МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри

Аліна САВЧЕНКО

“ ”_____2021 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ

“МАГІСТРА”

ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ “ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ”

Тема: “Метод проектування та розробки системи керування контентом інтернет-магазину на базі CMS 1C-Bitrix”

Виконавець: Тертичний Владислав В'ячеславович

Керівник: професор Віноградов Микола Анатолійович

Нормоконтролер: _____ Ігор РАЙЧЕВ

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра Комп'ютерних інформаційних технологій

Галузь знань, спеціальність, освітньо-професійна програма: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології"

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Аліна САВЧЕНКО

«____»_____2021р.

ЗАВДАННЯ

на виконання дипломної роботи студента

Тертичний Владислав В'ячеславович

(прізвище, ім'я, по батькові)

1. Тема роботи: «Метод проектування та розробки системи керування контентом інтернет-магазину на базі CMS 1C-Bitrix» затверджена наказом . ректора від 12.10.2021 за № 2228/ст.

2. Термін виконання роботи: з 12.10.2021 по 31.12.2021.

3. Вихідні дані до роботи: Теоретичне та практичне описання створення WEB-сайту. Опис використаного програмного забезпечення та використаних мов програмування: **PHP**, **JavaScript** та **HTML** (Hypertext Markup Language) і **CSS** (Cascading Style Sheets).

4. Зміст пояснювальної записки: вступ, огляд теоретичної бази, оцінювання затрат часу на виконання. Розробка ТЗ (технічного завдання). Опис вибраних компонентів. Розробка Web-сайту. Розробка документації для користування та розробці.

5. **Перелік обов'язкового ілюстративного матеріалу:** слайди, презентація.

6. Календарний план-графік

№ п/п	Завдання	Термін виконання	Підпис керівника
1.	Аналіз літератури та джерел за темою дипломного проекту	12.10.2021 – 15.10.2021	
2.	Розробка моделі роботи фреймворку 1c-bitrix	16.10.2021 – 19.10.2021	
3.	Архітектура MVC для Bitrix Framework	20.10.2021 – 24.10.2021	
4.	Швидкодія сайтів на Бітрікс	25.10.2021 – 31.10.2021	
5.	Розробка моделі взаємодії компонентів	01.11.2021 – 07.11.2021	
6.	Технічне завдання на створення сайту	08.11.2021 – 17.11.2021	
7.	Тестування	18.11.2021 – 01.12.2021	
8.	SEO аналіз	02.12.2021 – 11.12.2021	
9.	Створення презентації, доповіді та підготовка до захисту дипломної роботи	12.12.2021 – 20.12.2021	

7. Дата видачі завдання: 12.10.2021р.

Керівник дипломної роботи _____ Микола ВІНОГРАДОВ

(підпис керівника)

Завдання прийняв до виконання _____ Владислав ТЕРТИЧНИЙ

(підпис керівника)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Проектування та розробка системи керування контентом інтернет-магазину на базі CMS 1C-Bitrix» викладена на 89 сторінках, містить 10 рисунків, 3 таблиці, 12 літературних джерел.

Мета проекту **–** розробка системи керування контентом на базі CMS 1C-Bitrix для реалізації продажу товарів.

Об'єкт дослідження **–** власноруч розроблена система управління інтернет-магазином.

Предмет дослідження **–** Web-сайт, інтернет-магазин, створений на базі CMS 1C-Bitrix.

Ключові слова - WEB інтерфейс, 1с Бітрікс, PHP, MVC

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

MVC - Model View Controller

PHP - personal home page

JS - JavaScript

HTTP - Hyper Text Transfer Protocol

API - application programming interface

FTP - File Transfer Protocol

СУБД - система управління бази даних

БД - база данных

URL - Uniform Resource Locator

UML - Unified Modeling Language

ВСТУП

Всі програмні продукти Бітрікс зроблені на основі Bitrix Framework. У разі якщо адресуватися до розділу підтримки на веб-сайті 1С-Бітрікс, то можливо побачити це визначення:

Bitrix Framework - це розроблена на базі PHP фреймворк для розробки веб-додатків. На цій платформі фірмою «1С-Бітрікс» зроблені 2 відомих продукту: «1С-Бітрікс: Управління сайтом» і «1С-Бітрікс: Колективний портал».

Це визначення зрозуміло для веб-розробників, але досить важке для розуміння користувачам.

Framework - це певний «каркас», для створення програмних продуктів.
PHP - це мова програмування, на якому написаний даний «каркас».
На базі цього «каркаса», даної платформи фірма Бітрікс зробила програмні продукти «1С-Бітрікс: Управління сайтом» і «1С-Бітрікс: Колективний портал». Це вже готові CMS, які можливо ввести на хостинг, налаштувати і працювати з ними, як і з будь-який інший системою управління веб-сайтом.

З технічної точки зору програмні продукти Бітрікс (сайти і мобільні додатки) формуються на базі особистої платформи Bitrix Framework.

Недоліками Bitrix Framework є:
Швидкодія, через те, що реалізовано багато функціональних методів, не всі користувачі використовують їх вірно, що знижує загальну швидкодію системи;
Недостатньо дружній інтерфейс, система написано для покращення функціональності інтернет-магазину,що є перевагою, але в цей час розробники недостатньо попрацювали на User Experience адміністративної панелі системи 1С-Bitrix.

Для покращення цих показників необхідно проаналізувати, який з компонентів системи не використовуються за призначення та допрацювати ядро системи 1C-Bitrix та вимкнути ці компоненти для конкретно реалізованого проекту, що дасть змогу покращити швидкість завантаження сторінок та покращити їх індексацію в пошукових системах.

Розділ 1. Аналітичний огляд і постановка завдання

1.1. Аналіз літератури та джерел за темою дипломного проекту.

Програмування в Bitrix Framework не має особливих труднощів. Це просте програмування на PHP. Особливості, природно, є, але це не ті особливості, які роблять з "розробників програмного забезпечення на Бітрікс" якусь особливу касту.

Основними етапами підготовки до розробки проекту на базі системи керування сайтов 1С – бітрикс є:

- Командна PHP-рядок
- організація розробки
- Система контролю версій
- Папка / local
- Composer і Bitrix Framework
- Bitrix CLI
- Трохи теорії PHP
- Зауваження по $ arParams і $ arResult
- HTTP POST запити

- Теорія. Права доступу
- Теорія. Файли і База даних
- Робота з базами даних
- Теорія. відкладені функції
- Теорія. файл init.php
- Теорія. мовні файли
- Робота з мовними файлами
- Теорія. Гаджети та їх створення
- JS-бібліотека
- Підключення JS-коду
- JS-клас до шаблону компонента
- JS-розширення медіаплеєра
- Приклади кастомізації публічної частини
- Форматування дат в Javascript
- Типові помилки та поради
- Розширення (extensions)
- Інструмент bitrix / cli
- Bitrix / cli: збірка проекту з NPM
- вкладені бібліотеки
- Використання ES6
- Робота з магазином
- Товари та CIBlockElement :: GetList
- Призначені для користувача типи властивостей замовлення
- призначені для користувача обмеження
- Призначені для користувача правила компаній

➢ Кастомізація типів додаткових послуг

➢ Кастомізація служб доставок

➢ Кастомізація платіжних систем

➢ Кастомізація шаблону платіжної системи

➢ Власний обробник онлайн-каси

➢ Приклад створення власної служби доставки

➢ Приклад створення замовлення через API

➢ Приклад зміни замовлення через API

➢ Приклад поділу оплати на 2 частини

➢ Робота з модулем Push & Pull

➢ Оптимізація кількості запитів до сервера

➢ Push & Pull для гостей

➢ Підписка на події модуля

Детальніше розберемо правила та стандарти розробки проекту в системі 1С-Бітрікс

Довжина рядку - Потрібно ігнорувати рядки довжиною більше ніж 120 знаків. У разі якщо рядок вище даної величина, то треба застосувати критерії перенесення рядки.

Критерії перенесення рядків - У разі якщо довжина рядка вище 120 знаків, то потрібно скористатися належними правилами перенесення:

— виносити після коми або ж перед оператором;

— пернести рядок повинен бути зрушений порівняно верхнього на один знак табуляції;

— переноси зобов'язані бути в манері UNIX.

Прогалини і табуляція - Для форматування відступів в коді треба застосувати табуляцію. Впровадження прогалин забороняється:

— в разі застосування табуляції будь-хто має можливість налаштувати у власному редакторі бажаний відступ;

— застосовується один знак замість декількох;

— в разі змішування прогалин і табуляції слово стане скакати, руйнуючи форматування.

Форматування підпорядкованості. Підлеглий код зобов'язаний бути зміщений від головного рівно на один знак табуляції. Підлеглий код не має можливість перебувати на що ж рядку, власне що і ключовий. Приклад представлений нижче:

```
function func()
{
     if (condition)
     {
          while (condition2)
          {

          }
      }
}
```

Рис. 1.1. Критерії розстановки фігурних дужок

Відкриваюча дужка зобов'язана ставиться під відповідним оператором і на одному абзаці з ним. Закриває дужка зобов'язана ставиться під відповідною відкриваючою. Приклад:

```
if ($condition)
{
   ...
}
```

Рис. 1.2. Впровадження тернарного оператора "?:"

Умова заключається в дужки, що найбільш відокремлює його від решти коду. За здатністю, впливу, виконувані за умовою, зобов'язані бути простими функціями. У разі якщо цілий блок розгалуження погано читається, то варто поміняти його на if / else.

Приклад: (умова? Funct1 (): func2 ());

Вирази

Краще, щоб в будь-якої рядку знаходилося лише тільки один вислів. Приклад: Невірно: $ A = $ b; $ B = $ c; $ C = $ a; Вірно: $ A = $ b; $ B = $ c; $ C = $ a;

При написанні функцій слід строго використовуватися правило Форматування підпорядкованості: тіло функції має бути зрушене на один знак табуляції направо від самого функції. Фігурні дужки повинні застосовуватися щоразу, перебувати на окремих рядках, на одному рівні з анотацією. Приклад:

Невірно писати так:

```
if ($a == 0) $a = 10;
else{
$a = 5;
$b = 10;}
```

Вірно писати так:

```php
if ($a == 0)
{
  $a = 10;
}
else
{
  $a = 5;
  $b = 10;
}
```

Важкі конструкції

Важкі конструкції треба розбивати по рядках. Наприклад:

```
if(COption::GetOptionString("main", "new_user_registration", "N")=="Y"
&& $_SERVER['REQUEST_METHOD']=='POST' &&
     $TYPE=="REGISTRATION" && (!defined("ADMIN_SECTION") ||
ADMIN_SECTION!==true))
```

Рис. 1.3. Важкі конструкції по рядках

Дозволяється записати як:

```
if (COption::GetOptionString("main", "new_user_registration", "N") ==
"Y"
     && $_SERVER['REQUEST_METHOD'] == 'POST' && $TYPE == "REGISTRATION"
     && (!defined("ADMIN_SECTION") || ADMIN_SECTION !== true))
{
}
```

Надважкі конструкції рекомендовано бити на кілька більше нескладних. Наприклад:

```
if((!(defined("STATISTIC_ONLY") && STATISTIC_ONLY &&
substr($APPLICATION->GetCurPage(), 0,
     strlen(BX_ROOT."/admin/"))!=BX_ROOT."/admin/")) &&
COption::GetOptionString("main", "include_charset", "Y")=="Y"
     && strlen(LANG_CHARSET)>0)
```

Рис. 1.4. Надважкі конструкції

Можна записати так:

```
$publicStatisticOnly = False;
if (defined("STATISTIC_ONLY")
  && STATISTIC_ONLY
  && substr($APPLICATION->GetCurPage(), 0, strlen(BX_ROOT."/admin/"))
!= BX_ROOT."/admin/")
{
      $publicStatisticOnly = True;
}
if (!$publicStatisticOnly && strlen(LANG_CHARSET) > 0
      && COption::GetOptionString("main", "include_charset", "Y") ==
"Y")
{
}
```

Або ж так:

```
if (!defined("STATISTIC_ONLY") || ! STATISTIC_ONLY
  || substr($APPLICATION->GetCurPage(), 0, strlen(BX_ROOT."/admin/"))
== BX_ROOT."/admin/")
{
  if (strlen(LANG_CHARSET) > 0 && COption::GetOptionString("main",
"include_charset", "Y") == "Y")
  {
  }
}
```

Форматування масивів

Масиви, які записуються в кілька рядків, слід форматувати наступним чином**:**

```
$arFilter = array(
  "key1" => "value1",
  "key2" => array(
      "key21" => "value21",
      "key22" => "value22",
  )
);
```

Рис.1.5. Масиви в кілька рядків

Порожні рядки

Порожні рядки можуть допомогти бити код програми на закономірні розділи. Кількома рядками можуть ізолюватися секції в початковому файлі. Однією беззмістовний рядком відокремлюються один від одного для більше комфортного читання.

Пробіли

Після коми зобов'язаний бути пробіл. Після крапки з комою, в разі якщо вона не остання в рядку (наприклад, в конструкції for), зобов'язаний бути пробіл. Перед комою або ж крапкою з комою пропуски не ставляться. Всі оператори зобов'язані бути розділені пропуском від операндів з обох сторін. Підміна пробілу символом табуляції не допускається.

1.2. Розробка структури проекту

1.2.1. Файлова структура проекту

admin / - каталог з адміністративними скриптами модуля;

menu.php - файл з адміністративним раціони модуля;

classes / - скрипти з класами модуля;

general / - класи модуля, які не залежать від застосовуваної бази даних;

mysql / - класи модуля, призначені для роботи лише тільки з MySQL;

mssql / - класи модуля, призначені для роботи лише тільки з MS SQL;

oracle / - класи модуля, призначені для роботи лише тільки з Oracle;

lang / ID мови / - каталог з мовними файлами скриптів модуля;

lib / - каталог з файлами (API: класи, логіка) свіжого ядра D7 (може не існувати, в разі якщо у модуля немає особистих методів);

install / - каталог з файлами застосовуваними для установки і деінсталяції модуля;

admin / - каталог зі скриптами які підключають адміністративні скрипти модуля (викликають скрипти);

js / - каталог з js-скриптами модуля. Копіюються в / bitrix / js / ID_модуля /;

db / - каталог з SQL скриптами для інсталяції / деінсталяції бази даних;

mysql / - SQL скрипти для інсталяції / деінсталяції таблиць в MySQL;

mssql / - SQL скрипти для інсталяції / деінсталяції таблиць в MS SQL;

oracle / - SQL скрипти для інсталяції / деінсталяції таблиць в Oracle;

images / - каталог з зображеннями застосовуваними модулем; згодом установки модуля вони зобов'язані бути скопійовані в каталог / bitrix / images / ID модуля /;

templates / - каталог з компонентами 1.0 модуля. (Каталог зберігається лише тільки з метою порівнянності версій.);

ID модуля / - каталог з провідними файлами компонент;

lang / ID мови / ID модуля / - в заданому каталозі присутні мовні файли складову модуля;

components / простір імен / ім'я компонента / - каталог з компонентами 2.0 модуля;

themes / імя_модуля / - має css і малюнки для стилів адміністративної панелі, в разі якщо модуль в таких потребує (Застаріла, до версії 12.0);

panel / імя_модуля / - має css і малюнки для стилів адміністративної панелі, в разі якщо модуль в таких потребує.

index.php - файл з описом модуля;

version.php - файл з номером версії модуля. Версія не має можливість бути рівною нулю.

include.php - цей файл підключається в що момент, коли мова йде про включення модуля в коді, в ньому повинні перебувати підключення всіх файлів з бібліотеками функцій і класів модуля;

default_option.php - має массив з ім'ям $ ID модуля default_option, в якому задані сенсу за замовчуванням для характеристик модуля; [4]

Примітка: У разі партнерських модулів, в заголовку яких знаходиться баста (приклад - mycompany.forum) в імені змінної баста стане механічно замінена на знак підкреслення.

options.php - цей файл підключається на сторінці опції характеристик модулів в адміністративному раціони Установки та

prolog.php - файл має можливість включатися в усіх адміністративних скриптах модуля. Як правило в ньому орієнтується константа ADMIN_MODULE_NAME (ідентифікатор модуля), що застосовується в панелі управління;

.settings.php - файл опцій модуля, що описує опції модуля, які можна прочитати крізь \ Bitrix \ Main \ Config \ Configuration :: getInstance ($ module).

Будь-який модуль зобов'язаний бути коректно описаний в системі для того щоб, система знала, як з даним модулем працювати. Неправильно описані модулі можуть привести до абсолютної або ж вибіркової непрацездатності системи (наприклад, не буде працювати система оновлень).

Головним файлом, застосовуваним системою для маніпуляції модулем, вважається / bitrix / modules / ID модуля / install / index.php. (ID модуля в даному випадку - це абсолютний код партнерського модуля, який задається в форматі: код_партнера.код_модуля.) Провідне призначення цього файлу - це розміщення

в ньому класу з ім'ям, що збігається з ID модуля. (ID модуля тут застосовується в форматі код_партнера_код_модуля, наприклад як в імені класу, приклад:

```
<?
Class mymodule extends CModule
{
    var $MODULE_ID = "mymodule";
    var $MODULE_NAME;

    function DoInstall()
    {
        global $DB, $APPLICATION, $step;
        $APPLICATION->IncludeAdminFile(GetMessage("FORM_INSTALL_TITLE"),
SERVER["DOCUMENT_ROOT"]."/bitrix/modules/mymodule/install/step1.php");
    }

    function DoUninstall()
```

Рис. 1.6. Формат ID модуля

Обов'язкові методи цього класу:

DoInstall - запускається при натисканні кнопки Встановити на сторінці Модуля в частині адміністративного розділу.

