробка сайту відкриває світ нових можливостей і способів досягнення бізнес цілей за допомогою неймовірно великого арсеналу рекламних засобів і стратегій. Розробити сайт означає показати свою сучасність і актуальність. Розробити сайт - для організації означає «бути на рівні», бути конкурентоспроможним, адже в будь-якій справі застосування інноваційних технологій впливає на образ і оцінку організації. Саме тому, розробка сайту сьогодні є досить актуальною і затребуваною послугою.

Метою курсової роботи є створення інформаційної системи руху громадського транспорту. Дана тема актуальна, тому, що дуже багато людей користуються потягами, автобусами, літаками. І створення такого сайту посприяє кращій орієнтації з маршрутами та цінами на той вид транспорту який буде вибирати відвідувач сайту.

Для досягнення мети в роботі поставлено й вирішено такі теоретичні та практичні завдання:

- розробка сайту з використанням сучасних програмних засобів;

- розробка дизайну оформлення, що робить сайт більш привабливим для клієнтів;

- створення зручного інтерфейсу для можливості комфортного перебування користувачів на сайті;

- адаптивність для будь-якого гаджета.

Об’єктом дослідження є інформаційна система руху громадського транспорту.

Предмет дослідження – HTML5 , CSS3 , BOOTSTRAP5.01 , Angular10, Json-Server , Firebase , TypeScript , JavaScript , SCSS , MySQL.

Практичне значення отриманих результатів. Програма може використовуватись фізичними особами, які хочуть вибрати для себе маршрут яким вони будуть рухатись.

Для створення сайту використовувалось середовище розробки Visual Studio Code та Angular.

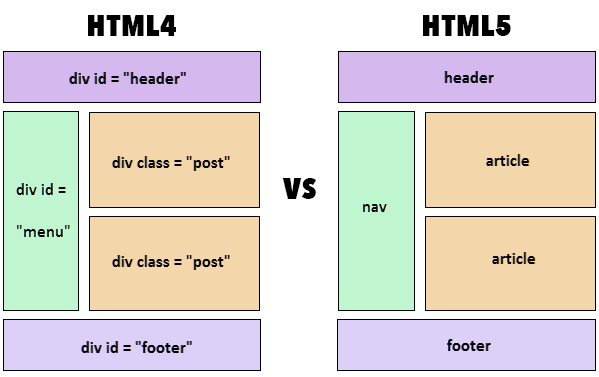
Курсова робота складається з трьох розділів («ТЕХНОЛОГІЇ РОЗРОБКИ САЙТУ», «СТВОРЕННЯ САЙТУ» ,та «РОЗРОБКА БАЗИ ДАНИХ»), висновків та списку використаних джерел. Загальний обсяг роботи – 37 сторінок.

# РОЗДІЛ 1. ТЕХНОЛОГІЇ РОЗРОБКИ САЙТУ

## HTML

Всі напевно хоч щось чули про HTML5 - нова версія мова розмітки сторінок в інтернеті. Вона до цих пір знаходиться в розробці, хоча з часу офіційного затвердження діючої версії HTML4 пройшло більше п'ятнадцяти років - фактично весь час активного розвитку Всесвітньої мережі. У чому ж полягають переваги HTML5 і принципові його відмінності від HTML4?

На відміну від розробленого ще в дев'яностих мови розмітки HTML4, коли навіть відносно нескладний мультимедійний контент був занадто «важким» для переважної більшості користувачів інтернету, в 2004 році, в якому фактично почалася робота над новою версією стандарту, в Мережі вже щосили йшло поширення аудіо і відеозаписів, як офіційне, так і неофіційне.



*Рис1.1. Новий стандарт*

Хорошим тоном для веб-сайтів стало розміщення на своїх сторінках анімаційних фільмів та відеороликів, увійшли в моду аудіоблоги.

Однак чинний стандарт не передбачав опису мультимедійного контенту засобами самої мови: для цього використовувалася функція вставки якогось об'єкта ( «object»), можливість відтворення якого залежала від того, чи встановлені в браузері клієнта сумісні з таким контентом додаткові плагіни третіх розробників. Говорячи про подібні плагинах, ми перш за все маємо на увазі Adobe Flash, QuickTime або менш поширені Real Player і Silverlight. Все це - «надбудови», доповнення до браузерів, які не є їх складовою частиною і виконують роль якихось посередників, які перетворять завантажується цифровий контент у відео і звук. Ця система досі цілком працездатна, однак відсутність загальноприйнятих стандартів призводить до того, що на різних сайтах використовуються різні технології, для перегляду їх медіаконтенту потрібні різні плагіни і, найголовніше, оскільки такі плагіни не належать безпосередньо в складі браузера, необхідна їх окрема установка .

У стандарті HTML5, що представляє собою поєднання власне HTML, а також CSS і JavaScript, спочатку реалізується принципово інший підхід, в якому медіаконтент є невід'ємною частиною коду сторінки. HTML5 забезпечує єдиний стандартний спосіб опису (синтаксис) вбудованого в сторінки аудіовізуального контенту.

Для інтеграції аудіо або відео в HTML5 досить використовувати простий набір тегів, що описують, який тип контенту ви маєте намір помістити ( «video» або «audio»), і звичайні посилання на відповідні мультимедійні дані.

Зрозуміло, у HTML5 є і безліч інших нововведень, але це найголовніше з них - зміна ідеологічного підходу.

* 1. **CSS**

CSS - це формальна мова, службовець для опису оформлення зовнішнього вигляду документа, створеного з використанням мови розмітки (HTML, XHTML, XML). Назва походить від англійського Cascading Style Sheets, що означає «каскадні таблиці стилів».

Призначення CSS - відокремлювати те, що задає зовнішній вигляд сторінки, від її змісту. Якщо документ створено тільки з використанням HTML, то в ньому визначається не тільки кожен елемент, але і спосіб його відображення (колір, шрифт, положення блоку і т. Д.). Якщо ж підключені каскадні таблиці стилів, то HTML описує тільки черговість об'єктів. А за все їх властивості відповідає CSS. В HTML досить прописувати клас, не перераховуючи всі стилі кожен раз.

Технологія:

- забезпечує відносно просту і швидку розробку, тому що одного разу створене оформлення можна застосовувати до багатьох сторінок;

- підвищує гнучкість і зручність редагування - досить внести правку в CSS, щоб оформлення змінилося всюди;

- робить код більш простим, знижуючи повторюваність елементів. Його простіше читати програмістам і пошуковим роботам;

- прискорює час завантаження, тому що CSS може кешуватися при першому відкритті, а в наступних зчитуються тільки структура і дані;

- збільшує кількість візуальних рішень для подання вмісту;

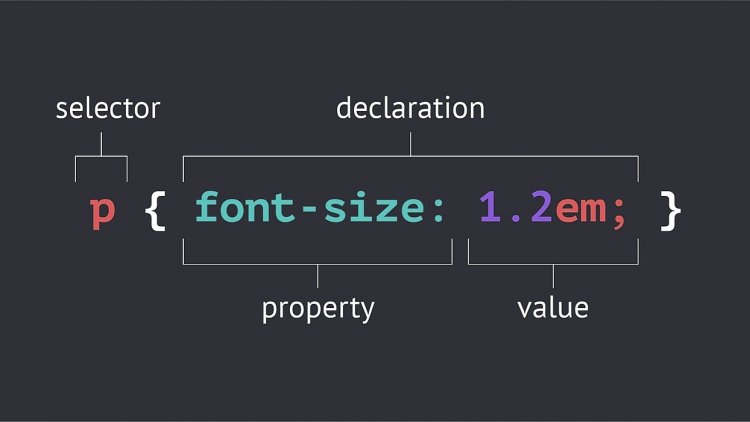
- забезпечує можливість легко застосовувати до одного документу різні стилі (наприклад, створювати адаптовану версію для мобільних пристроїв або спеціальні стилі для людей з вадами зору).

