НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Кафедра автоматизації проектування енергетичних процесів і систем

**КУРСОВА РОБОТА**

з дисципліни « Основи веб-програмування - 2»

(назва дисципліни)

на тему: «Розробка інтернет-магазину на базі архітектурного шаблону MVC»

Студент (ки) групи ТВ-61

напряму підготовки ***6.050103 Програмна інженерія***

Калашникова-Травіна В. В.

(прізвище та ініціали)

Керівник: \_доцент, к.т.н. Карпенко Є. Ю.\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна оцінка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

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

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

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

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали

Київ- 2018 рік

**Національний технічний університет України**

**„Київський політехнічний інститут** **імені Ігоря Сікорського”**

**Кафедра автоматизації проектування енергетичних процесів і систем**

***КУРСОВА РОБОТА***

**з дисципліни: « Основи веб-програмування - 2 »**

**тема: «Розробка інтернет-магазину на базі архітектурного шаблону MVC»**

|  |  |
| --- | --- |
| **Керівник доцент, к.т.н. Карпенко Є. Ю.** | **Виконав**  **Калашников-Травін В. В.** |
| **Допущено до захисту** | **Студент II курсу** |
| **„\_\_\_”\_\_\_\_\_\_\_\_\_201 р.** | **Групи ТВ-61** |
| **Захищено з оцінкою** | **залікова книжка ТВ-6133** |
| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |  |

***2018***

**Національний технічний університет України**

**“Київський політехнічний інститут імені Ігоря Сікорського”**

Факультет ТЕПЛОЕНЕРГЕТИЧНИЙ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

( повна назва )

Кафедра автоматизації проектування енергетичних процесів та систем

( повна назва )

Напрям підготовки 6.050103 Програмна інженерія

# (шифр і назва)

## З А В Д А Н Н Я

### НА КУРСОВУ РОБОТУ СТУДЕНТУ

Калашникову-Травіну Владиславу Володимировичу

(прізвище, ім’я, по батькові)

1.Тема роботи

**Розробка інтернет-магазину на базі архітектурного шаблону MVC**

керівник курсової роботи

доцент, к.т.н. Карпенко Є. Ю.

( прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

2. Строк подання студентом роботи

**8 червня 2018 року**

3. Вихідні дані до проекту:

**мова розмітки HTML5, таблиці стилів SCSS (компільований в CSS3), мова запитів MySQL, мови програмування PHP, TypeScript (компільований в JavaScript ECMA5), середовище розробки JetBrains PhpStorm 2018.1.4, середовище роботи з базами даних JetBrains DataGrip 2018.1.4**

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

**розробка PHP MVC фреймворку, розробка бази даних, побудова веб-додатку на базі створеного фреймворку**

#### КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів виконання дипломного  проекту (роботи) | Строк виконання етапів проекту  ( роботи ) | Примітка |
| 1. | Затвердження теми проекту | 27.04.2018 |  |
| 2. | Вивчення та аналіз задачі | 5.05.2018 |  |
| 3. | Побудова фреймворку | 19.05.2018 |  |
| 4. | Проектування бази даних | 26.05.2018 |  |
| 5. | Реалізація серверного функціоналу | 1.06.2018 |  |
| 6. | Верстка веб-інтерфейсу | 3.06.2018 |  |
| 7. | Оформлення пояснювальної записки | 5.06.2018 |  |

**Студент Калашников-Травін В. В.**

( підпис ) (прізвище та ініціали)

**Керівник курсової роботи Карпенко Є. Ю.**

( підпис ) (прізвище та ініціали)

# ЗМІСТ

[ЗМІСТ 4](#_Toc516035277)

[ВСТУП 5](#_Toc516035278)

[1. АРХІТЕКТУРНИЙ ШАБЛОН MVC 6](#_Toc516035279)

[1.1 Загальні відомості 6](#_Toc516035280)

[1.2 Реалізація MVC у веб-додатку 8](#_Toc516035281)

[2. ПОБУДОВА ВЕБ-ДОДАТКУ «ІНТЕРНЕТ-МАГАЗИН» 9](#_Toc516035282)