DoUninstall - запускається при натисканні кнопки Видалили на сторінці Модуля в частині адміністративного розділу, для реалізація видалення модуля.

Необов'язковий спосіб цього класу:

GetModuleRightList - повертає перелік оригінальних прав (або ролей) модуля.

Обов'язкові якості об'єкта цього класу:

MODULE_ID - зберігає ID модуля (повний код партнерського модуля);

MODULE_VERSION - нинішня версія модуля в форматі XX.XX.XX;

MODULE_VERSION_DATE - рядок має дату версії модуля; дата повинна бути задана в форматі YYYY-MM-DD HH: MI: SS;

MODULE_NAME - ім'я модуля;

MODULE_DESCRIPTION - опис модуля;

MODULE_GROUP_RIGHTS - в разі якщо заданий спосіб GetModuleRightList, то ця властивість матиме значення Y.

Приклад файлу з описом модуля Веб-форми:

```
<?
global $MESS;
$PathInstall = str_replace("\\", "/", __FILE__);
$PathInstall = substr($PathInstall, 0, strlen($PathInstall)-strlen("/index.php"));
IncludeModuleLangFile($PathInstall."/install.php");
include($PathInstall."/version.php");
if(class_exists("form")) return;
Class form extends CModule
{
   var $MODULE_ID = "form";
   var $MODULE_VERSION;
   var $MODULE_VERSION_DATE;
   var $MODULE_NAME;
   var $MODULE_DESCRIPTION;
   var $MODULE_GROUP_RIGHTS = "Y";

   function form()
   {
      $this->MODULE_VERSION = FORM_VERSION;
      $this->MODULE_VERSION_DATE = FORM_VERSION_DATE;
      $this->MODULE_NAME = GetMessage("FORM_MODULE_NAME");
                                          $this->MODULE_DESCRIPTION      =
GetMessage("FORM_MODULE_DESCRIPTION");
   }

   function DoInstall()
   {
      global  $APPLICATION;
```

```
            $FORM_RIGHT = $APPLICATION->GetGroupRight("form");
            if ($FORM_RIGHT=="W")
            {
               $step = IntVal($step);
               if($step<2)

$APPLICATION->IncludeAdminFile(GetMessage("FORM_INSTALL_TITLE"),

$_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/form/install/step1.php");
               elseif($step==2)

$APPLICATION->IncludeAdminFile(GetMessage("FORM_INSTALL_TITLE"),

$_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/form/install/step2.php");
            }
         }

         function DoUninstall()
         {
            global $APPLICATION;
            $FORM_RIGHT = $APPLICATION->GetGroupRight("form");
            if ($FORM_RIGHT=="W")
            {
               $step = IntVal($step);
               if($step<2)

$APPLICATION->IncludeAdminFile(GetMessage("FORM_UNINSTALL_TITLE"),

$_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/form/install/unstep1.php");
               elseif($step==2)

$APPLICATION->IncludeAdminFile(GetMessage("FORM_UNINSTALL_TITLE"),

$_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/form/install/unstep2.php");
```

```
        }
    }

    function GetModuleRightList()
    {
        global $MESS;
        $arr = array(
            "reference_id" => array("D","R","W"),
            "reference" => array(
                GetMessage("FORM_DENIED"),
                GetMessage("FORM_OPENED"),
                GetMessage("FORM_FULL"))
            );
        return $arr;
    }
}
?>
```

Характеристики модуля доступні для конфігурації в адміністративному інтерфейсі на сторінці Опції модулів (Установки> Опції продукту> Опції модулів).

Параметри

Параметри модуля доступні для зміни в адміністративному інтерфейсі на сторінці Налаштування модулів (Установки> Установки продукту> Налаштування модулів). При виборі модуля на даній сторінці, система підключає файл / bitrix / modules / ID модуля / options.php, призначений для управління параметрами модуля, призначення прав на модуль і т.п.

Параметри модуля зберігаються в базі даних.

При отриманні параметрів модуля, може використовуватися значення за замовчуванням, що задається в файлі / bitrix / modules / ID модуля / default_option.php. В даному файлі визначається масив $ID модуля_default_option, який зберігає значення за замовчуванням.

Приклад файлу / bitrix / modules / ID модуля / default_option.php:

```
<?
// установим строковый параметр
COption::SetOptionString("my_module_id", "MY_PARAMETER_ID", "VALUE");

// получим строковый параметр
$value = COption::GetOptionString("my_module_id", "MY_PARAMETER_ID", "DEFAULT_VALUE");
?>
```

Для роботи з параметрами модуля призначений клас COption. Методи класу:

- SetOptionString - установка строкових параметрів
- SetOptionInt - установка числових параметрів
- GetOptionString - отримання строкових параметрів
- GetOptionInt - отримання числових параметрів
- RemoveOption - видалення параметра

1.2.2. Розробка моделі роботи фрейворку 1c-bitrix

Будь-яке ПЗ, розвиваючись, має відповідати заявленій спочатку цілі. Це питання вирішує архітектурне проектування.

Архітектура структура продукту - підхід до проектування, що гарантує, що програмне забезпечення буде відповідати своєму призначенню.

Архітектура програмного забезпечення - спосіб структурування програмної системи . Система може включати кілька рівнів абстракції і мати багато рівнів роботи, і на кожному рівні може буде розроблена окрема архітектура.

Архітектура Bitrix Framework вирішує наступні завдання:

— *Наступність*. Кожен новий реліз продуктів підтримує всі попередні рішення і технології. Це дозволяє здійснювати перехід на нові версії продуктів сайтів, створених на практично будь-якої попередньої версії.

— Єдність принципів роботи з будь-якою версією і будь-яким рішенням на базі системи.

— *Безпека*. Архітектура дозволяє створити достатній рівень безпеки для сайтів будь-якої спрямованості.

— *Масштабованість*. Чи не накладено ніяких обмежень на розвиток проектів в міру зростання контенту, сервісів, числа користувачів.

— *Продуктивність*. Швидкість роботи системи залежить від якості налаштування її елементів, тобто в більшій мірі на продуктивність впливає рівень підготовки розробника проекту, можливості хостингу.

Можливість розвитку системи зусиллями сторонніх розробників. Архітектура не накладає ніяких обмежень на створення власних модулів, компонентів, рішень

1.2.3. Архітектура MVC для Bitrix Framework

Шаблон **MVC** для Bitrix Framework:

- Модель - це API;
- Представление - це шаблон;
- Контроллер - це компонент.

Модель–вигляд–контролер (**MVC**) — архітектурний підхід, на базі якого проектується та розробляється програмне забезпечення. (рис 2.5.).

Цей архітектурний підхід відображає розподіл системи на три частини: модель даних, вигляд (інтерфейс користувача) та контролер. Застосовується для того щоб максимально відокремити обчислення та логіку додатку від представлення для користувача, що робить шаблони проекту “чистими”, через те вся бізнес логіка відбувається в моделі.

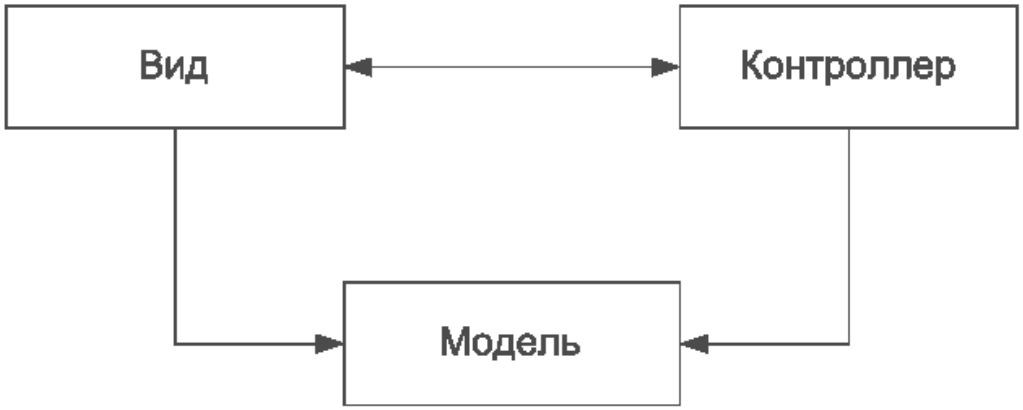


Рис. 1.7. Схема MVC

Модель–вигляд–контролер (або **Модель–представлення–контролер**, а нгл. *Model-view-controller*, **MVC**) — архітектурний шаблон, який

використовується під час проектування та розробки програмного забезпечення (рис 1).Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Bitrix Framework за рівнями архітектури структуру можна описати так:

- модулі
- компоненти
- файли сторінок
- шаблони
- компоненти
- сторінка
- компонент:
- виклик
- параметри
- сторінка:

Модуль – це так звана модель даних в MVC форматі і API для роботи з цими даними. Статичні методи класів модуля можуть викликатися в компонентах, шаблонах, інших модулях. Також завдяки продуманій архітектурі можуть бути створенні екземпляри класів.

Кілька десятків модулів системи містять набір функцій, необхідних для реалізації глобального завдання: робота інтернет-магазину, організація мережі та інші. Модулі також містять інструментарій який може використовувати адміністратор\менеджер системи для управління цими функціями.

Ядро продукту - файли, що знаходяться в директорії /bitrix/modules/, а також файли системних компонентів: /bitrix/components/bitrix/.

компоненти

Компонент – це літера С в моделі MVC, контролер і представлення для використання в публічній частині сайту. Компонент за допомогою моделі одного або декількох модулів керує даними. представлення компонента виводить дані на сторінку. [11]

Компоненти входять до складу модулів, але вирішують більш вузьку, приватну задачу - наприклад, виводять список новин або товарів. Вносити свої зміни в код продукту рекомендується на рівні компонентів. Програміст може модифікувати їх як завгодно, використовувати свої напрацювання і використовувати необмежену кількість шаблонів на кожен з компонентів. На одній сторінці сайту може розташовуватися кілька компонентів, крім того, їх можна включати в шаблон сайту. Таким чином, програміст має можливість зібрати сайт як конструктор, після чого доопрацювати необхідні компоненти для отримання бажаного результату як у функціональному, так і у візуальному плані.

1.2.4. Рівень доступу.

В системі Bitrix Framework підтримується два рівня розмежування доступу:

Рівень доступу на файли і каталоги

Цей рівень доступу перевіряється в пролозі, задається за допомогою спеціальної директиви файлу .access.php, що містить PHP массив наступного формату:

```
$PERM[файл][ID групи користувачів] = “ID рівня доступу”;
```

де:

файл – найменування файлу для яких призначаються права доступу;

Групи користувачів – ідентифікатор групи користувачів на яку поширюється даний рівень доступу

Ідентифікатор рівня доступу – підтримуються такі значення (в порядку зростання):

D – заборонений

R – читання

U – документообіг (файл може бути відредагований в режимі документообігу);

W – запис (файл може бути відредагований безпосередньо);

X – повний доступ (право на читання та записа)

Якщо користувач належить до кількох груп, то буде вибрано право з найбільшим рівнем доступу для користувача.

Якщо для файлу явно не заданий рівень прав, використовуючи правила розмежування доступу, то береться рівень прав заданий для верхніх каталогів.

Приклад 1

файл .access.php

```
<?
  $PERM[“index1.php”][“2”] = “U”;
  $PERM[“index1.php”][“3”] = “D”;
?>
```

При спробі відкриття сторінки /index.php користувач, що належить групі ID = 3, буде мати право доступу D (заборонено), користувач з групи ID = 2 буде мати право U (документообіг). Користувач, що належить до обох груп, матиме максимальний рівень доступу – U (документообіг).

1.2.5. Права логіки роботи модуля

Якщо мати на увазі звичайні статичні публічні сторінки, то до них буде застосовано тільки перший рівень доступу на файли і каталоги.

Якщо для користувач мінімальне право R (читання) і якщо цей файл є частиною того чи іншого модуля, то перевіряється другий рівень доступу, що реалізується в налаштуваннях відповідного модуля.

Використовуються дві реалізації розмежування прав доступу другого рівня:

- Права;
- Ролі.

Головна різниця в них полягає в тому, в тому випадку якщо користувач має декілька прав, то вибирається максимальне. Але якщо ж користувач володіє декількома ролями, то він матиме можливість сумарними можливостями цих ролей.

Модулі, в яких використовуються ролі, можна побачити в фільтрі Модуль на сторінці Налаштування > Користувачі > Рівні доступу в Адміністративному розділі.

Bitrix Framework реалізований на файлах, що дає більше можливостей розробнику сайту. Через те що файл в системі - це просто файл який виконується, то і виконувати він може що завгодно: хоч власний PHP код розробника, хоч вбудовані компоненти - в будь-якому порядку.

Файли можна правити як по FTP, так і через SSH, не вдаючись до додаткових інструментів СУБД. Їх легко копіювати, переміщати, робити резервні копії і т.п. Строго кажучи, ви можете весь контент зберігати в БД. Але для простих статичних сайтів це буде явне ускладнення і уповільнення. [10]

Реалізація на файлах здається проблематичною в тому плані, що від такої системи очікується десятки тисяч файлів на диску. Зазвичай це не так. Динамічна інформація (новини, каталог товарів, статті) зберігаються в БД модулем Інформаційні блоки. Тоді для виведення, наприклад, десятка тисяч товарів в інтернет-магазині використовується одна єдина фізична сторінка (файл). У цьому файлі викликається компонент Інфоблоки, який в свою чергу обирає і виводить товари з бази даних.

Наприклад, для каталогу товарів дійсно потрібно створити папку на диску, але тільки одну, наприклад, /catalog, помістити туди комплексний компонент і далі сторінки товарів можуть мати вигляд, наприклад: http: //***.com/catalog/1029.html природно, що ці адреси будуть "уявними" і оброблятися системою. Файлів під них в папці /catalog не створюється.

Однак, для кожного товару буде створений файл в кеші, щоб, при наступному зверненні покупця сервер не напружувався з запитами до БД.

Швидкодія сайтів на Бітрікс

Під прискоренням завантаження сторіно всі звикли розуміти роботу по серверній часті сайту. Однак юзеру немає різниці за скільки часу відкривається веб-сторінка у програміста, для нього є вкрай важливим, як швидко відкривається сторінка у нього. Тут вже необхідно виконати декілька важливих комплексних завдань.

Back-End

Не виконуючи налагодження програмного коду (серверної частини веб-сайту), звичайно ж, не обійтися. Тут розробнику потрібно використовувати засоби Моніторингу продуктивності, що дозволяє виконати перевірку всіх сторінок веб-сайту на відповідність якості коду, та множину SQL запитів. Крім того, цей функціонал дає вагомі рекомендації для покращення серверного програмного забезпечення.

Далі необхідно займатися кешуванням компонентів.

Один з важливих кроки - це впровадження технології Composite Site.

Територія розташування сайтів на територіально віддалених веб-серверах може суттєво вплинути на час завантаження інформації для користувача через більшу відповідь серверу. Особливо це актуально для програмного продукту, що орієнтуються на локальні ринки.

Перенесення місцезнаходження хостингу може дати істотний приріст, в деякий випадках прискорення доходило до 2 секунд.

Front-End

Виконавши роботу по серверній частині сайту слід переходити до налаштування інших компонентів, від яких залежить швидкість завантаження

веб-сторінки. У Bitrix Framework є в наявності модуль - Швидкість сайту. Його завдання - полегшити програмісту виконання задач по оптимізації завантаження сторінки.

У цього інструменту є діаграма по хітам, де можна подивитися кожен з етапів завантаження Navigation Timing.