Тобто каскадні таблиці служать не тільки для втілення дизайну, а й кардинально змінюють підхід до сайтобудування, спрощуючи працю розробників і забезпечуючи гнучкість реалізації. Ось для чого потрібен CSS.

Необхідність розробки CSS було визнано консорціумом W3C в 1990-х роках. У 1996 році був прийнятий стандарт CSS1, що дозволяє змінювати параметри шрифту, колір, атрибути тексту, вирівнювання і відступи. У 1998 році відбувся вихід CSS2, який додав можливості використання блокової верстки, звукових таблиць, що генерується змісту, покажчиків, сторінкових носіїв. Версія CSS3 помітно збільшила можливості стилів: стало доступним створення анімованих елементів без використання JavaScript, з'явилася підтримка згладжування, тіней, градієнтів і т. Д. Специфікація була розділена на модулі, кожен з яких став розвиватися окремо. З 2011 року ведеться розробка модулів CSS4. Можливості поки описані в чорнових варіантах.

CSS можна охарактеризувати простими словами як набір правил, що описують, як повинен виглядати елемент.

Правило складається з селектора і блоку оголошень.



*Рис 2.1. Застосування стилів на тег*

Селектор повідомляє, до якого елементу будуть застосовані описувані в CSS властивості стилю. Як селектора може виступати будь-який тег, якому задається форматування (розмір, колір і т. Д.). Якщо для тега потрібно задати різні стилі або застосувати один для відмінних елементів, використовуються класи і запис виду «Тег.Класс {властивість: значення;}». Ім'я класу задається латиницею, може містити підкреслення або дефіс. Якщо не вказувати тег, а починати запис з «.Класс», то можна використовувати правило для будь-якого тега. Якщо перераховувати кілька класів для одного тега, до нього застосовується все описані стилі. Ідентифікатор задає унікальне ім'я елемента для зміни стилю або звернення за допомогою скрипта.

Запис «# Ідентифікатор {властивість: значення;}». Назва ідентифікатора складається з букв латинського алфавіту, допустимо використовувати дефіс і підкреслення. Щоб застосувати ідентифікатор до конкретного тегу, вказується його ім'я, потім без пробілу і через знак решітки назва ідентифікатора.

Блок оголошень складається з пар «властивість: значення» (запис завжди черга двокрапка), розміщених в фігурних дужках. Записи закінчуються крапкою з комою. CSS нечутливий до табуляції, прогалин, регістру. Вибір способу запису (стовпчиком з відступом або просто в рядок) залишається на розсуд розробника. Якщо для одного селектора прописані різні значення для одного властивості, то пріоритет віддається нижньої записи.

Підключення CSS.CSS можна пов'язати з HTML декількома способами:

* Всередині тега с помощью атрибута style. При цьом немає спожи вказуваті селектор;
* додати тег <style> з атрибутом type = "text / css";
* підключити зовнішню таблицю стилів: <link rel = "stylesheet" href = "шлях до style.css" type = "text / css" />.

Третій спосіб є найбільш популярним і рекомендованим, тому що дозволяє в повній мірі користуватися перевагами поділу форми і змісту, що забезпечується за допомогою CSS.

* 1. **JavaScript**

JavaScript - один з найпопулярніших мов програмування в світі з більш ніж двадцятирічної історією. Це також один з трьох головних мов програмування для веб-розробників.

JavaScript можна швидко вивчити і легко використовувати в самих різних варіантах застосування, від простих поліпшень функціональності сайту до запуску ігор і веб-додатків. Або, що ще краще, існує тисячі шаблонів JavaScript і доступних додатків завдяки безкоштовним сайтам на кшталт Github.

JavaScript був створений за 10 днів автором Бренданом Айком (Brandan Eich), які працювали на Netscape в далекому 1995 році. Спочатку він був названий Mocha, ім'я мови було змінено на Mona і потім на LiveScript поки нарешті він не став JavaScript. Первісна версія мови була обмежена тільки браузером Netscape і пропонувала вузьку функціональність, але з часом він продовжив розвиватися частково завдяки спільноті розробників, які не залишали роботу над ним. У 1996 році мова JavaScript був стандартизований і було дано офіційне ім'я ECMAScript, з ECMAScript 2 вийшли в 1998 і ECMAScript 3 в наступному 1999 році. Це перетворилося в сьогоднішній JavaScript, які тепер працює не тільки в різних браузерах, але також на різних пристроях, включаючи мобільні та настільні комп'ютери. JavaScript продовжив зростання з тих пір до показників 92% сайтів, які використовують його в 2016 році. Всього за двадцять років він перетворився з примітивного мови програмування в один з найбільш популярних інструментів в арсеналі веб-розробника. Якщо ви використовуєте інтернет, то ви неодмінно стикалися з JavaScript.

JavaScript це мова з великими перевагами, які роблять його найкращим вибором серед подібних йому, особливо в деяких варіантах застосування. Всього кілька переваг використання JavaScript:

- Вам не потрібен компілятор, тому що веб-браузер інтерпретує його в HTML;

- Його простіше вивчати, ніж інші мови програмування;

- Помилки простіше виявити, а значить і справити;

- Він може прив'язуватися до спеціальних елементів сторінок або події на зразок натискання (click) або наведення миші (mouseover);

- JS працює в різних браузерах і на різних платформах;

- Ви можете використовувати JavaScript для валідації вхідних даних і зниження необхідності ручної перевірки даних;

- Він робить сайт більш інтерактивним і привабливим для відвідувачів;

- Він швидше і легше, ніж інші мови програмування

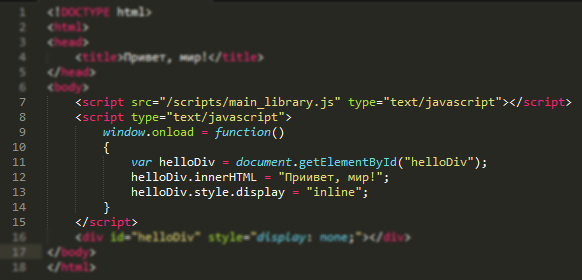
У кожної мови програмування є свої недоліки і слабкі місця. Однією і причин виникнення проблем є популярність мови. Коли мова програмування стає таким популярним як JavaScript, він стає об'єктом підвищеного інтересу для хакерів, шахраїв та інших шкідливих проявів третіх сторін, які намагаються знайти уразливості і слабкі місця в безпеці. Деякі слабкі місця:

* Вразливий відношенню до експлойтів (шкідливий код, який використовує уразливості програмного продукту);
* Може бути використаний для запуску шкідливого коду на комп'ютері користувача;
* Не завжди підтримується деякими браузерами або пристроями;
* Фрагменти JS коду можуть бути дуже великими;
* Може по різному відображатися на різних пристроях, що призводить до відсутності цілісності.

Код JavaScript зазвичай або вбудований в вашу сторінку, або посилається на окремий файл .js. Це мова, яка працює на стороні клієнта, що означає, що скрипт закачується на комп'ютер відвідувача і обробляється там, на противагу тому, як працюють мови боку сервера (виконуються на вашому сервері до того, як відправити файл відвідувачам).