[2.1 Обрання мови програмування 9](#_Toc516035283)

[2.2 Структура проекту 10](#_Toc516035284)

[2.3 Допоміжні класи 12](#_Toc516035285)

[2.3.1 Класи SQLBuilder та QueryBase 12](#_Toc516035286)

[2.3.2 Класи Router, RouteBase та похідні 13](#_Toc516035287)

[ВИСНОВКИ 14](#_Toc516035288)

[ЛІТЕРАТУРА 16](#_Toc516035289)

[ДОДАТОК 1. СКРІНШОТИ ВЕБ-САЙТУ 17](#_Toc516035290)

[ДОДАТОК 2. ДІАГРАМА КЛАСІВ ВЕБ-ДОДАТКУ 19](#_Toc516035291)

# ВСТУП

У 21 столітті мережа Інтернет стала невід’ємною частиною людського життя. Переважна кількість звичних речей, таких як купівля нового телевізора, оплата послуг, повсякденне спілкування, так чи інакше перенеслась у мережу. Технологія мережі Інтернет дозволяє спростити та пришвидшити рутинні людські справи та використовувати свій чай ефективно.

Інтернет-магазин (далі Магазин) – річ, звична, нині, для кожного з нас. Відтепер немає необхідності стояти в чергах та подорожувати по місту в покушах бажаного. Магазин дозволяє прямо з дому замовляти цікавлячі нас речі в декілька кліків.

Метою даної курсової роботи є створення Магазину речей для молоді. Клієнт Магазину матиме змогу переглядати наявні товари, шукати за фільтрами та замовляти їх, не виходячи із дому.

Інтерфейс магазину має бути доступним та зручним для всіх – від користувачів мобільних приладів і до повнорозмірних домашніх комп’ютерів. Саме тому, однією з вимог до результуючого сайту є адаптивність, тобто немає різниці, з якого приладу клієнт буде замовляти, верстка сайту буде автоматично підлаштовуватись під розміри клієнтського приладу.

Крім того, Магазин обов’язкого має десь зберігати дані – користувачі, товари, замовлення, доставка і так далі, тому окрім самого веб-додатка необхідно спроектувати базу даних, яка дозволить ефективно зберігати та обслуговувати клієнтів.

# АРХІТЕКТУРНИЙ ШАБЛОН MVC

## 1.1 Загальні відомості

Архітектурний шаблон – шаблон розробки програмного забезпечення, що являє собою вирішення архітектурних проблем розробки програмного забезпечення. Архітектурні шаблони виражають фундаментальну схему структурної організації певної програмної системи. Така схема складається із визначених наперед підсистем, а також точно визначає їхні сфери відповідальності та взаємовідносини.

Шаблон MVC (англ. Model-View-Controller – Модель-Вигляд-Контроллер) – один з найпопулярніших архітектурних шаблонів у світі; передбачає розділення системи додатку на три чітко розмежовані компоненти, кожна з яких відповідає строго за свою задачу. Основною ідеєю цього шаблону є розмежування користувацького інтерфейсу та робочої «моделі» додатку. Розділення додатку на частини спрощує подальший розвиток/розширення додатку, а також зменшує кількість залежностей у проекті за рахунок чіткого розподілення зобов’язаностей кожної з компонент.

Компонента View відповідає лише за відображення користувацького інтерфейсу. View є найменшою з даних трьох компонент та має найменшу кількість задач, а саме лише одну – вивід контенту користувачу. Насправді, View – це те, що бачить користувач в результаті роботи додатку. View є незалежною компонентою, тобто вона не знає про існування інших двох компонент та не може саморучно вплинути на їх роботу.

Компонента Model є «ядром» додатку MVC. Існування додатку зумовлюється поставленою перед ним задачею або цільовою метою. Всі процеси, що являють собою «центральні» процеси відбуваються в розпорядженні компоненти Model. Такі процеси також часто називають «бізнес-процесами». Model також є незалежною компонентою, тобто не залежить від View (не знає, як відображати дані/результати) та не залежить від Controller (не має прямих точок взаємодії з користувачем). Постає питання, як

саме має взаємодіяти користувач з додатком, тобто як користувач впливатиме «бізнес-процеси». Поєднуючою ланкою між цими компонентами є третя і остання компонента Controller.