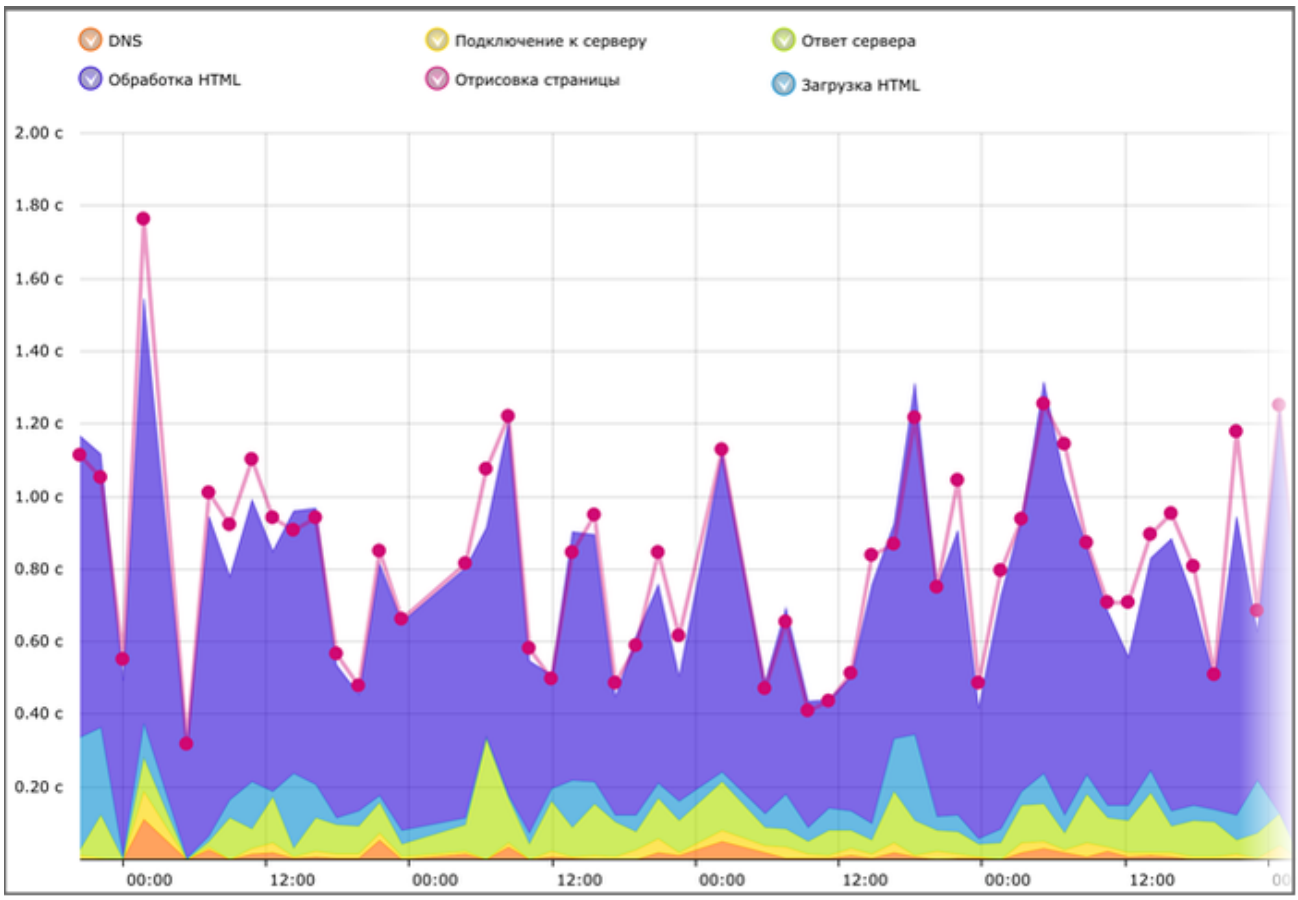


Рис.1.8. Графік завантаженості

На цьому графіку показана зеленим кольором робота серверної частини, а фіолетовим – як виконується front-end частина. Як можна побачити, що значущу частину часу завантаження веб-сторінки браузер витрачає не на back-end, а на front-end.

Заходи по зменшенню числа завантажуваних ресурсів:

— **Потрібно реалізувати об'єднання каскадних стилів і js скриптів**. Браузери мають обмеження на кількість одночасних з'єднань з сервером. Зазвичай це 8, але може доходити навіть до 15, як в Internet Explorer. Тобто, в тому випадку, якщо на сторінці 50 файлів, то браузер завантажить їх в декілька запитів. Якщо ж реалізувати об'єднання цих скриптів та стилів, то швидкість завантаження збільшиться;

— **Включивши CDN**. Однак до його підключення потрібно підходити обачно. Якщо ваш основний користувач знаходиться на певній території, то CDN може уповільнити, якщо найближчий сервер розташований далеко від користувачів;

— **Використовуйте кешування ресурсів в браузері** (Expires / Cache-Control: max-age, Last-Modified / E-tag). Потрібно віддавати заголовки для картинок, css і js-скрипта.

Якщо у вас є дата останнього оновлення, то це може бути дуже корисно. У цьому випадку на наступних хітах браузер буде робити conditional get запит. Тобто буде реалізований запит чи не змінився цей ресурс ось з цієї дати. І завантажувати тільки ресурси які були змінені. Але ж і в цьому випадку доводиться робити один додатковий запит.

Для того щоб уникнути відправки такого запиту досить налаштувати в сервері Apache або NGINX видачу заголовків Expires.

Всі картинки для шаблону повинні бути в sprite. А для дрібних картинок краще використовувати кодування Base64.

Заходи щодо зменшення обсягу даних при завантаженні:

GZIP-стиснення. Рекомендується використовувати серверне стиснення. Якщо такої можливості немає, то можна використовувати стандартний модуль

бітрікс “Компресія”. Однак використовуючи цей модуль потрібно бути обережним, його стиснення підтримується не всіма хостерами.

Мініфікація CSS і Javascript. Це - вирізання прогалин, перекладів рядків, зменшення локальних змінних і оптимізація коду (для js).

Перенос ресурсів на різні доменні імена, дозволяє обійти обмеження на 6 підключень.

Порядок підключення ресурсів на сторінці є дуже важливою частиною оптимізації завантаження CSS вгорі, Javascript - внизу. Обидва вони блокують рендеринг сторінки до завершення власного завантаження. Але якщо немає можливості прибрати css в низ сторінки: без нього веб-сторінка просто не буде відображена, то Javascript дозволяється повністю прибрати вниз або ж завантажувати асинхронно.

Для оптимізації front-end'a як чек-листа можна використовувати сторонні інструменти:

— Google PageSpeed Insights
— Audits в Chrome Developers Tools
— YSlow

Детальніше розглянемо Google PageSpeed Insights

PageSpeed Insights API

PageSpeed Insights API (PSI) дозволяє отримувати звіти про швидкість завантаження сторінок на мобільних пристроях і комп'ютерах, а також поради, як цю швидкість збільшити.

PSI надає як дані про те, наскільки швидко сторінка завантажувалася у справжніх користувачів, так і дані, отримані в результаті імітації процесу

завантаження. Оскільки імітація виконується в керованих умовах, з її допомогою зручно виявляти і усувати проблеми зі швидкістю, але є ризик втратити деякі з тих, які виникають в дійсності. Дані ж спостережень від користувачів відображають реальний стан справ, але набір доступних показників обмежений. Більш детальна інформація про ці два типи відомостей представлена на сторінці Що потрібно розуміти, працюючи з інструментами для оптимізації швидкості завантаження.

Оцінка швидкості завантаження

Вгорі звіту PSI показується загальна оцінка швидкості завантаження сторінки в балах. Ця оцінка розраховується за підсумками імітації завантаження за допомогою інструменту Lighthouse. Результат від 90 балів і вище вважається хорошим, від 50 до 90 - середнім. Якщо набрано менше 50 балів, значить сторінка завантажується повільно.

Після того як ви даєте PSI завдання проаналізувати сторінку за певним URL, виконується пошук відомостей про неї в звіті про зручність користування браузером Chrome. У звіт PSI включаються доступні дані за показниками першої відтворення контенту (FCP) і першої затримки введення (FID) для всього джерела або конкретної сторінки із зазначеним URL.

При аналізі зібраних у користувачів даних зі звіту про зручність користування браузером Chrome сервіс PSI розподіляє сторінки за трьома категоріями: з швидкої, звичайної і повільної завантаженням. Критерії класифікації вказані в таблиці нижче.

Табл.1.1.

Таблиця параметрів відповідності

	Высокая скорость	Средняя скорость	Низкая скорость
FCP	0–1000 мс	1000–2500 мс	Более 2500 мс
FID	0–50 мс	50–250 мс	Более 250 мс

В цілому приблизно у 10% сторінок спостерігається висока швидкість завантаження, у наступних 40% - середня, а у останніх 50% - низька. Числа округлені для простоти розуміння. Наведені в таблиці порогові значення однакові для мобільних пристроїв і комп'ютерів і введені з урахуванням особливостей людського сприйняття.

Розподіл значень FCP і FID, а також вбрання для кожного показника значення

У PSI показується розподіл значень FCP і FID, отриманих в результаті аналізу даних про певну сторінці або джерелі. Категорій при цьому теж три: "Швидко", "Середньо" та "Повільно". На діаграмі розподілу вони позначені зеленим, помаранчевим і червоним кольором відповідно. Наприклад, якщо в помаранчевої частини діаграми навпаки показника FCP зазначено частку 14%, значить 14% всіх наявних значень FCP знаходиться в діапазоні від 1000 до 2500 мс. Це агреговані дані про завантаження сторінки за минулі 30 днів.

Загальний результат

Швидкість завантаження сторінки оцінюється на основі значень обох показників:

— Висока - якщо і FCP, і FID високі.

— Низька - якщо FCP або FID низький.

— Середня - у всіх інших випадках.

Дані про фактичну швидкості завантаження в PSI оновлюються щодня і охоплюють останні 30 днів, в той час як дані звіту про зручність користування браузером Chrome, Популярні в BigQuery, оновлюються тільки раз на місяць.

Імітація завантаження сторінки

Інструмент PSI за допомогою технології Lighthouse отримує різні показники швидкості завантаження сторінки із зазначеним URL, такі як Перша отрисовка контенту, Час завантаження достатньої частини контенту, Індекс швидкості завантаження, Час закінчення роботи ЦП, Час завантаження для взаємодії і Приблизний час затримки при введенні.

За кожним показником надається окрема оцінка з позначкою у вигляді певного значка:

— Якщо показник високий, навпроти нього показується зелений кружок з галочкою.

— Якщо показник середній, навпроти нього показується помаранчевий гурток з буквою і.

— Якщо показник низький, навпроти нього показується червоний трикутник зі знаком оклику.

1.3. Медоти обробки данних використанні в розробці проекту

Big Data Processing - це набір методів та моделей програмування для доступу до широкомасштабних даних для отримання корисної інформації для підтримки та прийняття рішень [5]

Обробка великих даних передбачає дії, дуже подібні до обробки даних у середовищі транзакцій або сховища даних . На рисунку 1.9 показані різні етапи обробки великих даних; підхід до обробки великих даних такий:

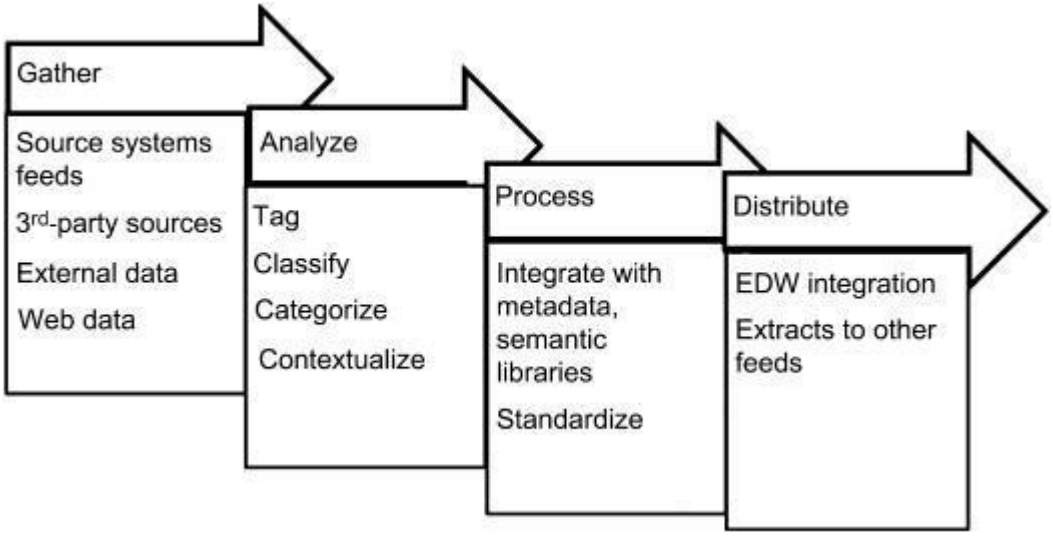


Рис. 1.9. Етапи обробки великих даних

●Зберіть дані.

●Проаналізуйте дані.

●Обробити дані.

●Поширюйте дані.

Хоча етапи аналогічні традиційній обробці даних, ключовими відмінностями є:

Дані спочатку аналізуються, а потім обробляються.

Стандартизація даних відбувається на етапі аналізу, що формує основу для етапу розподілу, де відбувається інтеграція сховища даних.

Якість даних не приділяється особливого значення, крім використання метаданих, основних даних та семантичних бібліотек для покращення та збагачення даних.

Дані готуються на етапі аналізу для подальшої обробки та інтеграції.

Етапи та їх діяльність детально описані у наступних розділах, включаючи використання метаданих, основних даних та процеси управління .

Дані збираються з різних джерел, включаючи системи реального часу, системи майже реального часу та пакетно-орієнтовані програми. Дані збираються та завантажуються до середовища зберігання даних, такого як Hadoop або NoSQL. Інший варіант - обробляти дані через платформу знань та зберігати результати, а не весь набір даних.

Етап аналізу - це етап виявлення даних для обробки великих даних та підготовки їх до інтеграції до структурованих аналітичних платформ або сховища даних. Етап аналізу складається з тегування, класифікації та категоризації даних, що дуже нагадує етап визначення моделі даних предметної області у сховищі даних.

Позначення тегів - поширена практика поширення Інтернету з 2003 року в Інтернеті. Тегування-це процес застосування терміну до неструктурованої частини інформації, яка надасть метаданим подібну атрибуцію до даних. Позначення тегами створює багатий неієрархічний набір даних, який можна використовувати для обробки даних за потоком на стадії процесу.

Класифікувати - неструктуровані дані надходять з кількох джерел і зберігаються в процесі збору. Класифікація допомагає згрупувати дані у тематично орієнтовані набори даних для зручності їх обробки. Наприклад, класифікація всіх даних клієнтів в одній групі допомагає оптимізувати обробку неструктурованих даних клієнтів.

Категоризувати - процес категоризації - це зовнішня організація даних з точки зору зберігання, де дані фізично групуються як за класифікацією, так і за типом даних. Категоризація буде корисною для управління життєвим циклом даних, оскільки дані зберігаються як модель одноразового запису на рівні зберігання. [5]

Обробка великих даних зазвичай виконується на великих кластерах товарних машин із загальним доступом. Одним з ключових уроків MapReduce є те, що необхідно розробити модель програмування, яка приховує складність базової системи, але забезпечує гнучкість, дозволяючи користувачам розширювати функціональні можливості для задоволення різноманітних обчислювальних вимог. Хоча додаток MapReduce порівняно з додатком MPI менш складний у створенні, він все одно може вимагати значних зусиль щодо кодування. У міру еволюції каркасів даних, зростає кількість API більш високого рівня, які покликані ще більше зменшити складності створення додатків з великою кількістю даних. Поточні фреймворки з великою кількістю даних, такі як Spark, дуже успішно зменшили необхідну кількість коду для створення конкретної програми. Майбутні API з використанням основних даних будуть продовжувати вдосконалюватися у чотирьох ключових областях; відкриття користувачам більш оптимальних процедур, що дозволяє прозорий доступ до різних джерел даних, використання графічних інтерфейсів

користувача (GUI) і дозволяє взаємодія між неоднорідними апаратними ресурсами.

Майбутні API більш високого рівня продовжуватимуть надавати фреймворкам з великою кількістю даних можливість відкривати оптимізовані процедури розробникам додатків, забезпечуючи підвищену продуктивність з мінімальними зусиллями з боку кінцевого користувача. Такі системи, як API Dataframe Spark, довели, що завдяки ретельному проектуванню високорівневий API може зменшити складність для користувачів, одночасно значно збільшивши продуктивність над API нижчого рівня.

Майбутнє застосування великих даних вимагатиме доступу до все більш різноманітних джерел даних. Майбутнім API потрібно буде приховати цю складність від кінцевого користувача та дозволити безперебійну інтеграцію різних джерел даних (структурованих та напів- чи неструктурованих даних) для зчитування з різних місць (HDFS, джерела потоків та бази даних).

Одним, відносно недослідженим, способом зменшення бар'єру входу до обчислень з великим об'ємом даних є створення графічного інтерфейсу, що дозволить користувачам без програмування та запитів запитів мати доступ до фреймворків, що інтенсивно передають дані. Використання графічного інтерфейсу також відкриває інші цікаві можливості, такі як взаємодія в режимі реального часу та візуалізація наборів даних.

API також потрібно буде продовжувати розвивати, щоб приховати складності все більш неоднорідного обладнання. Якщо співпроцесори будуть використовуватися на майбутніх машинах з великими даними, API -фреймворки, що інтенсивно передають дані, в ідеалі приховують це від кінцевого користувача. Користувачі повинні мати можливість написати код програми, а фреймворк вибере найбільш підходяще обладнання для його роботи. Це також

може включати переміщення всього або частини робочого навантаження в хмару за потреби.