Слід враховувати, що більшість браузерів також пропонують можливість користувачам відключати JavaScript. Тому слід продумати, що відбудеться в разі, якщо на якомусь комп'ютері код не буде інтерпретуватися.

Додати рядок коду JavaScript на сайт можна використовуючи тег <script>. Ви можете використовувати наступний приклад для наочності того, як це виглядає на практиці:



*Рис 1.3. Зразок підключення JS*

Як основне правило, вставка вашого коду JavaScript всередині тегів <header>, якщо тільки вам не потрібно його виконання в певний час або на певних елементах сторінки. Ви також можете зберегти свій код JavaScript в окремому файлі і викликати його там, де знадобиться на сайті.

Після ґрунтовного вивчення JavaScript можна удосконалювати свої навички практично в будь-якому напрямку. Рекомендується освоїти фреймворки і бібліотеки, що дозволить користуватися ефективними наборами функціональних класів. У перелік найпоширеніших технологій цієї групи входять jQuery, Angular і React. Також варто звернути увагу на надбудови TypeScript, Dart і CoffeeScript. Залежно від обраної технології, вони допоможуть зробити програмний код компактніше, суворіше або чистіше.

* 1. **Angular**

Angular (зазвичай так називають фреймворк Angular 2 або Angular 2+, тобто вищі версії) — написаний на TypeScript front-end фреймворк з відкритим кодом, який розробляється під керівництвом Angular Team у компанії Google, а також спільнотою приватних розробників та корпорацій. Angular — це AngularJS, який переосмислили та який був повністю переписаний тією ж командою розробників.

Спочатку переписаний AngularJS отримав назву Angular 2 від команди розробників, яка над ним працювала, але це призвело до плутанини серед інших розробників. Аби пояснити різницю між ними та наголосити, що це окремі проекти, команда вирішила для фреймворків версій 1.X застосовувати назву AngularJS, а для версій, починаючи з 2.0, — Angular без JS.

Як згадано вище, Angular — це ретельно переписаний AngularJS.

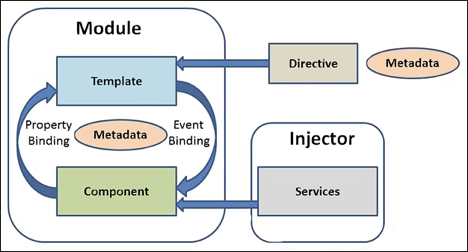
- Додано Angular CLI, що дає змогу розпочати створення нового додатка, просто написавши команду ng new [app name]

- Angular не використовує концепцію "області видимості" або контролерів, натомість як головну архітектурну концепцію він застосовує ієрархію компонентів

- Angular має інакший синтаксис написання виразів, застосовуючи "[ ]" для біндінгу даних властивостей, і "( )" для біндінгу даних івентів

- Модульність – значна частина основного функціоналу перенесена у модулі

Реліз версії Angular 8 запланований на березень-квітень 2019 року, а версії Angular 9 орієнтовно на вересень-жовтень 2019 року. Очікується, що кожна наступна версія буде зворотно сумісною з попередньою. Google також обіцяє публікувати оновлення двічі на рік.



*Рис 1.4. Загальний огляд архітектури Angular*

Перш за все, варто зазначити, що Angular застосунки пишуться на TypeScript, а не на чистому JavaScript. Версія синтаксису для JavaScript не отримала широкого розповсюдження, тому на даний момент у документації усі синтаксичні конструкції описані з використанням синтаксису TypeScript.

Коротко оглянемо основні пункти. Архітектура Angular складається з:

* Module;
* Component;
* Template;
* Service;
* Router;
* Pipe;
* Directives.

Модулі (Module) — структурні одиниці застосунку, які інкапсулюють певну логіку. В Angular це структури, які зберігають певні компоненти, директиви та сервіси, об’єднані певною логікою. Прикладом може слугувати профіль користувача, модуль для написання листа, огляд списку листів тощо.

Компоненти (Component) — typescript клас, який зберігає дані та логіку відображення цих даних у шаблоні (представленні). Шаблон тісно пов’язаний з компонентом. Дані з компонента можна з легкістю відображати у шаблоні, використовуючи спеціальний синтаксис. Компонент також може «знімати» дані з шаблона та отримувати їх безпосередньо у скрипті.

Шаблон (Template) — фрагмент html-коду з додаванням спеціального синтаксису. Він дозволяє впроваджувати в шаблон дані з компонента без використання innerHTML та подібних методів. Шаблон прописується у компоненті та є частиною його конфігурації.

Сервіс (Service) в Angular являє собою typescript класи, які виконують задачі, пов’язані з отриманням, зберіганням та обробкою даних. Наприклад, логування, перетворення даних для подальшої передачі у компонент, звернення до backend та ін. На відміну від компонентів та директив сервіси не працюють з представленнями (шаблонами) напряму.

Задачі сервісів:

* Надання даних застосунку. Сервіс сам може зберігати дані у пам’яті або, з метою отримання даних, звертатися до якогось джерела даних, наприклад, до сервера;
* Сервіс може організувати канал взаємодії між окремими компонентами застосунку;
* Сервіс може інкапсулювати бізнес-логіку, різноманітні обчислювальні задачі, задачі з логування, які краще виносити поза компоненти. Таким чином, код компонентів буде зосереджений, безпосередньо, на роботі з представленням. До того ж, можемо розв’язати проблему повторення коду, якщо нам знадобиться виконати одну й ту саму задачу у різних компонентах і класах.

Роутер (Router)— маршрутизатор, який призначений для переходу між екранами з метою відображення різного контенту.

Іншими словами, коли в адресному рядку браузера у вас змінюється фрагмент URL, маршрутизатор відстежує ці зміни та завантажує ту або іншу частину застосунку.

Завершимо теоретичну частину та перейдемо до написання коду.

Директиви та Пайпи — більш специфічні конструкції, які простіше продемонструвати у коді, ніж описати словами.

Ознайомитись повністю з архітектурою Angular у рамках однієї статті неможливо. Аби дізнатися більше, ви можете переглянути повну версію вебінару у нас на сторінці чи зареєструватися на повний курс «Angular трансформація».

Структурна директива — функціональна частина Angular, що дозволяє змінювати DOM-структуру за певними правилами, а саме: додавати чи видаляти елементи. Повністю розібратися з синтаксисом директив у межах однієї статті досить складно. Якщо вам необхідно вивчити як працює цей механізм Angular, ви можете досягти цього, приєднавшись до повного курсу «Angular трансформація». Наразі дамо коротке визначення.

\*ngFor працює на зразок циклу. Всередині прописано масив links з компоненту AppComponent. На кожній ітерації циклу ngFor ми почергово присвоюємо змінній topLink наступне значення з масиву links. Таким чином, відбувається ітерація масивом links одразу в шаблоні. Фактично, у відображеному фрагменті HTML коду побачимо стільки елементів <button>, скільки в нас елементів масиву links.

Всередині кнопки використовується конструкція {{topLink.label}}. З кожною ітерацією цю конструкцію буде замінено на реальні значення з ключа label в об’єкті topLink.

В Angular існують допоміжні директиви routerLink та routerLinkActive, які дозволяють досить легко організувати навігацію маршрутами всередині застосунку. До директиви routerLink у якості параметру передається шлях до маршруту, який необхідно буде відкрити при кліку на відповідну кнопку чи посилання. У нашому випадку це «/main» або «/chat». RouterLinkActive використовується для того, щоб присвоїти певний css клас активному у даний момент маршруту. У нашому випадку клас'button-active' буде встановлено на кнопку, яка містить посилання на активний маршрут.