Controller є невід’ємною частиною MVC-додатку, оскільки саме через нього відбувається обмін даними між користувачем і Model. В залежності від заданих користувачем умов, Controller обирає, що саме необхідно зробити Model, отримує результати і передає їх назад користувачеві через View. Також необхідно сказати, що Controller є також необхідним «бар’єром» для невалідних даних, тобто, якщо дані, задані користувачем, заздалегідь не є вірними, навіть початок роботи Model з ними вже не має ніякого сенсу. Наприклад, під час реєстрації на сайті, користувач випадково відправив пусту форму входу, тобто поля логін та пароль надійшли до додатку порожніми. Очевидно, що немає сенсу компоненті Model навіть починати працювати з цими даними, оскільки неможливо створити обліковий запис користувача з пустим полем логін. Розробники мають заздалегідь передбачати такі виключні ситуації та «віддавати» користувачу відповідну реакцію через View.

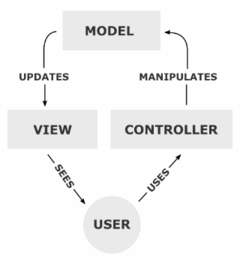


Рисунок 1.1 - Принцип взаємодії користувача з MVC-додатком; взаємодія компонент MVC одне з одним

## Реалізація MVC у веб-додатку

MVC у сфері веб розробки є дещо самобутнім поняттям. Web-MVC має багато Controllers та Models (мається на увазі класів), а також, частіше всього один клас View. Окрім того, Web-MVC відрізняється від традиційного MVC наявністю спеціального проміжного класу Router. Router має декілька задач, а саме:

розібрати користувацький URI (англ. - Uniform Resource Identifier)

знайти відповідні Controller, Model та Action

якщо вдалось знайти необхідні компоненти, виконати Action

віддати користувачеві готовий View

URI - компактний рядок літер, який однозначно ідентифікує окремий абстрактний чи фізичний ресурс. Таким чином, отримуючи URI від користувача, Router шукає серед доступних Controller та Action.

Action виступає лише певним методом, що належить певному класу Controller.

Наприклад, нехай маємо домен example.loc. Користувач хоче потрапити за посиланням example.loc/articles/view?id=4. В даному випадку маємо відповідний URI /articles/view?id=4. Дане посилання значить, що користувач хоче переглянути статтю з унікальним ідентифікатором, рівним 4.

Класичною ідеєю побудови класу Router є те, що першою частиною URI виступає назва Controller, другою частиною відповідно Action і третьою частиною є параметри запиту. На прикладі, це фактично означає, що буде створено екземпляр класу Articles та буде викликано в ньому метод view.

# ПОБУДОВА ВЕБ-ДОДАТКУ «ІНТЕРНЕТ-МАГАЗИН»

## 2.1 Обрання мови програмування

Для реалізації курсової роботи було обрано мову програмування PHP. PHP є однією з найпопулярніших мов програмування для побудови веб-сайтів. PHP (англ. PHP: Hypertext Preprocessor — PHP: гіпертекстовий препроцесор) – інтерпретована, мультипарадигмальна мова програмування, результатом якої є HTML-розмітка, що надсилається як відповідь користувачеві. PHP має влаштовані засоби роботи з користувацькими запитами, тому саме нею легко та швидко писати прості та середньої складності додатки. PHP – інтерпретована мова, це значить, що вона не компілюється в машинні коди для прямого її виконання процесором, її код перетворюється в спеціальний байт-код, що виконується віртуальною машиною. Це означає, що PHP – кросплатформенна мова, а отже PHP-сервер можливо запустити як на Linux, так і на Windows або Mac. PHP є об’єктно-орієнтованою, а також функціональною мовою програмування, що робить її схожою на сімейство С/С++ подібних мов.