Для системних адміністраторів розгортання інтенсивних для роботи фреймворків на комп'ютерному обладнанні все ще може бути складним процесом, особливо якщо потрібен великий стек. Майбутні дослідження потрібні для дослідження методів атомного розгортання сучасного стека великих даних на апаратному забезпеченні комп’ютера. Ці системи також повинні встановлювати та оптимізувати безліч параметрів конфігурації, які можуть мати великий вплив на продуктивність системи. Однією з ранніх спроб у цьому напрямку є Apache Ambari, хоча ще необхідно виконати додаткові роботи, такі як інтеграція системи з хмарною інфраструктурою. Чи могла система такого типу автоматично розгортати власний стек програмного забезпечення з великою кількістю даних у хмарі, коли локальний ресурс заповнився, і запускати програми в парі з локальним ресурсом?

1.4. Розробка моделі взаємодії компонентів

Актор веб-клієнта використовує певний веб-сайт для здійснення покупок в Інтернеті. Приклади використання найвищого рівня - це Перегляд товарів, Здійснення покупок та Реєстр клієнтів. Клієнт може використовувати варіант перегляду предметів як варіант використання верхнього рівня, якщо клієнт хоче лише знайти та побачити деякі товари. Цей варіант використання також може бути використаний як частина випадку використання "Зробити покупку". Випадок використання реєстру клієнтів дозволяє клієнту зареєструватися на веб-сайті, наприклад, отримати купони або бути запрошеним до приватних продажів. Зверніть увагу, що цей випадок використання Checkout включає

варіант використання, який не доступний сам по собі - оформлення замовлення є частиною здійснення покупки.

За винятком актора Веб-замовника, є ще кілька акторів, які будуть описані нижче з докладними прикладами використання**.**

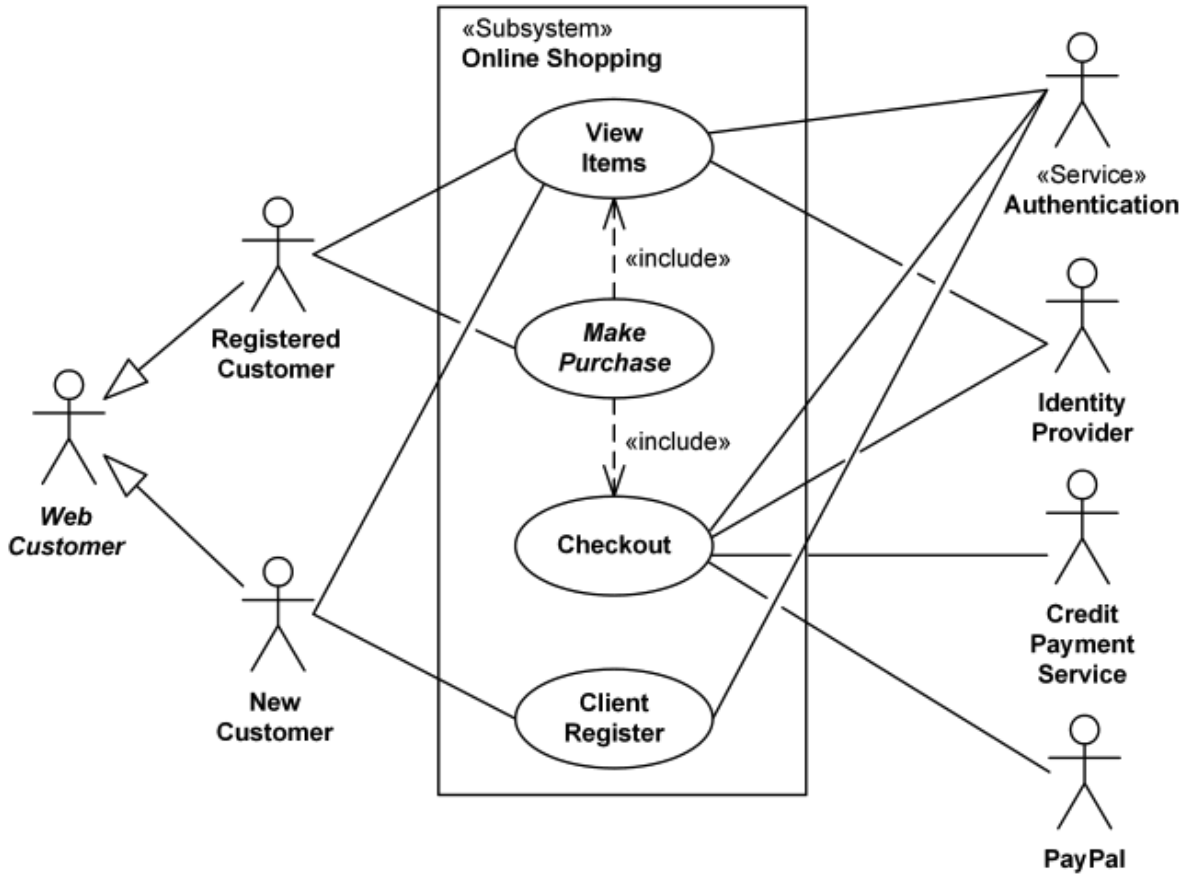


Рис. 1.10. Use case діаграма вищого рівня

Варіант використання "Перегляд товарів" поширюється на кілька додаткових випадків використання - клієнт може шукати товари, переглядати каталог, переглядати рекомендовані йому товари, додавати товари в кошик або список бажань. Усі ці випадки використання розширюють випадки використання, оскільки вони надають деякі додаткові функції, що дозволяють замовнику знайти товар.

Випадок використання автентифікації клієнта включений до перегляду рекомендованих предметів та додавання до списку бажань, оскільки обидва вимагають автентифікації клієнта. У той же час товар можна додати до кошика без аутентифікації користувача.

Рис.1.11. Приклад діаграми використання UML для Інтернет-магазину - перегляд випадку використання елементів.

Випадок використання Checkout включає кілька необхідних випадків використання. Веб-клієнт повинен бути автентифікований. Це можна зробити за допомогою сторінки входу користувача, файлу cookie для автентифікації користувача («Запам’ятати мене») або єдиного входу (SSO). Служба автентифікації веб-сайту використовується у всіх цих випадках використання,

тоді як SSO також вимагає участі зовнішнього постачальника ідентифікаційних даних.

Випадок використання Checkout також включає випадок використання платежу, який може бути здійснений за допомогою кредитної картки та зовнішньої служби оплати або через PayPal.

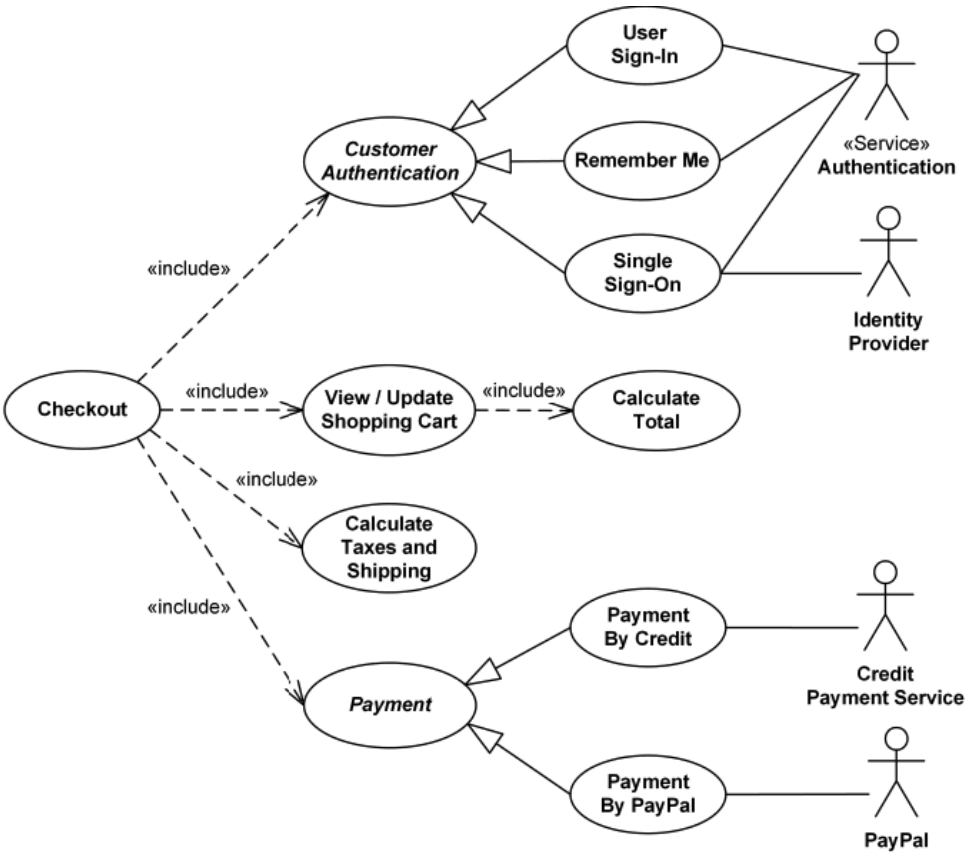


Рис.1.12. Приклад діаграми використання випадків використання UML

Система обробки кредитних карток (вона ж шлюз оплати кредитними картками) є предметом, тобто системою, що розробляється або розглядається. Основною дійовою особою системи є система обробки кредитних карток продавця. Продавець подає запит на операцію з кредитною карткою до шлюзу оплати кредитної картки від імені клієнта. Банк, який видав кредитну картку

клієнта, є суб'єктом, який може схвалити або відхилити операцію. Якщо транзакція затверджена, кошти будуть перераховані на банківський рахунок продавця.

Авторизація та фіксація випадків використання - найпоширеніший тип трансакції з кредитною карткою. Запитувана сума грошей повинна бути спочатку санкціонована Банком кредитних карток Клієнта, а якщо буде схвалена, надалі буде подана для розрахунку. Під час розрахунку кошти, затверджені для операції з кредитною карткою, вносяться на банківський рахунок продавця.

У деяких випадках вимагається лише авторизація, і транзакція не надсилається для розрахунку. У цьому випадку, як правило, якщо протягом певної кількості днів не вживаються подальші дії, термін дії дозволу закінчується. Продавці можуть подати цей запит, якщо хочуть перевірити наявність коштів на кредитній картці клієнта, якщо товар зараз відсутній на складі або якщо продавець хоче переглянути замовлення перед відправкою.

Приклад використання Capture (запит на отримання коштів, які раніше були санкціоновані) описує кілька сценаріїв, коли продавцеві потрібно виконати якусь раніше санкціоновану транзакцію - або подану через платіжний шлюз, або запитувану без використання системи, наприклад за допомогою голосової авторизації.

Розділ 2. Технічне завдання та створення сайту

2.1.Загальні поняття

Цей документ є офіційним, та показує як має працювати сайт з продажу текстильних виробів.

Структура інтернет-магазину

- Доставка и оплата
- Новинки
- Каталог
- Акции
- Fashion
- Sport
- Гарантія якості
- Оптовим покупцям
- Відгуки
- Контакти

Кафедра КІТ (47)				НАУ 21 40 27 000 ПЗ		
Виконав	Тертичний В.В			Технічне завдання та створення сайту	Літера	арк 49 / аркушів 21
Керівник	Віноградов М.А					
Консульт					УС-212М	122
Н.контрол	Райчев І Е					

Горизонтальне меню під фільтром:

1. Nike.
2. Adidas.
3. Puma.
4. Reebok.
5. H&M.

Підрозділи мають мати наступні фільтри:

— Розмір;

— Колір;

— Ціна;

— Виробик;

— Продавець;

— Сезон;

— Категорія.

2.1. Детальна структура

На головній сторінці - будуть знаходитися:

header сайту: Логотип + підпис, телефони, кнопку «замовити дзвінок», час роботи магазину, Кошик, пошук по сайту, посилання на сторінки в соціальних мережах головне меню, додаткове горизонтальне меню.

Тематична зона:

— Слайдер банерів, Новинки, Товари зі знижкою! Топ продажів, Оптовикам.

— Блок новин: назва, анотація, дата.

— Текст на головній.

— Блок переваг.

— Блок товарів з розпродажу: фото, ціна нова і стара ціна перекреслена, назва з посиланням для переходу до детального опису.

— Блок “Відгуки покупців” у вигляді відображення останніх відгуків клієнтів + посилання дивитися все відгуки.

— Блок з останніми статтями на сайті: назва, дата.

— Віджети соцмереж.

— Таблиця розмірів в сайдбарі у вигляді іконки, при натисканні на яку буде спливати скрипт.

— Онлайн консультант.

— Футер сайту: Дубляж кнопок головного меню, посилання на розділи, телефони, адреса, соцмережі, авторські права і розробник, лічильники відвідуваності.

— Розділ «Про нас» - загальна інформація про компанію з фото і відео.

— Розділ «Акції» - висновок товарів зі знижкою на сайті, помічених «галочкою» + іконка на фото.

— Розділ «Новинки» - виведення нових товарів на сайті, помічених «галочкою» + іконка на фото.

— Розділ «Умови покупки» - текстова сторінка.

— Розділ «Доставка» - текстова сторінка Види доставки: перерахувати.

— Розділ «Оплата» - текстова сторінка Види оплати: перерахувати.

— Розділ «Гарантія» - текстова сторінка. Вказати яка гарантія на вашу продукцію.

— Розділ «Оптовим покупцям» - текстова сторінка. Перерахувати знижки для оптовиків.

— Розділ «Відгуки» - форма на сайті з можливістю залишити відповідь адміном на відгук.

— Розділ «Новини» міститиме перелік новин, які будуть мати наступну структуру: назва новини, фото, текст і посилання для переходу на повний опис новини. Внутрішня сторінка міститиме фото, текст, кнопки «поділитися», блок з іншими новинами сайту.

— Розділ «Статті» міститиме перелік статей, який буде мати наступну структуру: назва статті, фото, текст і посилання для переходу на повний опис статті. Внутрішня сторінка міститиме фото, текст, кнопки «поділитися», блок з іншими статтями в розділі.

— Розділ «Контакти» міститиме загальну контактну інформацію, форму зворотного зв'язку, схему проїзду.

— Розділ продукції буде містити фільтр + товари з усіх його підрозділів.

— Підрозділ продукції буде містити фільтр + товари конкретного підрозділу.

— Товари, додані на сайті, будуть представлені у вигляді плитки. Кожна одиниця товару в даному розділі буде відображатися у вигляді блоку: артикулом, назва і фото з посиланням для переходу до детального опису товару, полотно, наявність, іконка з кол-вом відгуків до товару, вибором розміру, введенням кількості, ціною (нової і перекресленою якщо зі знижкою), введення примітки, кнопка «в корзину» для можливості оформлення замовлення. Знизу кожної сторінки підрозділу буде передбачений блок для додавання текстової інформації.

— Сторінка товару буде містити:

— Головне фото, додаткові фото, назву, артикул, вибір розміру, ціну (або перечеркнутую ціни, якщо задана), полотно, поле для введення бажаної кількості товару до придбання, кнопку купити, кнопку купити в один клік, Є в наявності / Немає в наявності , текстовий опис, блок з іконками з посиланнями на розділи: умови покупки, оплата, доставка, гарантія, кнопки соціальних мереж «поділитися», блок з текстовим описом, блок з відгуками про товар. Інші товари в розділі.

— Розділ «Кошик» (замовлення буде оформлятися в 1 крок) буде містити перелік замовлених товарів з назвою, артикулом, фотографіями, цінами і загальною сумою до оплати. Розміром знижки, інформації про доставку, Блоки для введення персональної інформації, вибору способу доставки і оплати. Особистий кабінет користувачів на сайті відсутній.

— В адміністративній панелі сайту буде передбачено ведення історії покупок з можливістю присвоєння статусу, пошуку товару по артикулу, пошуку товару за назвою.

— Купівля товарів на сайті Інтернет магазину буде здійснюватися без реєстрації (Швидке замовлення). Вся інформація про замовлення буде приходити Замовнику на електронну пошту.

2.2. SEO рішення

Функціонал сайту включатиме інтеграцію SEO модуля, який дозволяє вносити Meta-дані (title, description) для всіх сторінок сайту.

Функціонал сайту також передбачатиме установку ЧПУ модуля який дозволяє відображати адресу посилання на транслітерації.

2.3. Технічні характеристики програмних засобів

Реалізація програмної частини веб-системи буде здійснюватися із застосуванням CMS Bitrix, яка дозволить забезпечити високу швидкість роботи динамічних частин веб-системи. Як використовуваного сервера БД буде використовуватися сервер MySQL, що володіє достатнім швидкодією і стійкістю до відмов. В якості мови програмування буде використовуватись скриптова мова програмування PHP версії 7.2. ця версія має достатню швидкодію та надійність для використання в розробці проекту. Для реалізації серверу для постановки будемо використовувати Amazon Web Services (AWS).

Amazon Web Services або **AWS** є дочірньою компанією Amazon.com, що надає платформу хмарних обчислень в оренду приватним особам, компаніям та урядам на основі платної підписки. Існує і безкоштовна підписка, яка доступна протягом перших 12 місяців. Технологія дозволяє абонентам мати у своєму розпорядженні повноцінний віртуальний кластер комп'ютерів, який завжди доступний через Інтернет. Віртуальні комп'ютери AWS мають більшість атрибутів реального комп'ютера, включаючи апаратні пристрої (процесор, відеокарту, локальну та оперативну пам'ять, жорсткий диск або SSD-накопичувач); операційну систему на вибір; мережу; і попередньо встановлені прикладні програми, такі як веб-сервер, база даних, CRM і т. д.