Іншою важливою частиною цього фрагменту є тег <router-outlet> </router-outlet>. Саме нижче нього буде підставлено HTML-код того компоненту, який активується на поточному маршруті. Алгоритм наступний: активується певний маршрут, який ініціалізує компонент, прописаний в конфігурації маршрутизатора. У свою чергу, компонент вставляє свій HTML-код нижче директиви <router-outlet> </router-outlet>. Таким чином, ця директива працює як маркер і вказує на місце, куди буде динамічно завантажуватись згенерований зміст.

Зверніть увагу на початок, де для path вказано пустий рядок. По суті реалізовується redirect механізм. Якщо ви перейдете на головну сторінку сайту, ви автоматично потрапите на маршрут main.

Можемо запустити проект та побачити, що вийшло. З цією метою переходимо в консоль та викликаємо команду ng serve. Якщо перейти назад до браузера та оновити сторінку застосунку, побачимо зміни. Тепер з’явилось дві кнопки: «Головна» та «Чат». При натисканні на будь-яку з цих кнопок на сторінці змінюється текст: «main works!» та «chat works!» відповідно. Також значення змінюється у рядку url.

Розглянемо метод getBotMessage. У ньому відбувається пошук відповіді. Якщо відповідь знайдено, вона повертається. В іншому випадку повертається стандартне повідомлення. По суті ми лише звертаємось до об’єкта за ключем та повертаємо знайдене значення.

Розглянемо метод getBotAnswer. У ньому відбувається наступне: створюємо новий екземпляр Messagе з даних, вказаних користувачем. Додаємо їх до потоку this.conversation, використовуючи метод next(). На потік ми підписані у компоненті. Таким чином, після виклику next() компонент одразу отримає дані та зможе відобразити їх у необхідній формі. У нашому випадку текст, який ввів користувач у текстове поле буде відображено в історії листування.

Потім створюємо екземпляр класу Message, який міститиме відповідь від чат-боту. Для отримання відповіді викликаємо методgetBotMessage().

Далі, через деякий час, додаємо об’єкт з відповіддю до потоку this.conversation. Відповідно, компонент, який підписаний на цей потік, зможе негайно відобразити відповідь від чат-бота у листуванні.

Для затримання відповіді від чат-бота використовуємо setTimeout. Затримка необхідна для створення у користувача відчуття, що він спілкується з людиною, а не автоматом.

Директива ngClass дозволяє встановити певний css клас в залежності від автора повідомлення — бота чи користувача.

Значення з input-поля, завдяки директиві [(ngModel)], автоматично переноситься до поля класу value. Цей механізм працює і у зворотному напрямку. Якщо призначити полю класу value значення, воно автоматично відобразиться у шаблоні в полі вводу. Такий механізм має назву двобічне зв’язування (two way data binding).

З кліком на кнопку «Відправити», а також при натисканні на Enter у полі вводу, буде викликано метод sendMessage(), який викличе метод сервісу та передасть до нього рядок, введений користувачем.

Більш детально робота з шаблоном розглядалася на вебінарі «Як створити свій Angular застосунок за дві години», який можна знайти за посиланням.

* 1. **TypeScript**

TypeScript — мова програмування, представлена Microsoft восени 2012; позиціонується як засіб розробки веб-застосунків, що розширює можливості JavaScript.

Розробником мови TypeScript є Андерс Гейлсберг (англ. Anders Hejlsberg), який створив раніше C#, Turbo Pascal і Delphi.Код експериментального компілятора, котрий транслює код TypeScript в представлення JavaScript, поширюється під ліцензією Apache, розробка ведеться в публічному репозиторії через сервіс CodePlex. Специфікації мови відкриті і опубліковані в рамках угоди Open Web Foundation Specification Agreement (OWFa 1.0).

TypeScript є зворотньо сумісним з JavaScript. Фактично, після компіляції програму на TypeScript можна виконувати в будь-якому сучасному браузері або використовувати спільно з серверною платформою Node.js.

Переваги над JavaScript:

* можливість явного визначення типів (статична типізація),
* підтримка використання повноцінних класів (як в традиційних об'єктно-орієнтованих мовах),
* підтримка підключення модулів.

За задумом ці нововведення мають підвищити швидкість розробки, прочитність, рефакторинг і повторне використання коду, здійснювати пошук помилок на етапі розробки та компіляції, а також швидкодію програм.

Планується, що в силу повної зворотної сумісності адаптація наявних застосунків на нову мову програмування може відбуватися поетапно, шляхом поступового визначення типів. Підтримка динамічної типізації зберігається — компілятор TypeScript успішно обробить і не модифікований код на JavaScript.

Основний принцип мови — будь-який код на JavaScript сумісний з TypeScript, тобто в програмах на TypeScript можна використовувати стандартні JavaScript-бібліотеки і раніше створені напрацювання. Більш того, можна залишити наявні JavaScript-проекти в незмінному вигляді, а дані про типізації розмістити у вигляді анотацій, які можна помістити в окремі файли, які не заважатимуть розробці і прямому використанню проекту (наприклад, подібний підхід зручний при розробці JavaScript-бібліотек).

На момент релізу представлені файли для сприйняття розширеного синтаксису TypeScript для Vim і Emacs, а також плагін для Microsoft Visual Studio.

Одночасно з виходом специфікації розробники підготували файли з деклараціями статичних типів для деяких популярних JavaScript-бібліотек, серед яких jQuery.

# РОЗДІЛ 2. РОЗРОБКА БАЗИ ДАНИХ

**2.1.MySql – база даних**

Шведська компанія MySQL AB спочатку розробила MySQL в 1994 році. Тоді американська технологічна компанія Sun Microsystems повністю придбала право власності, купивши MySQL AB в 2008 році. Американський технологічний гігант Oracle в 2010 році придбав Sun Microsystems, а з тих пір MySQL практично належала Oracle.

Що стосується загального визначення, MySQL це система керування базами даних з відкритим вихідним кодом (СУРБД) з моделлю клієнт-сервер. СУРБД - це програмне забезпечення або служба, яка використовується для створення та управління базами даних на основі реляційної моделі.

База даних - це просто набір структурованих даних. Наприклад, коли ви робите Селфі: ви натискаєте кнопку і фотографуєте себе. Ваша фотографія - це дані, а галерея вашого телефону - це база даних. База даних - це місце, в якому зберігаються дані. Слово «реляційний» означає, що дані, що зберігаються в наборі даних, організовані у вигляді таблиць. Кожна таблиця пов'язана в деякому роді. Якщо програмне забезпечення не підтримує реляційну модель даних, просто назвіть її СУБД.

Відкритий вихідний код означає, що ви можете вільно використовувати і змінювати його. Будь-хто може встановити програмне забезпечення. Ви також можете вивчити і налаштувати вихідний код, щоб він краще відповідав вашим потребам. Однак GPL (GPU Public License) визначає, що саме ви можете зробити в залежності від умов. Комерційна ліцензована версія доступна, якщо вам потрібно більш гнучке володіння і розширена підтримка.

Комп'ютери, які встановлюють і запускають програмне забезпечення СУРБД, називаються клієнтами. Коли їм потрібно отримати доступ до даних, вони підключаються до сервера СУРБД. Це система «клієнт-сервер».