## Структура проекту

Було прийнято рішення розбити проект на чотири частини:

* Framework
* Config
* Public
* Application

Framework – ядро додатку. Являє собою набір класів та методів, якими можна користуватися під час написання інших частин програми. Framework є багаторазовим, тобто написавши його один раз, ним можна користуватися безліч разів для різних проектів. Слід звернути увагу на те, що ядро залишається незмінним під час написання всього проекту. Framework включає в себе базові класи та інтерфейси Controller, Model, засоби роботи з базою даних, класи Router, Route та похідні від нього, допоміжні класи та бібліотеки, а також класи виключень для обробки виключних ситуацій, що виникають всередині ядра, ззовні.

Config – частина, що відповідає за зчитування конфігураційних файлів проекту. Містить у собі *.php* файли, що являють собою функції, повертаючі зчитані масиви ініціалізаційних даних. Таких файлів може бути безліч, але всі вони мають одну задачу – ініціалізувати веб-додаток при зверненні користувача до нього. Config наповнюється такими «функціями» з ходом побудови додатку. Так, наприклад, тут можуть бути ініціалізатори баз даних, допоміжних бібліотек, журналюючих класів тощо. Цією частиною додатку користуються як Framework, так і Application.

Public – найменша частина додатку, але не за значенням. Містить у собі «точку входу» в додаток – файл *index.php*, що відповідає за підключення необхідних файлів класів, об’явлення констант, завантаження маршрутів Route для роутера Router та ініціалізацію журналювання. Слід помітити, що для перенаправлення запиту до цієї папки, використовується файл .htaccess, який встановлює правила для сервера при обробці користувацьких запитів.

Public – єдина папка, до якої користувач має прямий доступ, тобто може отримати (завантажити) з неї файли. Інші частини додатку є комерційними та не є вільними для росповсюдження, оскільки вони містять у собі логіку додатку, «бізнес-логіку», комерційний код тощо. Тож, Public, окрім як точка входу, являє собою репозиторій ресурсів, необхідних користувачеві, таких як каскадні таблиці стилів CSS, JavaScript файли, зображення та шрифти.

Application – рівень додатку. У цій частині знаходяться всі контроллери та моделі, маршрути Route, об’єкти «бізнес-логіки», шаблони та сторінки. Під час побудови додатку, ця частина найбільш підлягає метаморфозам.

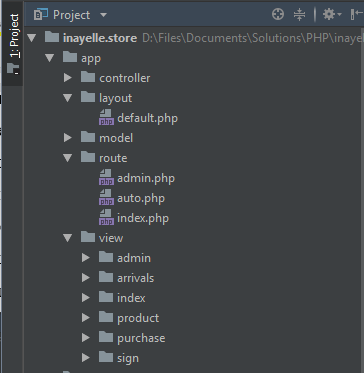
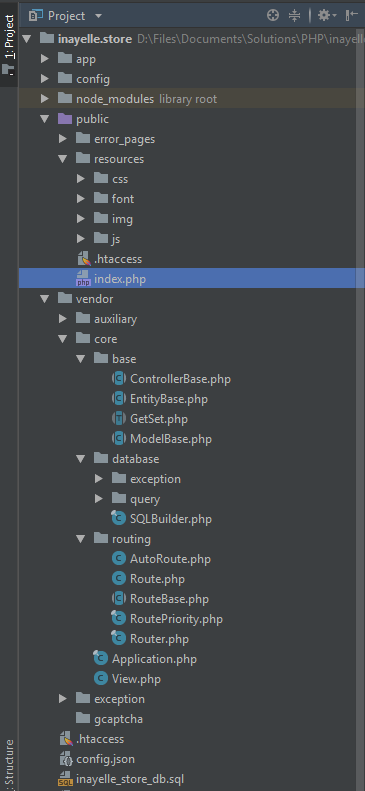


Рисунок 2.2.1, 2.2.2 – загальний вигляд структури проекту

## 2.3 Допоміжні класи

### 2.3.1 Класи SQLBuilder та QueryBase

Для більш ефективної та швидкої роботи з базою даних було створено абстракцію над вбудованими в PHP засобами роботи з базами даних PDO. SQLBuilder являє собою повністю статичний клас, що реалізований за паттерном «фабрика». SQLBuilder вміє створювати підключення до бази даних, а також створювати екземпляри класу QueryBase чотирма різними способами:

* створити запит на вибірку
* створити запит на створення
* створити запит на оновлення
* створити запит на видалення

Результатом виконання цих статичних методів є деякі похідні від QueryBase класи, що відповідають за покрокове створення запиту. Похідні від QueryBase класи мають контракт на магічний метод *\_\_invoke()*, результатом виконання якого є вказівник *this* зі створеним запитом. Таким чином забезпечується можливість створювати «ланцюгові конструкції», за допомогою яких генерується SQL-запит. Слід помітити, що такий підхід дуже різко мінімізує кількість помилок при побудові запитів.

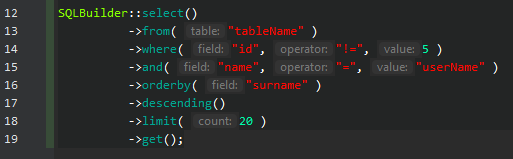


Рисунок 2.3.1 – «ланцюгова» побудова запиту на вибірку

### 2.3.2 Класи Router, RouteBase та похідні

Клас Router виконує роль шукача. Користувач надає запит за певним URI, а Router за цим URI має знайти відповідний контроллер та метод у ньому, виконати його та повернути графічне представлення. Router містить у собі список доступних шляхів (похідні класи від RouteBase), і які він перебирає в пошуку потрібного шляху.

Клас RouteBase є абстрактним класом і надає інтерфейс для перевірки на співпадіння пари URL-шляху (метод *matchURI*).

Збудований фреймворк дозволяє додавати шляхи декількома способами.

Крім того, на кожен шлях можливо додати певний список правил-методів, що будуть викликані перед тим, як викликати Action. Це зроблено для більшої гнучкості сайту. Як приклад, існує шлях adminRoute, доступ до перегляду якого є строго обмежено – дозвіл є лише у адміністраторів. У випадку невиконання даних умов, користувач перенаправляється на одну із помилкових сторінок, відповідно до HTTP статус-коду.

Звичайний шлях – клас Route. Одному представленню URI відповідає одна пара Controller-Action. Метод matchURI порівнює URI з заданим шляхом оператором рівності.

Автоматичний шлях – клас AutoRoute. Має два способи застосування:

* автоматичний MVC диспатч
* диспатч с перевизначеним регулярним виразом

Перший спосіб дозволяє не піклуватись про ручне додавання стандартних MVC-шляхів, оскільки AutoRoute перевіряє URI не звичайним порівняням, а регулярним виразом.

Другий спосіб дозволяє прикладному програмісту створювати власні (короткі) шляхи за регулярним виразом.

Всі шляхи зберігаються в Application\\Routes у вигляді файлів-функцій. Під час ініціалізації додатку, всі шляхи завантажуються з файлів у масив шляхів в класі Route. При обробці користувацького URI необхідно викликати

# метод Router::dispatch, який визначить контроллер та метод. У випадку неспівпадіння шляхів, користувач буде направлений на сторінку, що відповідає HTTP статус-коду 404 (Page not found).ВИСНОВКИ

Під час реалізації курсової роботи було створено власний фреймворк мовою PHP, на його базі побудовано веб-сайт «Інтернет-магазин», що дозволяє швидко знайти, обрати, замовити, відфільтрувати товари. Верстка   
сайту є адаптивною, тому забезпечена комфотна робота з будь-яких пристроїв.

Додаток реалізує в собі роботу з базою даних, диспатчеризацію, обробку користувацьких даних. Також було створено меню для адміністраторів задля простого та швидкого доступу до бази даних.

Після створення курсового проекту, було покращено навички роботи з мовою програмування PHP, мовою структурованих запитів SQL, здобуті навички використання паттернів GOF.