Кожна система AWS також віртуалізує консольний ввід/вивід (клавіатура, дисплей і миша), що дозволяє користувачам AWS підключитися до своєї системи AWS за допомогою браузера. Браузер виступає як вікно у віртуальний комп'ютер, дозволяючи користувачу входити в систему, налаштовувати та використовувати свої віртуальні системи так само, як справжній, фізичний комп'ютер. Це дозволяє їм налаштувати систему так, щоб надавати інтернет-орієнтовані сервіси та послуги своїм клієнтам.

Технологія AWS базується на серверних кластерах (фермах), розташованих по всьому світі. Плата за користування базується на комбінації використання апаратних засобів/ОС/програмного забезпечення/мережевих функцій, вибраних користувачем, а також вимог до доступності, надлишковості (redundancy), безпеки та додаткових параметрів. Виходячи з того, що користувач потребує і оплачує, він може зарезервувати один віртуальний комп'ютер (VM), кластер віртуальних комп'ютерів (VM Cluster), фізичний (реальний) комп'ютер (Server), призначений для його виняткового використання, або навіть кластер фізичних комп'ютерів (Server Cluster). Компанія Amazon зобов'язується керувати та оновлювати програмне та апаратне забезпечення для підтримання необхідних стандартів безпеки. [6]

2.4. Розробка файлової структури проекту

Имя	Размер файла	Изменен	Тип	Права на доступ сервера	Права на доступ продукта
..					
.git		04.11.2019 16:15:52	Папка	755 th_admin psacln	Полный доступ
about		16.10.2019 18:32:54	Папка	755 th_admin psacln	Полный доступ
auth		16.10.2019 18:32:54	Папка	755 th_admin psacln	Полный доступ
bitrix		07.04.2020 00:59:30	Папка	755 th_admin psacln	Полный доступ
catalog		16.10.2019 18:29:34	Папка	755 th_admin psacln	Полный доступ
contacts		16.10.2019 18:29:34	Папка	755 th_admin psacln	Полный доступ
delivery		16.10.2019 18:32:54	Папка	755 th_admin psacln	Полный доступ
desktop_app		16.10.2019 18:29:34	Папка	755 th_admin psacln	Полный доступ
local		16.10.2019 18:30:50	Папка	755 th_admin psacln	Полный доступ
news		16.10.2019 18:29:34	Папка	755 th_admin psacln	Полный доступ
payment		16.10.2019 18:32:54	Папка	755 th_admin psacln	Полный доступ
personal		16.10.2019 18:30:20	Папка	755 th_admin psacln	Полный доступ
search		16.10.2019 18:32:54	Папка	755 th_admin psacln	Полный доступ
store-reviews		16.10.2019 18:32:54	Папка	755 th_admin psacln	Полный доступ
upload		31.03.2020 14:16:22	Папка	755 th_admin psacln	Полный доступ

Рис. 2.1. Файлова структура проекту

Детально розберемо кожну директорії та підв’яжемо під неї функціональну характеристику на сайті.

.**git** - **Git** - це безкоштовна та з відкритим кодом розповсюджена система управління версіями, призначена для швидкого та ефективного управління всіма проектами від маленьких до дуже великих проектів.

Git легко засвоїти та має блискавичну швидкість. Він випереджає такі засоби SCM, як Subversion, CVS, Perforce та ClearCase з такими функціями, як розгалужене по гілкам місцеве розташування , та безліч робочих процесів.

about/ - сторінка детальної інформації про компанію.

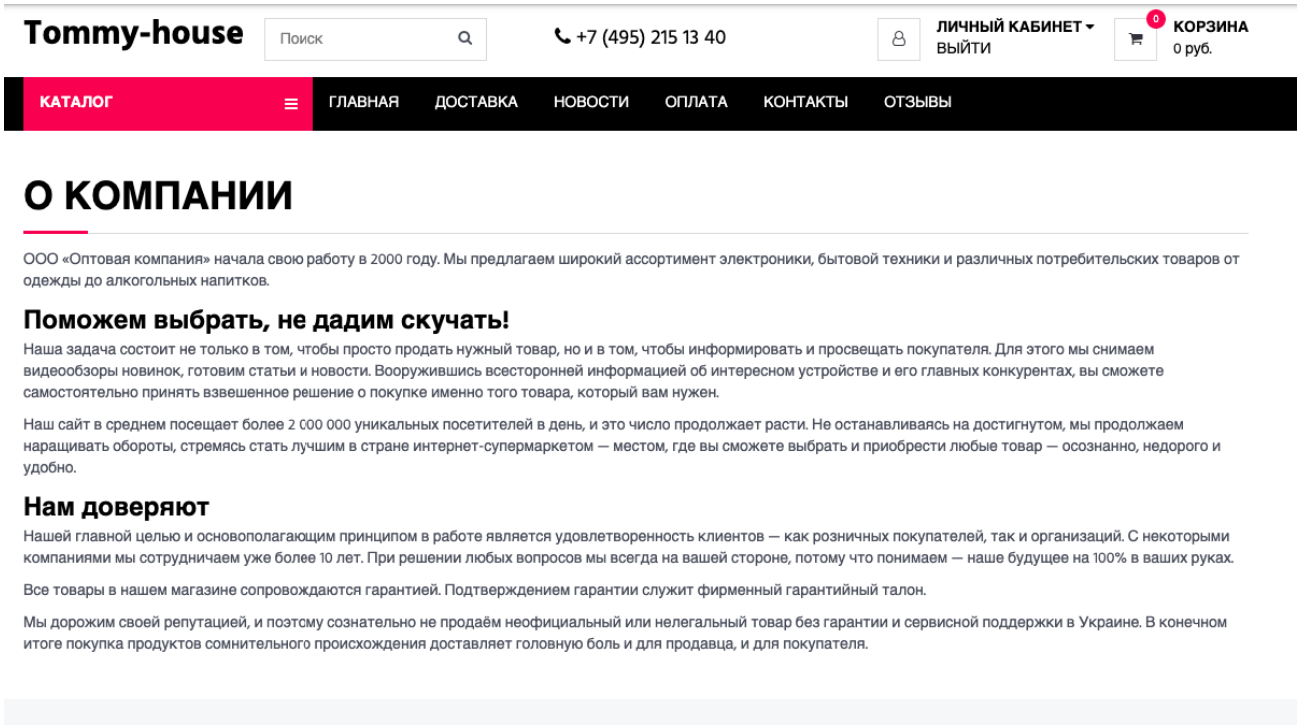


Рис. 2.2. Сторінка “Про компанію”

auth/ - ця директорія відповідає для реєстрацію та авторизацію на сайті, використовуючи стандартний спосіб авторизації в ядрі CMS 1c-Bitrix

створюється сесія користувача та з'являється можливість використовувати особистий кабінет, редагувати інформацію про себе та створювати замовлення.

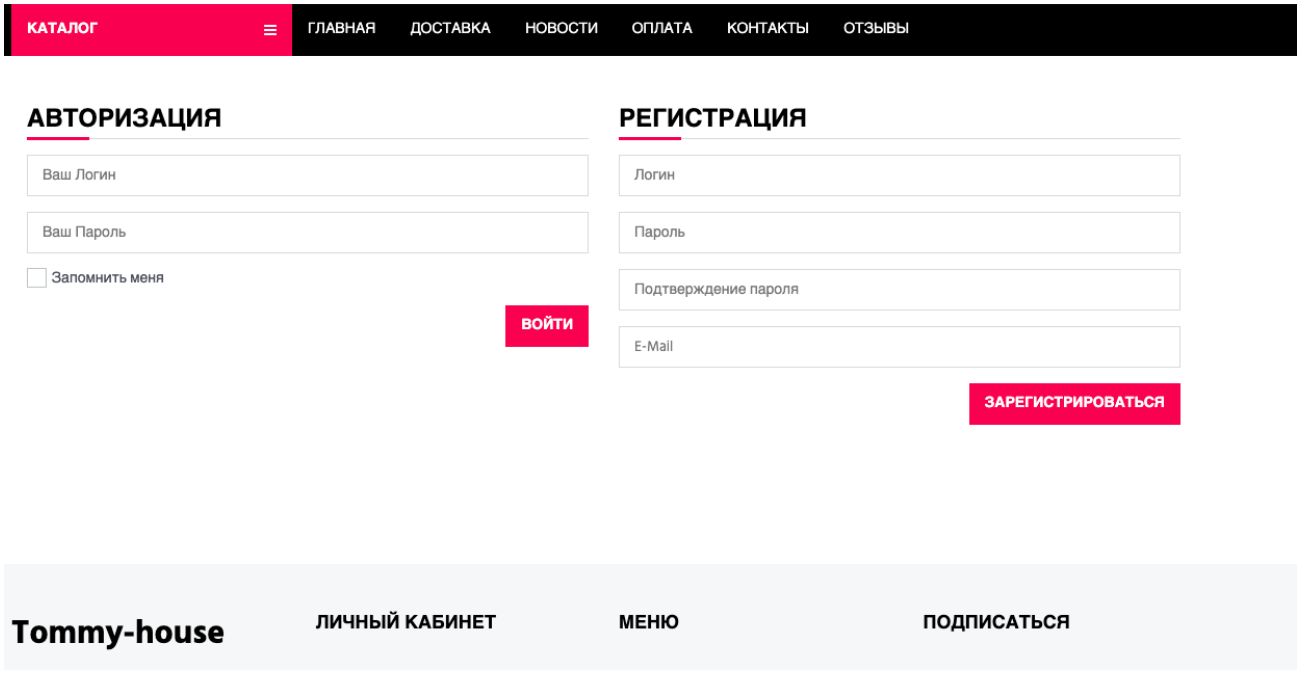


Рис. 2.3. Сторінка “Авторизація та реєстрація”

birtix/ - в цій папці знаходиться ядро CMS 1c-bitrix, в ній реалізована вся адміністративна частина сайту.

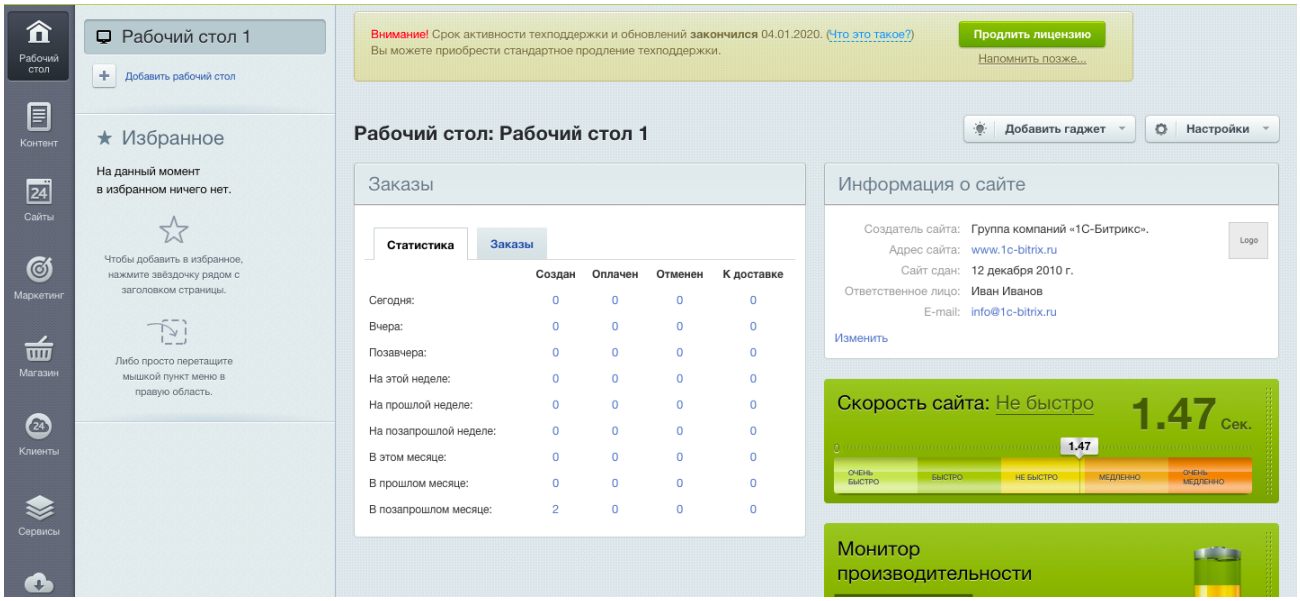


Рис. 2.4. Сторінка “Адміністратор”

catalog/ - директорія використовуються для виводу списка товарів та детальної сторінки товарів використовуючи маршрутизацію в файлі urlrewrite.php. Це означає, що при відкриті сторінки catalog/ в браузері, буде виконуватися машрутизація та підключення bitrix компоненту bitrix.catalog. Функціональність компонента об'єднує можливості декількох односторінкових компонентів: фільтру, компонента порівняння, виведення елементів розділу, пов'язаних елементів та ін. Розташувавши комплексний компонент на сторінці, можна отримати повнофункціональний каталог. Компонент є стандартним і входить в дистрибутив модуля.

```
2 =>
array (
  'CONDITION' => '#^/catalog/#',
  'RULE' => '',
  'ID' => 'bitrix:catalog',
  'PATH' => '/catalog/index.php',
  'SORT' => 100,
),
```

Рис. 2.5. Сторінка “Каталог товарів”

delivery/ - директорія використовуються для виводу статичної сторінки з інформацією про те, як здійснюється доставка, яка ціна доставки, терміни, та регіони в які вона здійснюється.

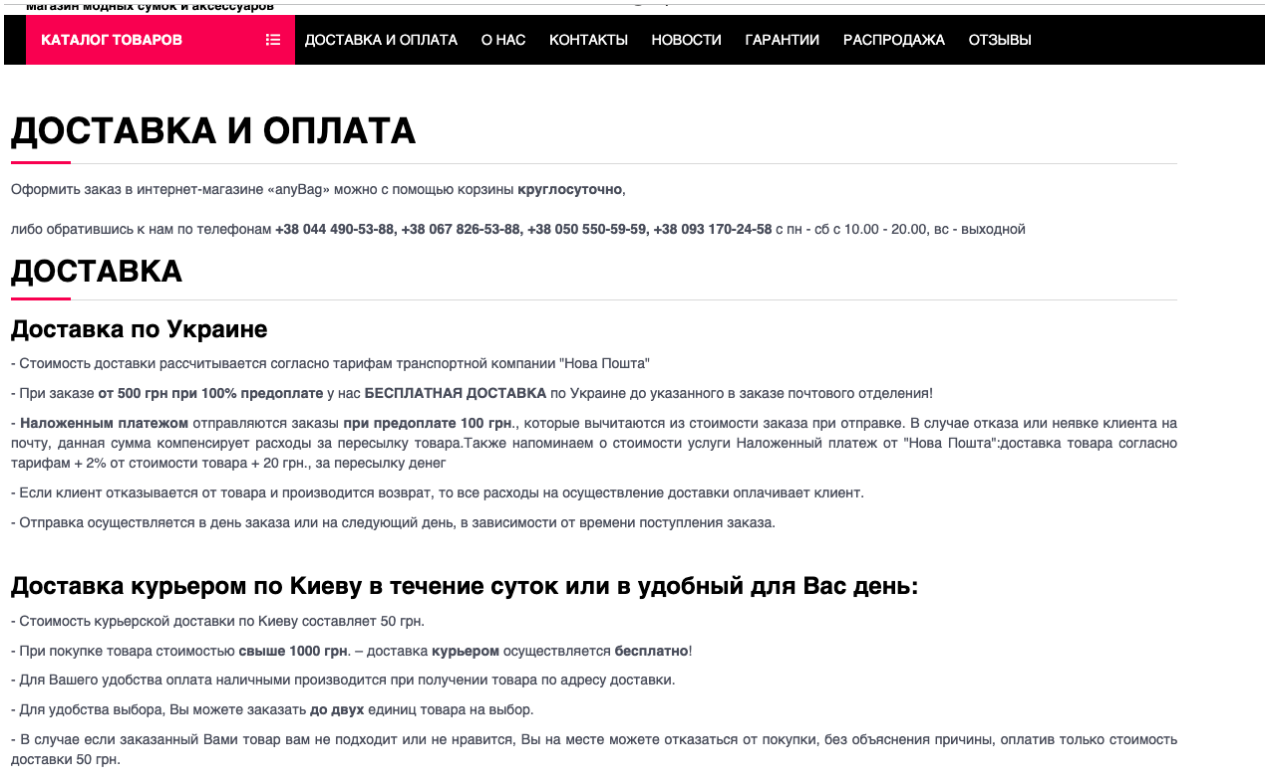


Рис. 2.6. Сторінка “Доставка і оплата”

contacts/ - директорія використовуються для виводу статичної інформації про способи комунікації з менеджерами інтернет магазину.