MySQL є одним з багатьох варіантів програмного забезпечення СУРБД. Вважається, що СУРБД і MySQL однакові через популярність MySQL. Назвіть кілька великих веб-додатків, таких як Facebook, Twitter, YouTube, Google і Yahoo! всі використовують MySQL для зберігання даних. Хоча спочатку він створювався для обмеженого використання, тепер він сумісний з багатьма важливими обчислювальними платформами, такими як Linux, macOS, Microsoft Windows і Ubuntu.

MySQL і SQL не збігаються. Пам'ятайте, що MySQL є однією з найпопулярніших торгових марок програмного забезпечення СУРБД, яка реалізує модель клієнт-сервер. Отже, як клієнт і сервер взаємодіють в середовищі СУРБД? Вони використовують специфічний для домену мову - мову структурованих запитів (SQL). Якщо ви коли-небудь стикалися з іншими іменами, в яких є SQL, такими як PostgreSQL і сервер Microsoft SQL, вони, швидше за все, є брендами, які також використовують синтаксис SQL. Програмне забезпечення СУРБД часто пишеться на інших мовах програмування, але завжди використовує SQL в якості основного мови для взаємодії з базою даних. Сам MySQL написаний на C і C ++.

Інженер-комп'ютерник Тед Кодд розробив SQL на початку 1970-х років на основі реляційної моделі IBM. Він став більш широко використовуватися в 1974 році і швидко замінив аналогічні, тоді застарілі мови, ISAM і VISAM. Крім історії, SQL повідомляє серверу, що робити з даними. Це схоже на ваш пароль або код WordPress. Ви вводите його в систему, щоб отримати доступ до області панелі управління.

MySQL дійсно не єдина СУРБД на ринку, але вона є однією з найпопулярніших і поступається тільки Oracle Database, коли оцінюється з використанням таких важливих параметрів, як кількість згадок в результатах пошуку, професійних профілів в LinkedIn і частоти технічних дискусій на інтернет-форумах. Той факт, що багато великих технологічні гіганти покладаються на нього, ще більше зміцнює заслужену позицію.

Захист даних забезпечується двома підсистемами: таблицею привілеїв і плагінами безпеки.

Перша потрібна, щоб захистити частину даних від певної групи користувачів. Вона змушує клієнтів, які роблять запити, авторизуватися в системі, щоб та могла переконатися в праві клієнта на отримання запитуваної інформації. Тобто управляти базою в повній мірі може обмежене число осіб. За бажанням можна продовжити певному колу осіб можливість вносити в таблиці будь-які зміни або видаляти з них дані, але при цьому залишити за ними можливість додавати нові одиниці даних в існуючу базу.

Найближчий приклад такої системи - права доступу в WordPress і DataLife Engine.Модулі безпеки розширюють базові механізми захисту цілісності даних. Наприклад, створюють на сервері більш сувору політику створення паролів або додаткове сховище для конфіденційної інформації.

**2.2.Реалізація MySQL скрипту**

Створюєм таблиці та їхні змінні

USE Travel

CREATE TABLE CITY (ID INT NOT NULL, NAME VARCHAR(30))

CREATE TABLE CUSTOMER (ID INT NOT NULL, ID\_CITY INT, FIRSTNAME VARCHAR(30), LASTNAME VARCHAR(30), GENGER VARCHAR(1),

DATE\_BIRTH DATE, PHONE\_NUMBERS INT, EMAIL VARCHAR(30), DISCOUNT INT);

Створюєм продавців,замовників та продукт та прописуєм їм інформацію

CREATE TABLE PRODUCT (ID INT NOT NULL, PRODUCT\_NAME VARCHAR(30), PRODUCT\_DESCRIPTION VARCHAR(200),

CATEGORY VARCHAR(15), PRODUCT\_WEIGHT INT, PRODUCT\_PRICE INT);

CREATE TABLE ORDERS (ID INT NOT NULL, PRODUCT\_ID INT, CUSTOMER\_ID INT, TRANSACTION\_TYPE

VARCHAR(15), DATE\_ORDER DATE, AMOUNT INT);

ALTER TABLE PRODUCT ADD CONSTRAINT PRODUCT\_ID PRIMARY KEY (ID);

ALTER TABLE ORDERS ADD FOREIGN KEY (PRODUCT\_ID) REFERENCES PRODUCT(ID);

ALTER TABLE CUSTOMER ADD CONSTRAINT CUSTOMER\_ID PRIMARY KEY(ID);

ALTER TABLE ORDERS ADD FOREIGN KEY(CUSTOMER\_ID) REFERENCES CUSTOMER(ID);

ALTER TABLE CITY ADD CONSTRAINT ID\_CITY PRIMARY KEY (ID);

ALTER TABLE CUSTOMER ADD FOREIGN KEY (ID\_CITY) REFERENCES CITY(ID);

INSERT INTO CITY(ID, NAME) VALUES (1, 'Lviv')

INSERT INTO CITY(ID, NAME) VALUES (2, 'Kiyv')

INSERT INTO CITY(ID, NAME) VALUES (3, 'Dnipro')

INSERT INTO CITY(ID, NAME) VALUES (4, 'Chernigiv')

INSERT INTO CITY(ID, NAME) VALUES (5, 'Donetsk')

INSERT INTO CITY(ID, NAME) VALUES (6, 'Kharkiv')

INSERT INTO CITY(ID, NAME) VALUES (7, 'Ternopil')

INSERT INTO CITY(ID, NAME) VALUES (8, 'Portava')

INSERT INTO CITY(ID, NAME) VALUES (9, 'Chernivtsi')

INSERT INTO CITY(ID, NAME) VALUES (10, 'Batyatichy')

SELECT \* FROM CITY

INSERT INTO CUSTOMER(ID, ID\_CITY, FIRSTNAME, LASTNAME, GENGER, DATE\_BIRTH, PHONE\_NUMBERS, EMAIL, DISCOUNT)

VALUES (1,10, 'Oleg', 'Barylko', 'M', '10/10/2001', '11111', 'olegbarylko@gmail.com',5)

INSERT INTO CUSTOMER(ID, ID\_CITY, FIRSTNAME, LASTNAME, GENGER, DATE\_BIRTH, PHONE\_NUMBERS, EMAIL, DISCOUNT)

VALUES (2,1, 'Yaroslav', 'Ryvko', 'M', '10/10/2001', '22222', 'yaroslav@gmail.com',3)

INSERT INTO CUSTOMER(ID, ID\_CITY, FIRSTNAME, LASTNAME, GENGER, DATE\_BIRTH, PHONE\_NUMBERS, EMAIL, DISCOUNT)

VALUES (3,2, 'Ostap', 'Felypets', 'M', '10/05/2000', '33333', 'ostap@gmail.com',10)

INSERT INTO CUSTOMER(ID, ID\_CITY, FIRSTNAME, LASTNAME, GENGER, DATE\_BIRTH, PHONE\_NUMBERS, EMAIL, DISCOUNT)

VALUES (4,3, 'Denys', 'Fedoranych', 'M', '10/11/2001', '44444', 'denys@gmail.com',4)

INSERT INTO CUSTOMER(ID, ID\_CITY, FIRSTNAME, LASTNAME, GENGER, DATE\_BIRTH, PHONE\_NUMBERS, EMAIL, DISCOUNT)

VALUES (5,4, 'Bondar', 'Volodimir', 'M', '12/11/1991', '55555', 'volo@gmail.com', 15)