# ЛІТЕРАТУРА

1. [php.net – офіційна документація мови PHP](http://php.net)
2. [https://habr.com/post/181772/ - ресурс Habrahabr](https://habr.com/post/181772)
3. [https://ru.wikibooks.org/wiki/PHP - ВикиУчебник](https://ru.wikibooks.org/wiki/PHP)
4. [http://phpfaq.ru/pdo - Руководство по PDO](http://phpfaq.ru/pdo)
5. Самоучитель PHP, Кузнецов М., Симдянов И. : «БХВ-Петербург», 2018. — 448 с.
6. MySQL по максимуму, 3-е издание, Бэрон Шварц: Питер Пресс, 2018.-864 с.
7. PHP Objects, Patterns, and Practice, 4th edition, Matt Zandstra: Apress, 2006.-511 с.
8. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес; [пер. с англ.: А. Слинкин науч. ред.: Н. Шалаев]. - Санкт-Петербург [и др.] : Питер, 2014. - 366 с. : ил. ; 24 см.
9. PHP and MySQL Web Development, 4th edition, Luke Welling, Laura Thomson

# ДОДАТОК 1. СКРІНШОТИ ВЕБ-САЙТУ



Рисунок д.1 – головна сторінка сайту (ПК)

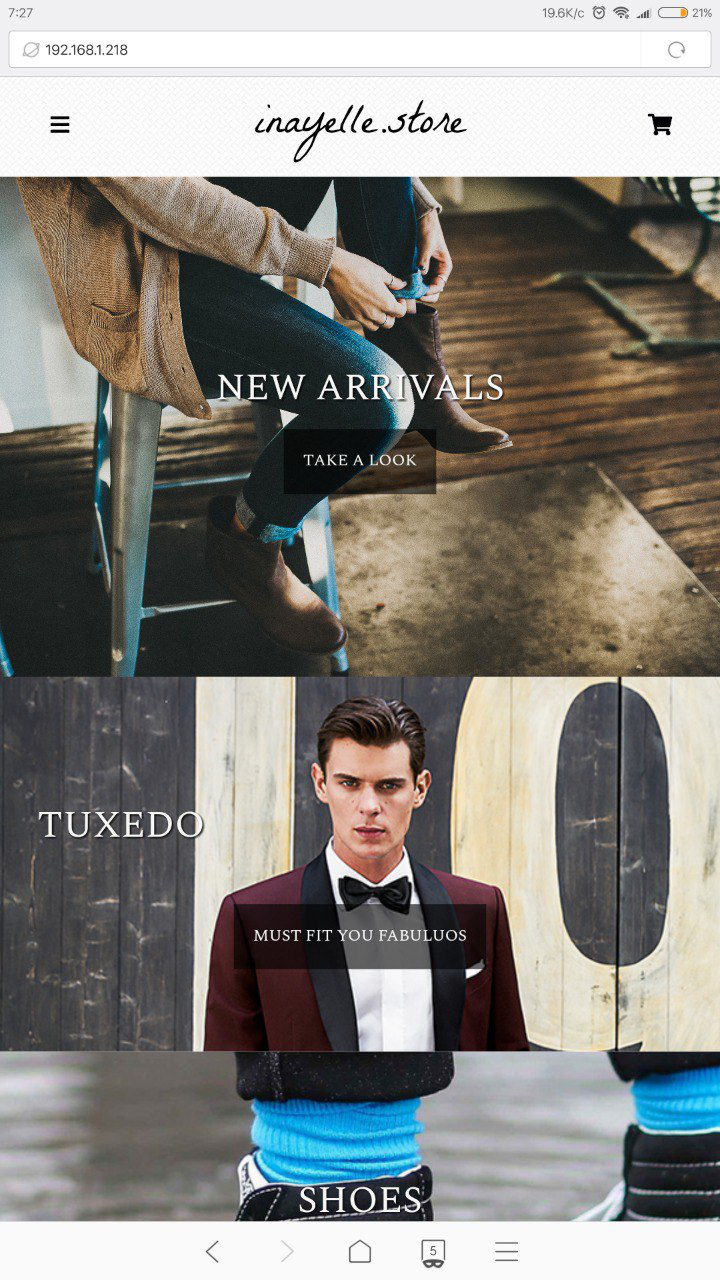


Рисунок д.2 – головна сторінка сайту (смартфон)