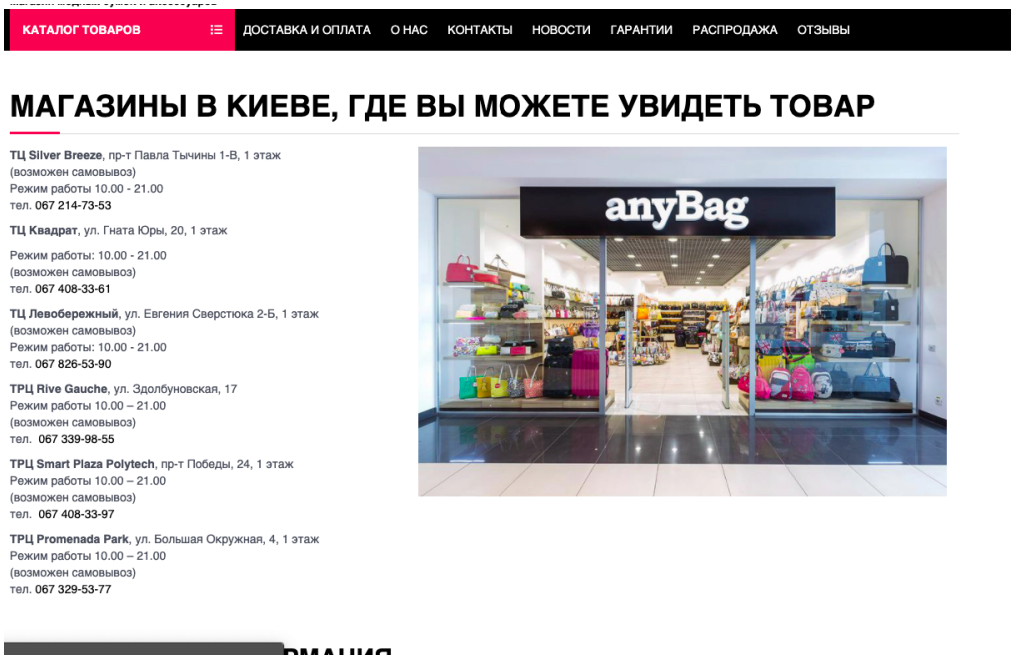


Рис. 2.7. Сторінка “Контакти”

local/ - директорія в якій зібрані всі кастомні файли, що використовуються для доробки функціоналу який не входить в стандартну збірку дистрибутиву 1C-Bitrix, завдяки таку підходу в доопрацюванні продукту, зберігається можливість легкого оновлення стандартних компонентів і модулів в системі, а також зберігається можливість офіційної підтримки від компанії 1C-Bitrix.

news/ - директорія служить для виводу інформації про новин про акції та оновлення в інтернет магазині.

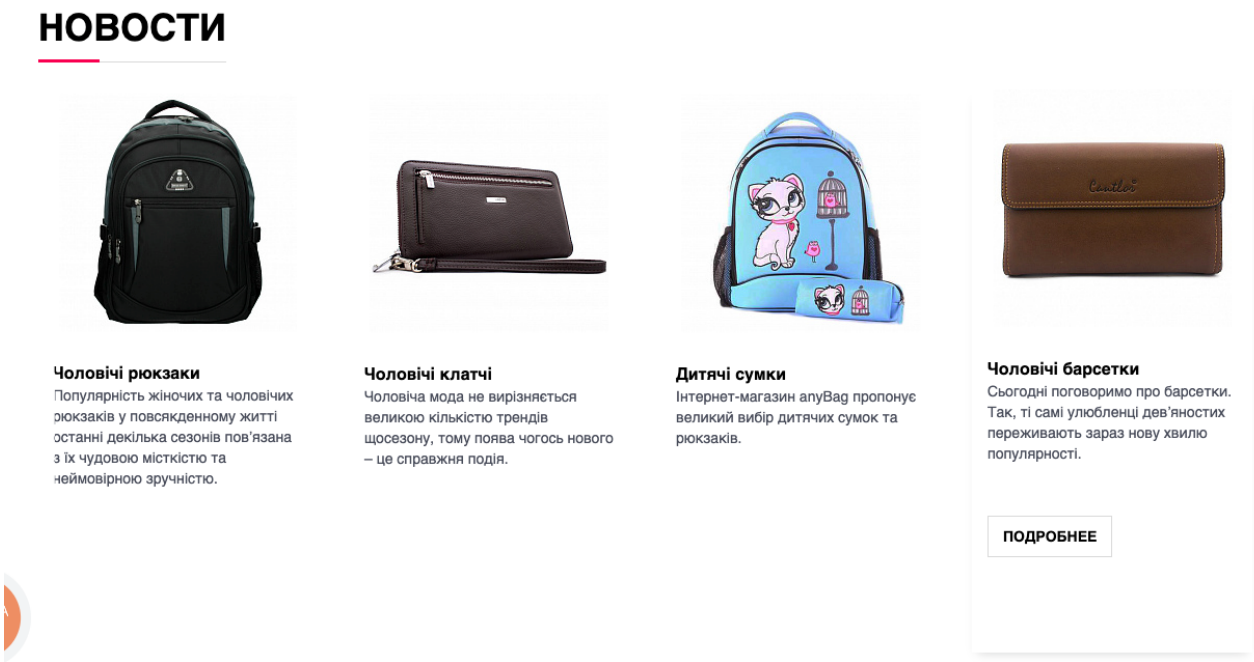


Рис. 2.8. Сторінка “Новини”

personal/ - сторінка особистого кабінету, на ній можно ввести дані про себе, потім ці дані підтягнуться в спеціальні поля при оформлені замовлення клієнтом.

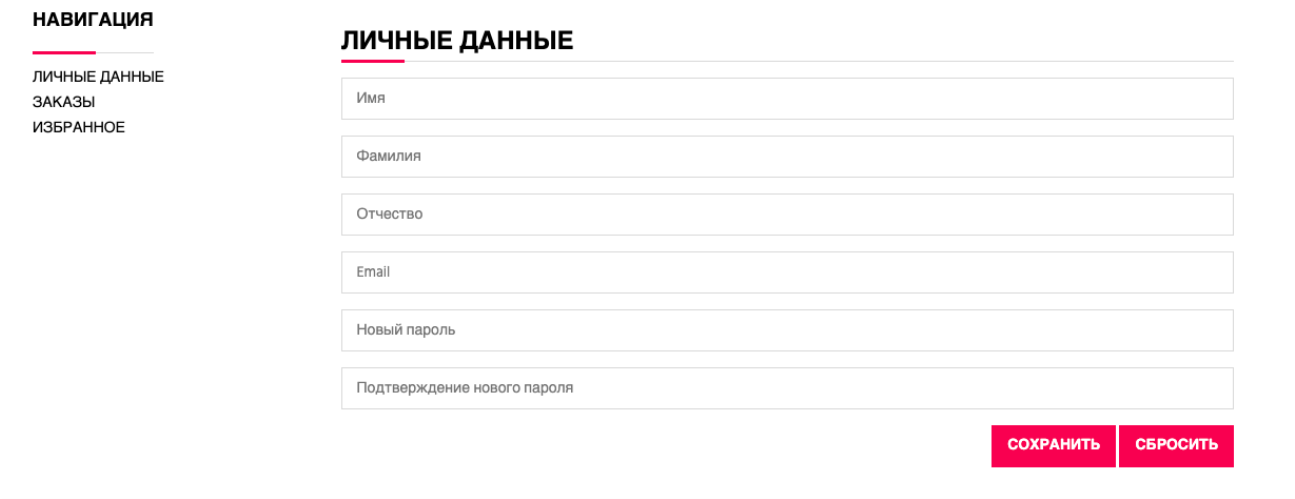


Рис. 2.9. Сторінка “Особисті дані”

personal/order/list - сторінка служить для відображення історії заказів, на ній є фільтр по відміненним замовленням та існує можливість оплатити замовлення.

Рис. 2.10. Сторінка “Історія замовлень”

2.5. Маршрутизація та обробка сторінок

Правила будування URL є одними з найважливішими для успішної SEO оптимізації. Правильно побудовані посилання можуть підняти вас в пошуковій видачі на перші місця. В цій частині ми розглянемо які існують правила створення URL та як це реалізовано в даному проекті.

Що таке URL?

URL - позначає Uniform Resource Locator. URL це лише адреса, який виданий унікальному ресурсу в інтернеті. У теорії, кожен коректний URL веде на унікальний ресурс. Такими ресурсами можуть бути HTML-сторінка, CSS-файл, зображення і т.д. На практиці, існують деякі винятки, коли, наприклад, URL веде на ресурс, який більше не існує або який був переміщений. Оскільки ресурс, доступний по URL, а також сам URL обробляються веб-сервером, його власник повинен уважно стежити за розміщеними ресурсами і пов'язаними з ними URL. (developer.mozilla.org).

Обробка адресів (*UrlRewrite*) використовується,щоб скрипт сторінки міг відображатися і по своєму фізичному адресу і по будь-якому іншому, вказаному в правилі, адресу. Наприклад, можна налаштувати маршрутизацію адресів , щоб скрипт у файлі /test/test.php , що відповідає за адресу /test/test.php?id=20 давав відповідь також по адресу /store/20.php. Адрес, за яким буде відображатися скрипт, може не існувати на сервері в файловій структурі. Якщо такий адрес фізично існує, то буде викликаний скрипт за цією адресою. Правила системи обробки адресів в даному випадку виконуватись не буде.

Правила маршрутизації налаштовуються і зберігаються в кореневій директорії сайту використовуючи файл urlrewrite.php . В файлі знаходиться масив **$arRewrite**. Файл **urlrewrite.php** виглядає наступним чином:

```
<?
$arRewrite = array (
    array (
        "CONDITION" => "#^/store/#" ,
        "RULE" => "" ,
        "ID" => "bitrix:store" ,
        "PATH" => "/max/store /index.php" ,
    ),
    array (
        "CONDITION" => "#^/blog/#" ,
        "ID" => "bitrix:blog" ,
        "PATH" => "/blog/index.php" ,
    ),
    array (
        "CONDITION" => "#^/blog/([0-9]+)/([0-9]+)/#" ,
        "RULE" => "mode=write&CID=$1&GID=$2" ,
        "ID" => "bitrix:blog" ,
        "PATH" => "/blog/index.php" ,
    )
);
?>
```

Кожне правило маршрутизації має бути унікальним в рамках одного сайту. Умови до виконання реалізуються в ключі масиву "**CONDITION** " і може містити шаблон регулярного вираз.

Регулярні вирази (що позначаються англійською як RegEx або як regex) є інструментальним засобом, який застосовується для різних варіантів вивчення та обробки тексту: пошуку, перевірки, пошуку та заміни того чи іншого елемента, що складається з букв або цифр (або будь-яких інших символів, в в тому числі спеціальних символів та символів пунктуації). Спочатку регулярні

висловлювання прийшли у світ програмування з-поміж наукових досліджень, які проводилися в 50-ті роки в галузі математики. [7]

Наприклад, умова:

"CONDITION" => "#^/blog/([a-zA-Z]+)/([0-9]+)/#"

показує, що правило повинно виконуватись для всіх адрес, які починаються з підстроку виду:

/blog/<string>/<number>/

Правило також може містити фізичну адресу існуючого скрипта, що підключиться в разі виконання всіх умов. Дана адреса реалізується в ключі масиву "**PATH**". Наприклад, якщо в системі обробки адресів зарезервовано правило:

```
масив ( "CONDITION" => "#^/store/#" ,
    "PATH" => "/catalog/index.php" )
```

і користувач запитав сторінку: /store/20.php яка фізично не існує, то система обробки адресів підключає скрипт: /catalog/index.php

2.6. Структура таблиць в БД

По стандарту в системі 1С-bitrix використовується вільна реляційна система управління базами даних MySql.

Важливими перевагами пакету MySQL є многопоточність, одночасна підтримка масиву запитів, можливість реалізації зв'язків з приєднанням багатьох даних за один прохід, записи фіксованої і змінної довжини,в комплекті з вихідним файлом використовується ODBC драйвер, маштабована система привілеїв і паролів, до 16 ключів в таблиці, кожен ключ може мати до 15 полів. Реалізована підтримка ключових полів і спеціальних полів в використовуючи оператор CREATE, інтерфейс з мовами C і php. Заснована на потоках, швидка система пам'яті, утиліта перевірки і ремонту таблиці, всі дані зберігаються в форматі ISO8859_1. Всі операції роботи з рядками не звертають уваги на регістр символів в оброблюваних рядках, псевдоніми застосовні як до таблиць, так і до окремих колонках у таблиці, все поля мають значення за замовчуванням. INSERT можна використовувати на будь-якому підмножині полів. Легкість керування таблицею, включаючи додавання та видалення ключів і полів.

Можна виконувати команди SQL безпосередньо з командного рядка системи unix або з інтерактивного режиму MySQL. СУБД MySQL має бібліотеку з API. Її можна використовувати для запитів до бази даних, вставки даних, створення таблиць і т.п.

Також доступний 32-бітний ODBC драйвер для MySQL. Він дозволяє запитувати і отримувати дані з інших джерел з підтримкою ODBC.

Крім технічних подробиць можна додати, що MySQL працює як на Unix, так і на платформі Windows, він дуже простий і зручний в роботі.

І щоб підвести підсумок, хотілося б розповісти одну історію. Однією невеликій фірмі потрібно зробити на сайті базу даних по товарах. Людина, якій доручили зробити базу даних, створив її на Delphi. Так ось, коли прийшов час розміщувати базу в мережі, виявився дуже неприємний факт. Все ISP працюють на Unix, відповідно для розміщення подібної бази потрібно ставити у провайдера свій комп'ютер і платити, в середньому, близько 50 у.о. за хостинг. У разі ж якщо СУБД зроблена під MySQL, то платити доведеться лише за мегабайти. [8]

Детально розглянемо структуру таблиць які потрібні для реалізації дипломного проекту.

Таблиця користувачів `b_user` містить основні дані про користувача сайту: ідентифікатор, e-mail, логін, пароль, хеш контрольного слова, дата реєстрації та додаткові встановлені поля особистих і робочих даних.

Структура таблиці `b_user`:

`LOGIN` - логін користувача,

`PASSWORD` - хеш пароля користувача,

`CHECKWORD` - хеш контрольного слова,

`ACTIVE` - активність користувача ('Y', 'N'),

`NAME` - ім'я користувача,

`LAST_NAME` прізвище користувача,

`EMAIL` електронна адреса користувача,

`LAST_LOGIN` - дата останнього авторизованого входу,

`DATE_REGISTER` - дата реєстрації,

`LID` прив'язка до сайту,

`PERSONAL_* - поля персональних даних

`WORK_*` - поля даних пов'язаних з роботою,

`ADMIN_NOTES` - коментар адміну,

`PERSONAL_BIRTHDAY` - дата народження,

`EXTERNAL_AUTH_ID` - ідентифікатор зовнішнього сервісу авторизації,

`SECOND_NAME` - по-батькові,

`CONFIRM_CODE` - код підтвердження при відновленні пароля,

`LOGIN_ATTEMPTS` - кількість спроб авторизації,

`LAST_ACTIVITY_DATE` - дата останньої активності.

Таблиця товарів `b_catalog_product` має такі поля:

ID – ідентифікатор

QUANTITY – кількість товару

QUANTITY_TRACE – відстеження кількості товару

WEIGHT – вага

TIMESTAMP_X – час створення

PRICE_TYPE – тип ціни

RECUR_SCHEME_LENGTH – тривалість рекурсивної оплати

RECUR_SCHEME_TYPE – тип рекурсивної оплати

TRIAL_PRICE_ID – ціна пробного періоду

WITHOUT_ORDER – без замовлення

SELECT_BEST_PRICE – вибирати найкращу ціну

VAT_ID – ідентифікатор ставки ПДВ

VAT_INCLUDED – чи включено ПДВ у ціну

Таблиця замовлень `b_sale_order` має такі поля:

ID - ідентифікатор

LID - сайт

PERSON_TYPE_ID - тип покупайся (юр. лице чи фіз. лице)

PAYED - оплачений (true/false)

DATE_PAYED - дата оплати

EMP_PAYED_ID - номер документа оплати

CANCELED - відмінене замовлення (true/false)

DATE_CANCELED - дата відміни

EMP_CANCELED_ID - номер документа відміни

REASON_CANCELED - причина відміни

STATUS_ID - статус замовлення

DATE_STATUS - дата останньої зміни статуса

EMP_STATUS_ID - номер документа статуса

PRICE_DELIVERY - ціна доставки

PRICE_PAYMENT - ціна до оплати

ALLOW_DELIVERY - дозволена доставка (true/false)

DATE_ALLOW_DELIVERY - дата дозволу доставки

DEDUCTED - відвантажений (true/false)

DATE_DEDUCTED - дата відвантаження

EMP_DEDUCTED_ID - номер документу відвантеження

REASON_UNDO_DEDUCTED - причина отгрузки

MARKED - заблокований

DATE_MARKED - дата блокування

EMP_MARKED_ID - номер блокування

REASON_MARKED - причина блокування

RESERVED - зарезервована

PRICE - ціна замовлення

CURRENCY - валюта замовлення

DISCOUNT_VALUE - знижка

USER_ID - ID користувача в таблиці (b_user)

PAY_SYSTEM_ID - платіжна система

DELIVERY_ID - система доставки

DATE_INSERT - дата створення замовлення

DATE_UPDATE - дата оновлення

USER_DESCRIPTION - коментар користувача

ADDITIONAL_INFO - додаткова інформація

PS_STATUS - статус платіжної системи

PS_STATUS_CODE - код статусу платіжної системи

PS_STATUS_DESCRIPTION - опис статусу платіжної системи

PS_SUM - сума до сплати

PS_CURRENCY - валюта платіжної системи

COMMENTS - коментар менеджера

TAX_VALUE - ставка податків

SUM_PAID - сплачена сума

CREATED_BY - ким створено

DATE_BILL - дата оплати рахунку

ACCOUNT_NUMBER - номер замовлення

TRACKING_NUMBER - ТТН

XML_ID - зовнішній код замовлення

VERSION_1C - версія 1С

VERSION - кількість оновлень с 1С

Розділ 3. Тестування та SEO аналіз

Тестування – є найважливішим етапом циклу розробки програмного продукту. Основна ціль тестування – пошук багів (помилок).