INSERT INTO CUSTOMER(ID, ID\_CITY, FIRSTNAME, LASTNAME, GENGER, DATE\_BIRTH, PHONE\_NUMBERS, EMAIL, DISCOUNT)

VALUES (6,5, 'Yarish', 'Taras', 'M', '08/05/1993', '66666', 'taras@gmail.com', 7)

INSERT INTO CUSTOMER(ID, ID\_CITY, FIRSTNAME, LASTNAME, GENGER, DATE\_BIRTH, PHONE\_NUMBERS, EMAIL, DISCOUNT)

VALUES (7,6, 'Yevgen', 'Osoba', 'M', '01/01/2001', '55555', 'yevgen@gmail.com',10)

INSERT INTO CUSTOMER(ID, ID\_CITY, FIRSTNAME, LASTNAME, GENGER, DATE\_BIRTH, PHONE\_NUMBERS, EMAIL, DISCOUNT)

VALUES (8,7, 'Didus', 'Nazar', 'M', '02/07/2000', '77777', 'nazar@gmail.com',5)

INSERT INTO CUSTOMER(ID, ID\_CITY, FIRSTNAME, LASTNAME, GENGER, DATE\_BIRTH, PHONE\_NUMBERS, EMAIL, DISCOUNT)

VALUES (9,8, 'Kuzik', 'Iruna', 'W', '04/07/2003', '99999', 'irusik@gmail.com',9)

INSERT INTO CUSTOMER(ID, ID\_CITY, FIRSTNAME, LASTNAME, GENGER, DATE\_BIRTH, PHONE\_NUMBERS, EMAIL, DISCOUNT)

VALUES (10,9, 'Alina', 'Maksumiv', 'W', '01/04/2001', '000000', 'alina@gmail.com',7)

SELECT \* FROM CUSTOMER

INSERT INTO PRODUCT (ID, PRODUCT\_NAME, PRODUCT\_DESCRIPTION,CATEGORY,PRODUCT\_WEIGHT,PRODUCT\_PRICE)

VALUES(1,'lviv - Kyiv', '3 hours', 'tran', 400, 340)

INSERT INTO PRODUCT (ID, PRODUCT\_NAME, PRODUCT\_DESCRIPTION,CATEGORY,PRODUCT\_WEIGHT,PRODUCT\_PRICE)

VALUES(2,'Khust - lviv', '10 hours', 'train', 250 ,200)

INSERT INTO PRODUCT (ID, PRODUCT\_NAME, PRODUCT\_DESCRIPTION,CATEGORY,PRODUCT\_WEIGHT,PRODUCT\_PRICE)

VALUES(3,'Mukachevo - Lviv', '4 hours', 'train', 330 ,330)

INSERT INTO PRODUCT

(ID, PRODUCT\_NAME, PRODUCT\_DESCRIPTION,CATEGORY,PRODUCT\_WEIGHT,PRODUCT\_PRICE)

VALUES(4,'Kyiv - Kharkiv', '6 hours' , 'train', 200 ,350)

INSERT INTO PRODUCT (ID, PRODUCT\_NAME, PRODUCT\_DESCRIPTION,CATEGORY,PRODUCT\_WEIGHT,PRODUCT\_PRICE)

VALUES(5,'Kyiv - Kremenchuk', '5 hours' , 'train', 150 ,250)

INSERT INTO PRODUCT (ID, PRODUCT\_NAME, PRODUCT\_DESCRIPTION,CATEGORY,PRODUCT\_WEIGHT,PRODUCT\_PRICE)

VALUES(6,'Kyiv - Mariupol', '8 hours', 'train', 310 ,210)

INSERT INTO PRODUCT (ID, PRODUCT\_NAME, PRODUCT\_DESCRIPTION,CATEGORY,PRODUCT\_WEIGHT,PRODUCT\_PRICE)

VALUES(7,'Uzhorod - Chop', '1 hour', 100 ,50)

INSERT INTO PRODUCT (ID, PRODUCT\_NAME, PRODUCT\_DESCRIPTION,CATEGORY,PRODUCT\_WEIGHT,PRODUCT\_PRICE)

VALUES(8,'Chop - Lviv', '7 hours' , 'train', 0.5 ,150)

INSERT INTO PRODUCT (ID, PRODUCT\_NAME, PRODUCT\_DESCRIPTION,CATEGORY,PRODUCT\_WEIGHT,PRODUCT\_PRICE)

VALUES(9,'Chop - Kyiv', '14 hours' , 'train', 1.0 ,250)

INSERT INTO PRODUCT (ID, PRODUCT\_NAME, PRODUCT\_DESCRIPTION,CATEGORY,PRODUCT\_WEIGHT,PRODUCT\_PRICE)

VALUES(10,'Stryi - Lviv', '2 hours', 'train', 180 ,60)

INSERT INTO PRODUCT (ID, PRODUCT\_NAME, PRODUCT\_DESCRIPTION,CATEGORY,PRODUCT\_WEIGHT,PRODUCT\_PRICE)

VALUES(11,'Uzhorod - Lysuchansk', '15 hours', 'train', 280 ,650)

SELECT \* FROM PRODUCT

INSERT INTO ORDERS (ID,PRODUCT\_ID, CUSTOMER\_ID,TRANSACTION\_TYPE,DATE\_ORDER,AMOUNT)

VALUES(1,1,1,'CASH','2021-05-13',3)

INSERT INTO ORDERS (ID,PRODUCT\_ID, CUSTOMER\_ID,TRANSACTION\_TYPE,DATE\_ORDER,AMOUNT)

VALUES(2,2,2,'CASH','2021-05-13',4)

INSERT INTO ORDERS (ID,PRODUCT\_ID, CUSTOMER\_ID,TRANSACTION\_TYPE,DATE\_ORDER,AMOUNT)

VALUES(3,3,3,'CASH','2021-05-04',5)

INSERT INTO ORDERS (ID,PRODUCT\_ID, CUSTOMER\_ID,TRANSACTION\_TYPE,DATE\_ORDER,AMOUNT)

VALUES(4,4,4,'CREDIT','2021-05-09',3)

INSERT INTO ORDERS (ID,PRODUCT\_ID, CUSTOMER\_ID,TRANSACTION\_TYPE,DATE\_ORDER,AMOUNT)

VALUES(5,5,5,'CASH','2021-05-10',2)

INSERT INTO ORDERS (ID,PRODUCT\_ID, CUSTOMER\_ID,TRANSACTION\_TYPE,DATE\_ORDER,AMOUNT)

VALUES(6,6,6,'CREDIT','2021-05-14',4)

INSERT INTO ORDERS (ID,PRODUCT\_ID, CUSTOMER\_ID,TRANSACTION\_TYPE,DATE\_ORDER,AMOUNT)

VALUES(7,7,7,'CREDIT','2021-05-21',5)

INSERT INTO ORDERS (ID,PRODUCT\_ID, CUSTOMER\_ID,TRANSACTION\_TYPE,DATE\_ORDER,AMOUNT)

VALUES(8,8,8,'CREDIT','2021-05-25',2)

INSERT INTO ORDERS (ID,PRODUCT\_ID, CUSTOMER\_ID,TRANSACTION\_TYPE,DATE\_ORDER,AMOUNT)

VALUES(9,9,9,'CREDIT','2021-05-03',1)

INSERT INTO ORDERS (ID,PRODUCT\_ID, CUSTOMER\_ID,TRANSACTION\_TYPE,DATE\_ORDER,AMOUNT)

VALUES(10,10,10,'CREDIT','2021-05-04',4)