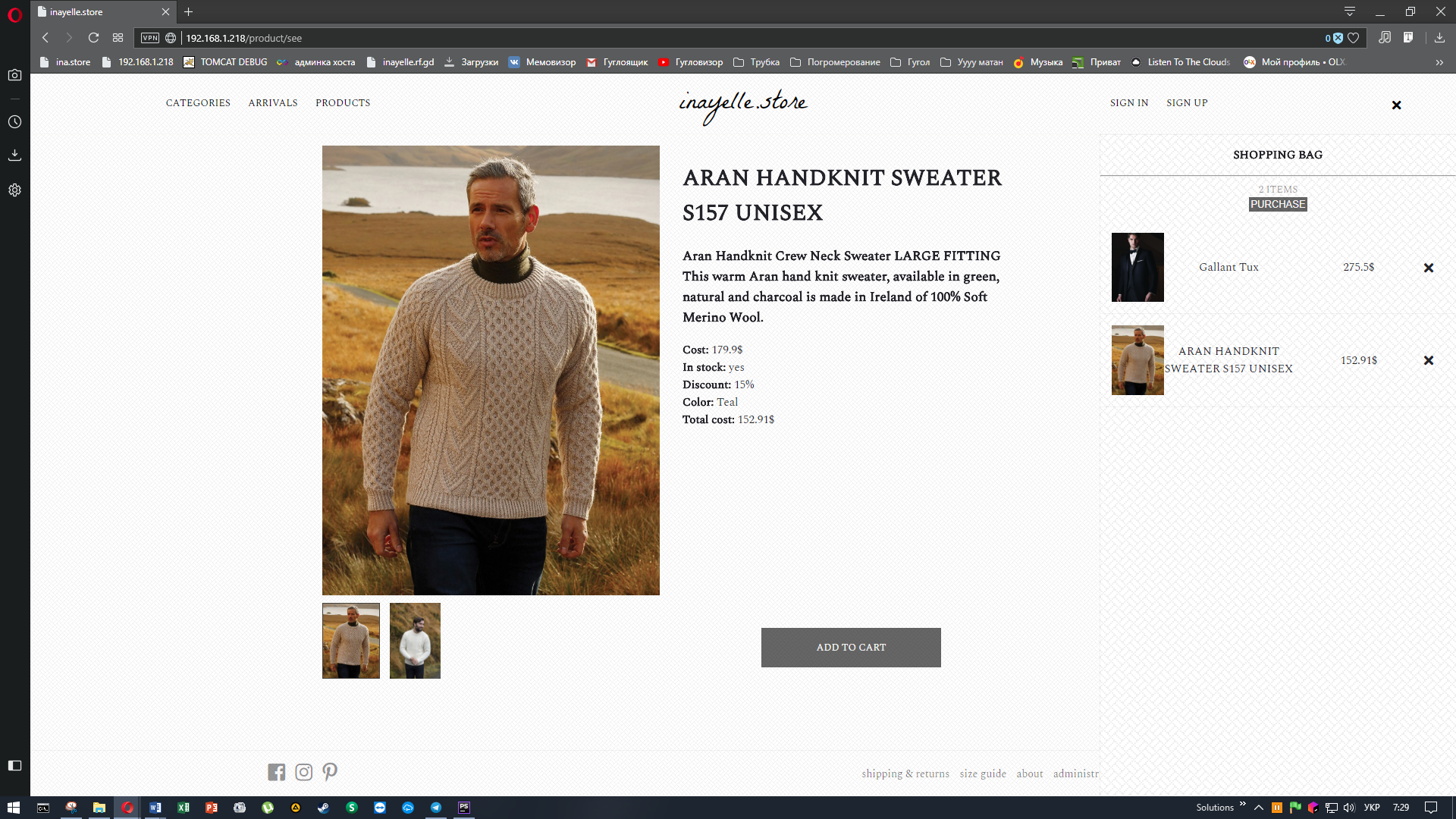


Рисунок д.3 – перегляд сторінки товару та кошику

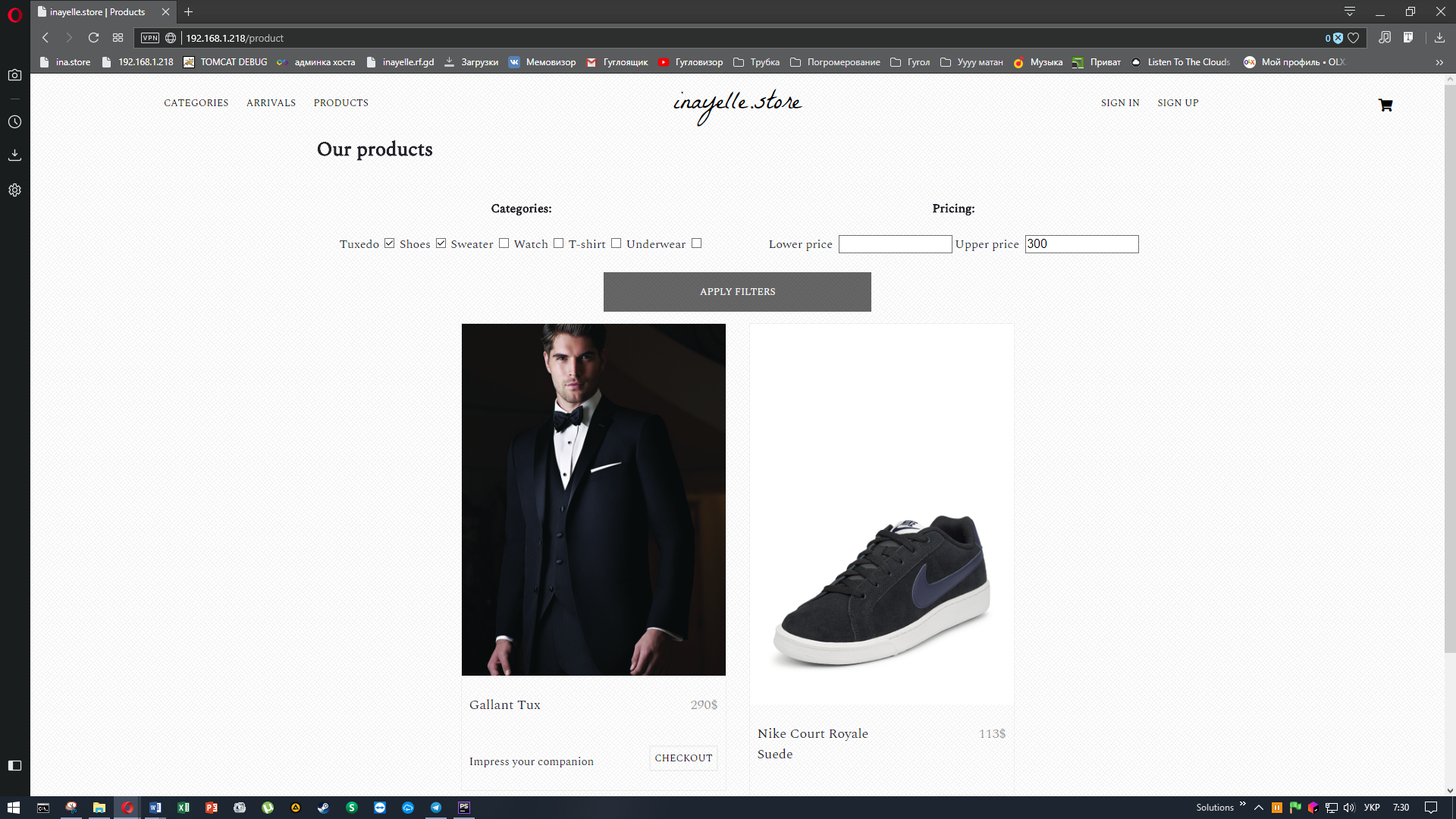


Рисунок д.4 – фільтрація товарів за категоріями і цінами

# ДОДАТОК 2. ДІАГРАМА КЛАСІВ ВЕБ-ДОДАТКУ