Основними принципами тестування - є кроки які демонструють користувачеві, наскільки проект є корисним, правильним та логічно побудованим, чи добре сприймається користувачем і чи правильно робить весь реалізований функціонал додатку.

Основні етапи тестування діляться на 4 рівне важливих пунктів:

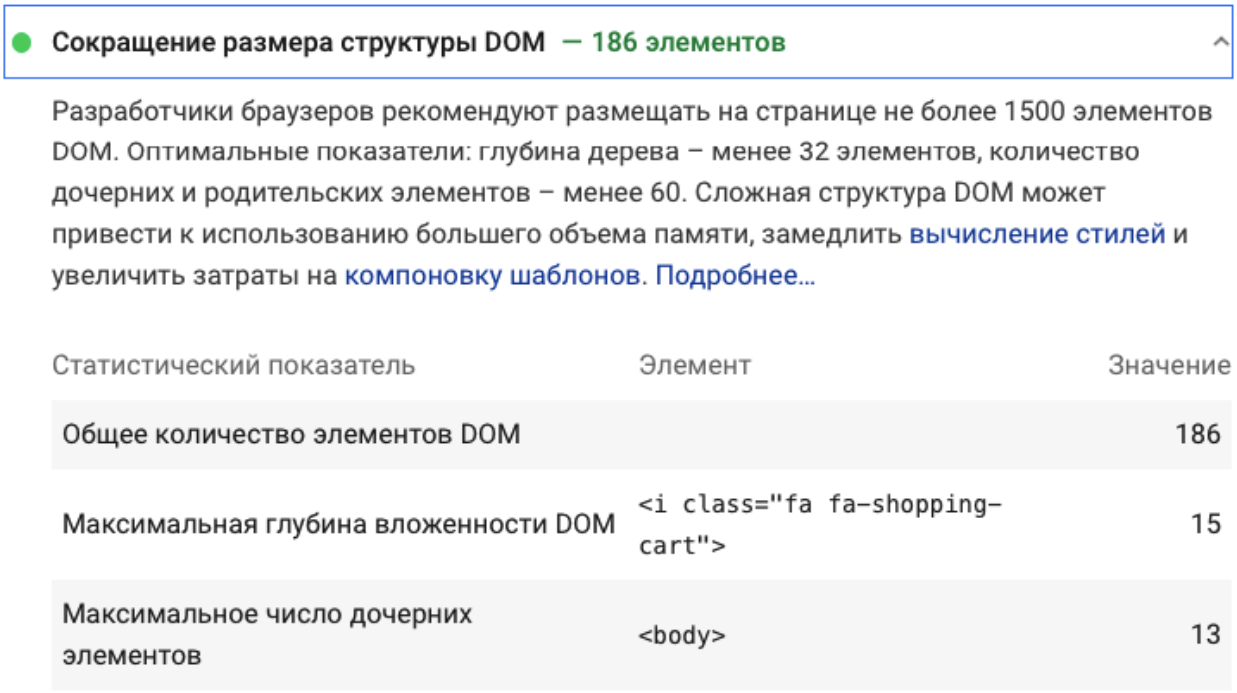
1. Функціональне тестування;
2. Тестування юзабіліті;
3. Тестування продуктивності;
4. Тестування безпеки;
5. Роздивимось кожний етап тестування детальніше.

3.1. Функціональне тестування

Головна ціль цього тестування – перевірити всі функції на сайті та коректність їх виконання, це потрібно робити саме людині.

3.2. Тестування юзабіліті

Ключова частина даного етапу тестування – це зовнішній вигляд сайту, та як зручно користувачу знайти інформацію яка йому потрібна, для цього використовуємо сервіс https://developers.google.com/speed/pagespeed/insights/ По деякий пунктам можна зрозуміти, що DOM елементи не перевантажені, на сторінці не знаходиться багато елементів, та користувачу легко знайти потрібну інформацію.[12]



Сокращение размера структуры DOM — 186 элементов

Разработчики браузеров рекомендуют размещать на странице не более 1500 элементов DOM. Оптимальные показатели: глубина дерева – менее 32 элементов, количество дочерних и родительских элементов – менее 60. Сложная структура DOM может привести к использованию большего объема памяти, замедлить вычисление стилей и увеличить затраты на компоновку шаблонов. Подробнее...

Статистический показатель	Элемент	Значение
Общее количество элементов DOM		186
Максимальная глубина вложенности DOM	`<i class="fa fa-shopping-cart">`	15
Максимальное число дочерних элементов	`<body>`	13

Рис. 3.1. Оптимізація DOM

3.3. Тестування продуктивності

Основною метою даного типу тестування – показати як буде працювати сайт, якщо на нього одночасно зайдуть багато користувачів. Це робиться автоматизовано спеціальними програмами.

Тим самим результат дає те, чи зміг наш проект витримати, наприклад, 100 користувачів, які одночасно купували товар або авторизувалися на сайті, відповідь показує, чи реально витримати сайт таке навантаження.

Даний вид тестування можна виконати використовуючи сервіс https://loaddy.com/ Потрібно вставити адресу сайту та почекати 1.5 хв

Після очікування отримуємо детальний результат тестування продуктиності, проаналізувавши який може зробити висновки

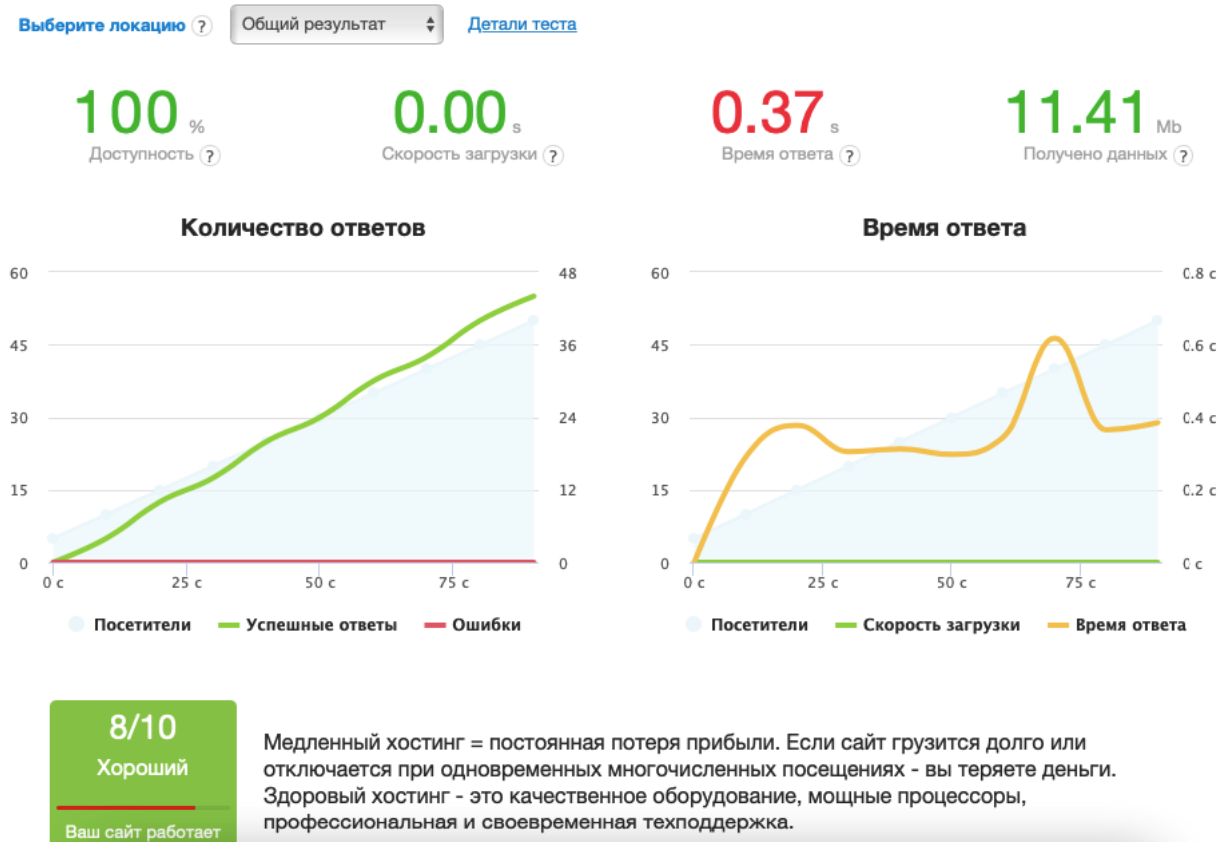


Рис 3.2. Тестування продуктивності

Навантаження подавалось в арифметичній прогресії, поступово, що дає можливість зрозуміти, на який кількостях віртуальний користувачі відбувалось просідання трафіку. Як можна бачити по графікам час відгуку сторінки коливався від 0.5с до 0.75с, що є чудовим результатом, та показує, що сайт може

одночасно обслуговувати 1000 користувачі без «просідань трафіку».

Ще одним способом тестування продуктивності є сервіс https://developers.google.com/speed/pagespeed/ , підставивши необхідну адресу для тестування ми можемо побачити результати відповіді сервера та оцінку продуктивності.

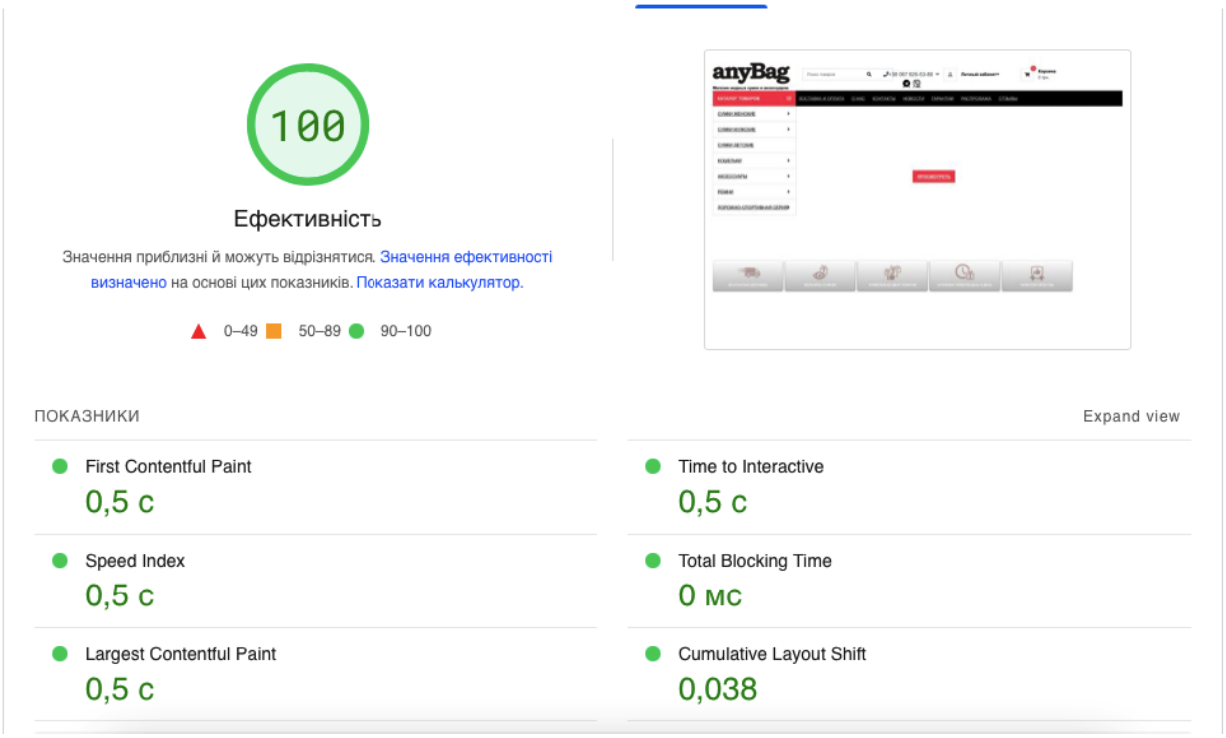


Рис 3.3. Оцінка продуктивності

Також можна побачити по яким критеріям відбувається оцінка продуктивності

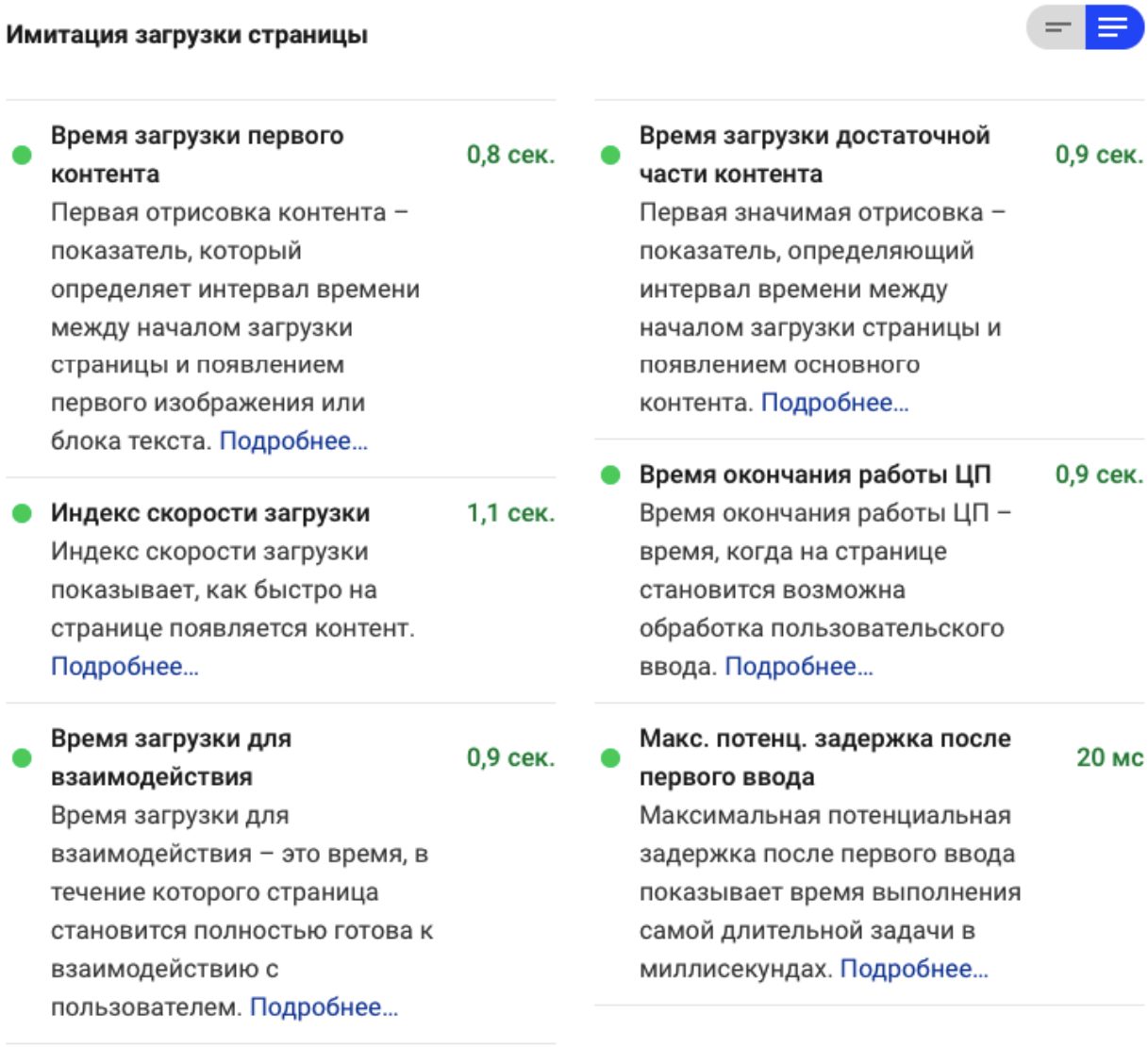


Рис. 3.4. Критерії оцінювання продуктивності

3.4. Тестування безпеки

Безпекове тестування проводитимо за допомогою сервісу PentestTools WebsiteScanner. Ввівши url свого сайту в input, отримуємо детальний звіт про безпековий стан сайту.