INSERT INTO ORDERS (ID,PRODUCT\_ID, CUSTOMER\_ID,TRANSACTION\_TYPE,DATE\_ORDER,AMOUNT)

VALUES(11,11,3,'CREDIT','2021-05-01',2)

SELECT \* FROM ORDERS

SELECT PRODUCT\_NAME, CATEGORY, PRODUCT\_PRICE

FROM PRODUCT

WHERE PRODUCT\_PRICE = (SELECT MAX(PRODUCT\_PRICE)

FROM PRODUCT)

SELECT PRODUCT\_NAME, CATEGORY, PRODUCT\_PRICE

FROM PRODUCT

WHERE PRODUCT\_PRICE = (SELECT MIN(PRODUCT\_PRICE)

FROM PRODUCT)

SELECT PRODUCT\_NAME, CATEGORY, PRODUCT\_PRICE

FROM PRODUCT

WHERE PRODUCT\_PRICE > (SELECT AVG(PRODUCT\_PRICE)

Записуєм команди пошуку

FROM PRODUCT)

SELECT \*

FROM CUSTOMER

WHERE ID\_CITY = 2

UNION

SELECT \*

FROM CUSTOMER

WHERE ID\_CITY = 3

SELECT ID, FIRSTNAME

FROM CUSTOMER

UNION

SELECT PRODUCT\_ID, TRANSACTION\_TYPE

FROM ORDERS

SELECT PRODUCT\_NAME, CATEGORY, PRODUCT\_PRICE

FROM PRODUCT

WHERE PRODUCT\_PRICE >= 200 AND PRODUCT\_PRICE <= 300

SELECT ID, FIRSTNAME, LASTNAME

FROM CUSTOMER

WHERE LASTNAME LIKE 'B%'

SELECT NAME FROM CITY

GROUP BY NAME

SELECT \* FROM CUSTOMER WHERE ID=9 OR GENGER='M'

SELECT ID\_CITY, COUNT(\*)

FROM CUSTOMER

GROUP BY ID\_CITY;

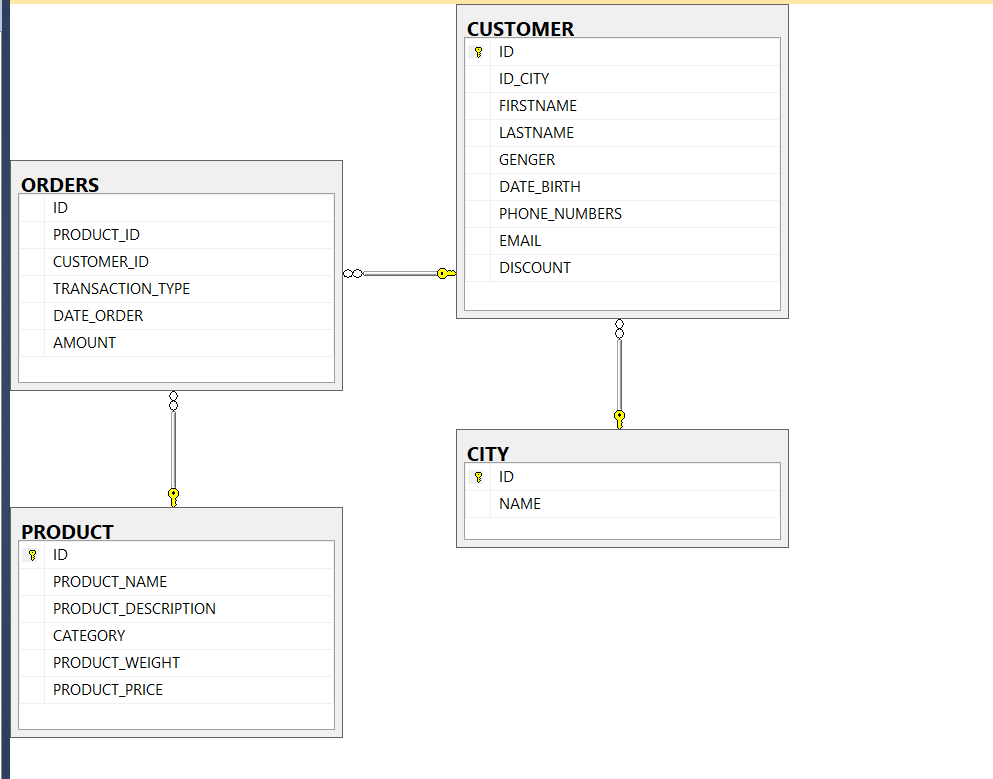
SELECT ID, FIRSTNAME, DISCOUNT

FROM CUSTOMER

WHERE DISCOUNT>7

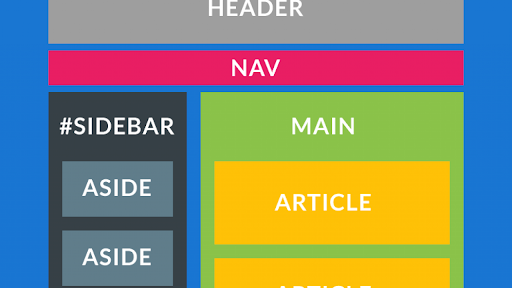
ORDER BY DISCOUNT

-Створення полів для таблиці Order



# РОЗДІЛ 3. СТВОРЕННЯ САЙТУ

## 3.1.Структура сайту



*Рис 3.1. Семантика сайту*

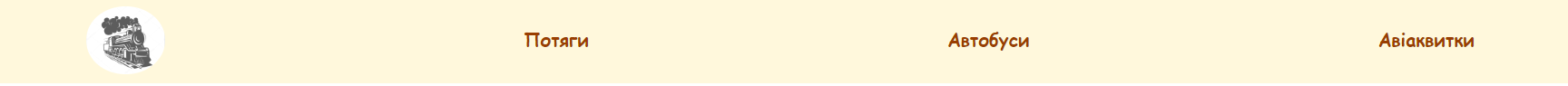
## 3.2.Шапка сайту

Структура сайту ділиться на зовнішню та внутрішню.

Основними елементами Зовнішньої структури є:

- **Шапка сайту**, яка містить логотип і назву сайту. Шапка сайту може вирішувати декілька завдань. По-перше, це візитна картка інтернет-ресурсу.

По-друге, шапка сайту дуже часто використовується для розміщення в ній основного меню.

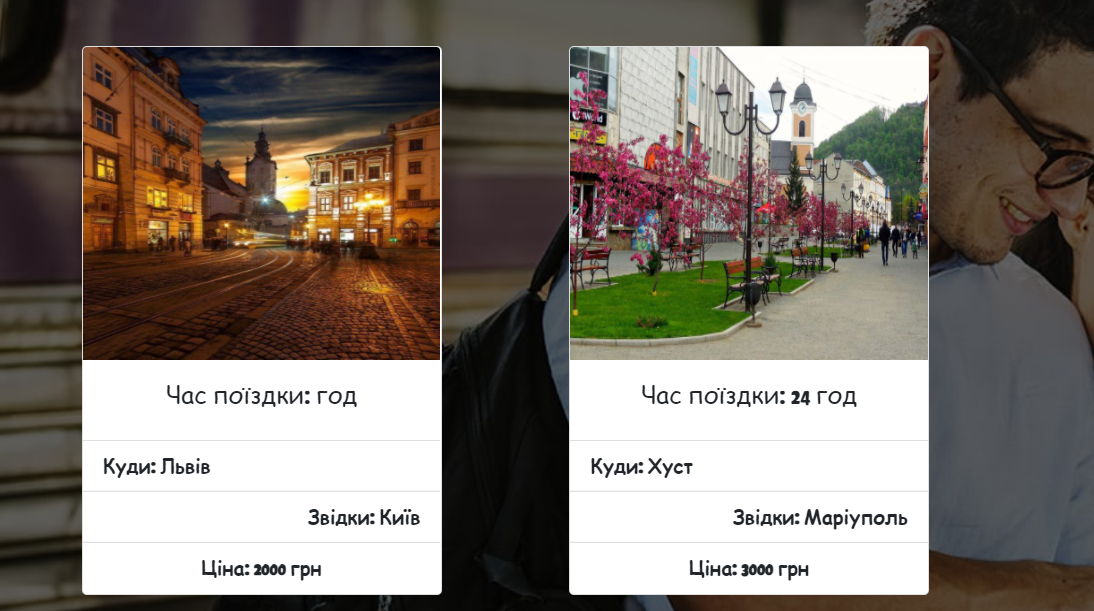


*Рис 3.2. Шапка сайту*

Хедер (header) - це елемент веб-сторінки, що знаходиться вище області контенту. Якщо описати веб-сторінку «архітектурними» термінами, то футер - це підвал сайту, область контенту - стіни і вікна, а хедер - дах. Або, як його називають в нашому середовищі - шапка сайту. Як не назви, header - стратегічна область, яку прискіпливо оцінює кожен користувач перш, ніж почати скролінг і вивчення сторінки. Шапка сайту показується в перші секунди взаємодії, тому вона стала справжнім випробувальним полігоном для психологів і маркетологів. Будучи своєрідним привітанням, цей елемент повинен доносити ключову інформацію про веб-проект, допомагати в навігації і одночасно робити гарне враження на підсвідомому рівні.

## 3.3. Сторінка з контентом

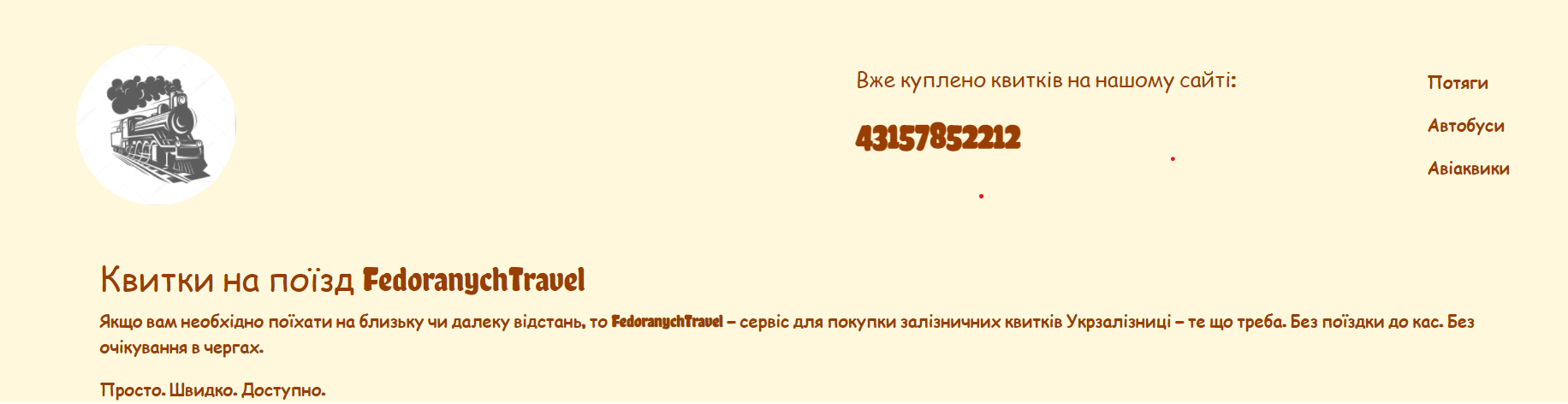
**Контент** (англ. content – зміст) – змістове наповнення веб-сторінки, доступне користувачу: тексти, зображення, відео, звукові дані та інше. Перед нами головна сторінка шкільного сайту, основним контентом якої є новини.



Також на цій сторінці відвідувач сайту може знайти для себе саме той маршрут який йому буде до вподоби

**3.4. Підвал сайту**

Підвал сайту або футер, є менш функціональною частиною. В ньому містяться контактні дані та повідомлення про авторські права. Окрім цього, на сайті можуть бути присутні посилання на інші сайти, різноманітні інформери) та інші елементи



**-Адміністратор** **веб-сайту** встановлює, налаштовує та керує інструментами та програмним забезпеченням, пов'язаними з роботою веб-сайтів. Він знає мови програмування, які використовуються при створенні веб-сайтів

Внутрішня структура сайту, як правило, представлена картою сайту – діаграмою, що візуально відображає ієрархію сторінок сайта, схему зв’язків і переходів між ними.

# ВИСНОВКИ

В курсовій роботі було проаналізовано, як правильно використовуват

и MySql у роботі з даними, а також розглянуто особливості WEB розробки.

При розробці даної роботи були виконані всі вимоги технічного завдання. Було перепроектовано структуру бази даних та реалізовано скрипти на мові програмування HTML для роботи з нею; було розроблено модуль для збереження, редагування та перегляду сторінок сайту.

При розробці даного порталу було використано мову програмування ANGULAR та сервер баз даних MySQL. В якості візуального редактора використаноHTML . Для розробки дизайну сайту та обробки графічних зображень використовувалися такі графічні редактори як CSS

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ботт Эд. Разработка веб-сайтов. – М., 2004.

2. Гукин Д. – FrontPage для “чайников”. – К.,1996.

3. Комисаров Д.А., Станкевич А.Г. Персональный учитель по персональному

robota.html4. JavaServerFaces[Електрон. ресурс]. – 2019. – Режим доступу:https://uk.wikipedia.org/wiki/JavaServer\_Faces

5. Мансфильд Рон. Делаем качественный сайт. – К., 1997

6. JSF мертв? [Електрон. ресурс]. – 2017. – Режим доступу:https://habr.com/ru/post/315894/

7. Простоеприложение JSF 2.0 использующеекомпонентыPrimeFaces[Електрон. ресурс]. – 2018. – Режим доступу:http://www.quizful.net/post/jsf2primefaces

8. Шаблон наблюдателя в Java[Електрон. ресурс].-2020.-Режим доступу:

https://www.codeflow.site/ru/article/java-observer-pattern

9. Спостерігач (шаблон проектування)[Електрон. ресурс].–2019. – Режим доступу:https://uk.wikipedia.org/wiki/%D0%A1%D0%BF%D0%BE%D1%81%D1%82%D0%B5%D1%80%D1%96%D0%B3%D0%B0%D1%87\_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD\_%D0%BF%D1%80%D0%BE%D1%94%D0%BA%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)

10. РазработкаWeb-приложений при помощиJavaServerFaces)[Електрон. ресурс].– 2017. – Режим доступу:<http://khpi-iip.mipk.kharkiv.edu/library/>extent/prog/jsf/jsf.html

11. Руководство по JSF. Введение. [Електрон. ресурс].–2015.– Режим доступу:https://proselyte.net/tutorials/jsf-tutorial/introduction/

13.Технології створення програмних продуктів та інформаційних систем : навч. посіб. / М. Ю. Карпенко, Н. О. Манакова, І. О. Гавриленко ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2017. – 93 с.