Рис. 3.5. Аудит безпеки

Табл. 3.1

Вразливість cookie

Cookie Name	URL	Evidence
PHPSESS ID	https://anybag.ua	Set-Cookie: PHPSESSID=TaEpYMe2wXPFL4rnsZ7QE0U06hj87jL3; path=/; HttpOnly, new=1; expires=Sun, 28-Nov-2021

Опис ризику:

Оскільки для файлу cookie не встановлено прапорець Secure, браузер надішле його через незашифрований канал (звичайний HTTP), якщо буде зроблений такий запит. Таким чином, існує ризик того, що зловмисник перехопить зв'язок із відкритим текстом між браузером і сервером і вкраде файл cookie користувача. Якщо це файл cookie сеансу, зловмисник може отримати несанкціонований доступ до веб-сеансу жертви.

Рекомендація:

Коли файл cookie містить конфіденційну інформацію або є маркером сеансу, його завжди слід передавати за допомогою зашифрованого каналу. Переконайтеся, що для файлів cookie, які містять таку конфіденційну

інформацію, встановлено прапорець безпеки. Знайдені вразливості серверного програмного забезпечення:

Табл. 3.2

Вразливість серверного програмного забезпечення

CVSS	CVE	Summary	Affected software
6.4	CVE-2020-7069	У версіях PHP 7.2.x нижче 7.2.34, 7.3.x нижче 7.3.23 і 7.4.x нижче 7.4.11, коли режим AES-CCM використовується з функцією openssl_encrypt() з 12 байтами IV, лише перші 7 байт IV фактично використовується. Це може призвести як до зниження безпеки, так і до неправильного шифрування даних.	PHP 7.2.30
5	CVE-2018-19935	ext/imap/php_imap.c у PHP 5.x і 7.x до 7.3.0 дозволяє віддаленим зловмисникам викликати відмову в обслуговуванні (розіменування покажчика NULL і збій програми) за допомогою порожнього рядка в аргументі повідомлення функції imap_mail.	PHP 7.2.30
5	CVE-2019-11048	У версіях PHP 7.2.x нижче 7.2.31, 7.3.x нижче 7.3.18 і 7.4.x нижче 7.4.6, коли завантаження файлів HTTP дозволено, надання занадто довгих імен файлів або імен полів може призвести до спроби механізму PHP спробувати виділити занадто великі пам’яті, досягти ліміту пам’яті та припинити обробку запиту, не очищаючи тимчасові файли, створені запитом на завантаження. Це потенційно може призвести до накопичення неочищених тимчасових файлів, які вичерпують дисковий простір на цільовому сервері.	PHP 7.2.30
5	CVE-2020-7070	У версіях PHP 7.2.x нижче 7.2.34, 7.3.x нижче 7.3.23 і 7.4.x нижче 7.4.11, коли PHP обробляє вхідні значення cookie HTTP, назви файлів cookie декодуються url. Це може призвести до того, що файли cookie з такими префіксами, як __Host, плутають із файлами cookie, які декодують такий префікс, що призведе до того, що зловмисник зможе підробити файл cookie, який має бути безпечним. Дивіться також CVE-2020-8184 для отримання додаткової інформації.	PHP 7.2.30
4.3	CVE-2015-9251	jQuery до версії 3.0.0 уразливий до атак міжсайтових сценаріїв (XSS), коли міждоменний запит Ajax виконується без параметра dataType, що спричиняє виконання текстових/яваскриптових відповідей.	jQuery 2.2.0
4.3	CVE-2019-11358	jQuery до версії 3.4.0, який використовується в Drupal, Backdrop CMS та інших продуктах, неправильно обробляє jQuery.extend(true, {}, ...) через забруднення Object.prototype. Якщо необроблений вихідний об’єкт містив перелічувану властивість __proto__, він міг би розширити рідний Object.prototype.	jQuery 2.2.0

4.3	CVE-2020-11022	У версіях jQuery, які перевищують або дорівнюють 1.2 і до 3.5.0, може виконуватися передача HTML з ненадійних джерел - навіть після його очищення - одному з методів маніпуляції DOM jQuery (наприклад, .html(), .append() та інші ненадійний код. Ця проблема виправлена в jQuery 3.5.0.	jQuery 2.2.0
4.3	CVE-2020-11023	У версіях jQuery, більших або рівних 1.0.3 і раніше 3.5.0, передача HTML, що містить	jQuery 2.2.0
4.3	CVE-2019-20372	NGINX до 1.17.7, з певними конфігураціями error_page, дозволяє контрабанду HTTP-запитів, про що свідчить здатність зловмисника читати неавторизовані веб-сторінки в середовищах, де NGINX працює через балансувальник навантаження.	Nginx 1.16.1

Опис ризику:

Ці вразливості наражають уражені програми ризику несанкціонованого доступу до конфіденційних даних і, можливо, атак відмови в обслуговуванні. Зловмисник може шукати відповідний експлойт (або створити його самостійно) для будь-якої з цих вразливостей і використовувати його для атаки на систему.

Рекомендація:

Необхідно оновити уражене програмне забезпечення до останньої версії, щоб усунути ризик виникнення цих вразливостей.

Відсутній заголовок безпеки: X-XSS-Protection

Табл. 3.3

X-XSS-Protection

URL	Evidence
https://anybag.ua	Заголовки відповіді не містять заголовок безпеки HTTP X-XSS-Protection

Опис ризику:

Заголовок HTTP X-XSS-Protection наказує браузеру припинити завантаження веб-сторінок, коли вони виявляють атаки міжсайтових сценаріїв (XSS). Відсутність

цього заголовка наражає користувачів програми на XSS-атаки, якщо веб-додаток містить таку вразливість.

Рекомендація:

Необхідно встановити заголовок X-XSS-Protection на X-XSS-Protection: 1; режим=блок.Знайдено серверне програмне забезпечення та технології

Програмне забезпечення / Версія

Категорія - Nginx 1.16.1

Веб-сервери - PHP 7.2.30

Мови програмування -PHP

CMS - 1С-Бітрікс

Веб-фреймворки - Twitter Bootstrap

Віджети - Facebook

Аналітика - Google Analytics UA

Скрипти шрифтів - Google Font API

Менеджери тегів - Менеджер тегів Google

Фреймворки JavaScript - Select2, jQuery 2.2

Опис ризику:

Зловмисник може використовувати цю інформацію для здійснення конкретних атак на визначений тип і версію програмного забезпечення.

Рекомендація:

Необхідно виключити інформацію, яка дозволяє ідентифікувати програмну платформу, технологію, сервер та операційну систему: заголовки сервера HTTP, метаінформацію HTML тощо.

Список проведених тестів (13/13)

Перевірка доступності веб-сайту...

Перевірка безпечного прапорця cookie...

Перевірка відсутності заголовка HTTP - Strict-Transport-Security...

Перевірка відсутності заголовка HTTP - Політика безпеки вмісту...

Перевірка відсутності заголовка HTTP - X-XSS-Protection...

Перевірка відсутності заголовка HTTP - Referrer...

Перевірка технологій веб-сайту...

Перевірка на наявність вразливостей програмного забезпечення на стороні сервера...

Перевірка прапора файлу cookie HttpOnly...

Перевірка політики доступу клієнта...

Перевірка файлу robots.txt...

Перевірка використання ненадійних сертифікатів...

Перевірка ввімкнених методів налагодження HTTP…

Статистика

URL-адреси у вигляді павуків: 9

Загальна кількість помилок запиту HTTP: 1

Загальна кількість HTTP-запитів: 18

Виявлено унікальних точок введення: 339

3.4. SEO аналіз та оптимізація

Пошукова оптимізація (англ. Search engine optimization, SEO) - комплекс заходів щодо внутрішньої і зовнішньої оптимізації для підняття позицій сайту в результатах видачі пошукових систем по певних запитах користувачів, з метою збільшення мережевого трафіку (для інформаційних ресурсів) і потенційних клієнтів (для комерційних ресурсів) і подальшої монетизації (отримання доходу) цього трафіку. SEO може бути орієнтоване на різні види пошуку, включаючи пошук зображень, відеороликів, пошук новин і специфічні галузеві пошукові системи. (Вікіпедія)

Основні фактори SEO оптимізації:

1. Вік домену – чим старший домен, тим більше він впливає на індексацію.
2. Довжина домену – дуже довгі домени гірше індексуються пошуковими системами.
3. Довжина вмісту – вміст , що містить більше слів, може охопити більш широку аудиторію і ранжується в алгоритмі краще порівняно з більш короткими, поверхневими статтями.
4. Швидкість завантаження сторінки – чим більша швидкість, тим краще.
5. Унікальний контент – чим контент на сторінці більш унікальний, тим краще.
6. Оптимізація зображень – сторінки з малим розміром картинок, швидше завантажуються, та як наслідок краще індексуються пошуковими роботами.
7. Частота оновлень контенту – чим частіше з'являється щось нове, тим краще
8. Правопис – якщо контент сторінки граматично та лексично складено то така сторінка займе позиції вище.
9. Адаптивність сторінки – якщо сторінка коректно відображається на мобільному пристрої, це добре вплине на ранжування сайту.

10. Захований контент на мобільному – якщо контент на ПК версії та мобільній відрізняєтсья це може погано вплинути на SEO
11. Мультимедіа – на сайті повинні бути фото, відео
12. Некоректні посилання – якщо на сторінку є посилання які нікуди не ведуть, це дуже погано вплине на оцінку сторінки
13. Помилки HTML - багато помилок HTML або неохайного кодування може бути ознакою сайту низької якості. Незважаючи на суперечливість, багато хто з SEO вважають, що добре кодована сторінка використовується як сигнал якості.
14. Довжина URL – сторінки з занадто довгими адресами гірше індексуються
15. Наявність SiteMap – наявність карти сайту, позитивно сказується на ранжирувані сторінки
16. Пріоритетність сторінки в sitemap.xml
17. Вік сторінки - хоча пошуковити більш лояльні дл свіжому вмісту, стара версія сторінки, що часто поновлюється, може перевершити нову сторінку.
18. Сторінка «зверність до нас» - наявність такої сторінки майже обов'язкова, так пошуковики розуміють, що в випадку проблем на сайті користувач може зв'язатися з розробниками сайту
19. Час роботи сайту – якщо сайт тривалий час не відповідає, такі сайти знижуються в видачі, то так важливо вибирати якісний домен
20. Наявність SSL сертифікату
21. Наявність навігації – наявність так званих “хлібних крихт” обов’язкове
22. Наявність відео з YouTube - відео що вмонтовані на сайт з YouTube отримують пільгову обробку в програмах SERP
23. Наявність атрибутів alt в зображень
24. Налаштовані 301 редиректи

25. Наявність заповненого атрибуту title в посиланнях

Перевіримо індексація сайту на даний момент:

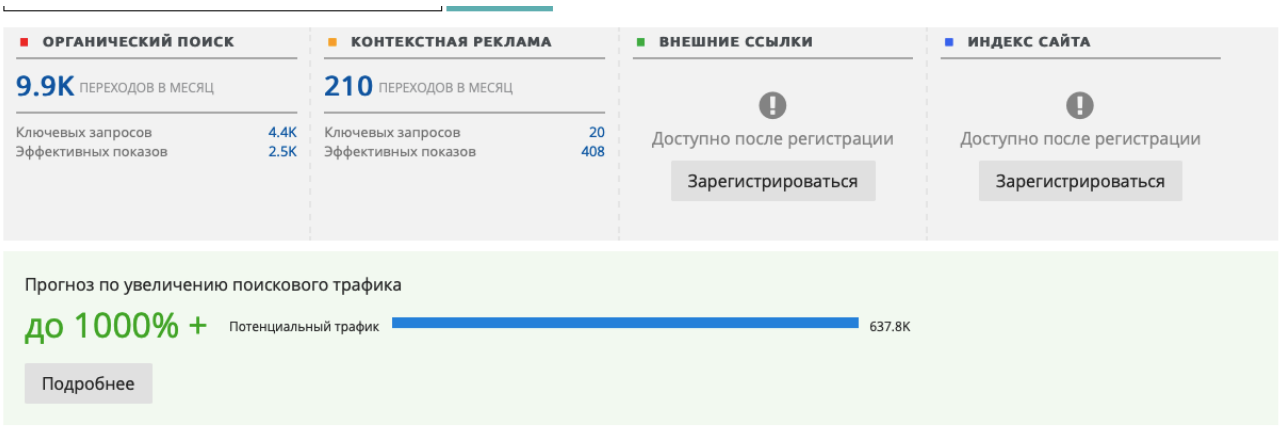


Рис. 3.5. Перевірка індексації пошукових систем

Проаналізувавши дані на Рис 3.5 можно зробити висновок, що веб-сайт має досить хороший органічний пошук, та забезпечує необхідну планову статистику по SEO оптимізації.

Також можна дізнатися про джерело переходу, який був запит до пошукових систем і з якої країни.

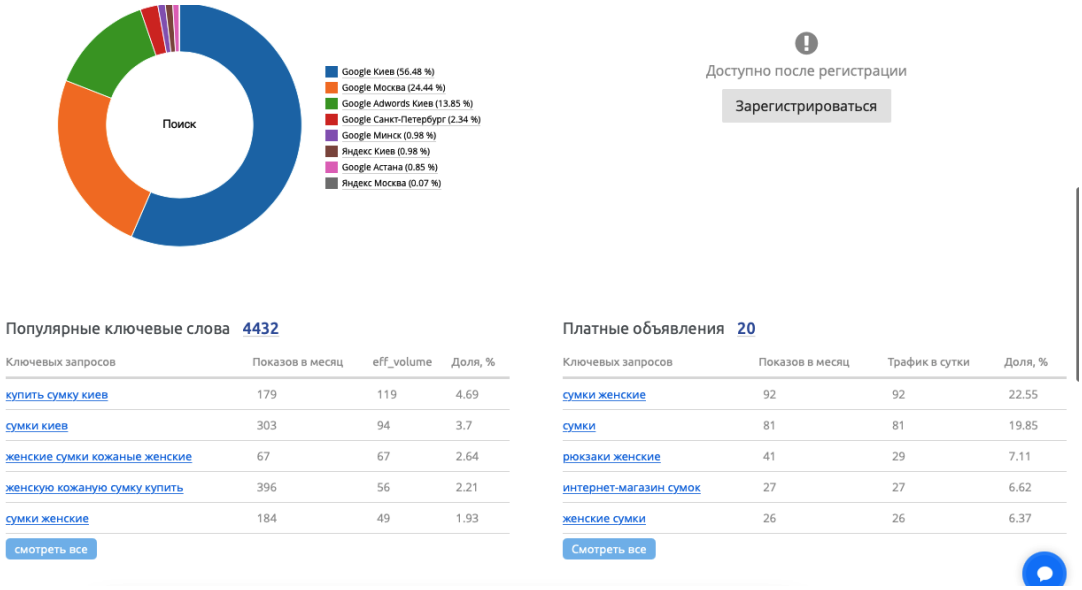


Рис. 3.6 Перевірка джерел переходу пошукових систем

Виходячи з результату перевірки на Рис 3.6, видно, що найбільше переходів було з України, а основні запити до пошукових систем, включали в себе інформацію про купівлю сумок.

3.5. Шляхи покращення SEO індексації

1. Публікація релевантного, авторитетного вмісту

Якісний, авторитетний контент є основним драйвером вашого рейтингу пошукових систем. Якісний вміст, створений спеціально для цільового користувача, збільшує відвідуваність сайту, що підвищує авторитет і релевантність вашого сайту.

Ключові слова

Потрібно визначити певну ключову фразу для кожної сторінки вмісту на вашому веб-сайті.

Зміст

Крім URL-адреси сторінки, заголовка та заголовків, вміст має найбільший вплив на рейтинг у пошукових системах. Потрібно повторювати ключову фразу кілька разів по всій сторінці — один-два рази в початковому та заключному абзацах і ще два-чотири рази в решті вмісту.

2. Регулярність оновлення свій вміст

Для оптимізації SEO показників слід регулярно оновлювати вміст, це вважається одним із найкращих показників релевантності сайту, тому потрібно слідкувати за його оновленням. Разом з цим необхідно оновлювати дані в файлі sitemap.xml

Рис. 3.7. Приклад файлу sitemap.xml

3. Метадані

Під час розробки веб-сайту кожна сторінка містить пробіл між тегами <head> для вставки метаданих або інформації про вміст вашої сторінки. Однак важливо переглядати та оновлювати метадані, коли сайт змінюється з часом.

Метадані заголовка

Метадані заголовків відповідають за заголовки сторінок, які відображаються у верхній частині вікна браузера, і як заголовок у результатах пошукової системи. Це найважливіші метадані на сторінці.

Описові Метадані

Метадані опису – це текстовий опис, який браузер може використовувати під час пошуку на вашій сторінц. Хороший мета-опис зазвичай містить два повних речення. Пошукові системи можуть не завжди використовувати мета-опис, але важливо дати їм можливість.

Метадані ключових слів

Метадані ключових слів рідко використовуються, якщо взагалі коли-небудь використовуються для зведення в таблицю рейтингу пошукових систем. Однак повинні знати свої ключові фрази, тому не завадить додати їх до метаданих ключових слів. Вам потрібно включити різноманітні фрази. Як загальне правило, намагайтеся містити приблизно 3-7 фраз, кожна фраза складається з 1-4 слів. Чудовим прикладом може бути «диплом із інформатики».

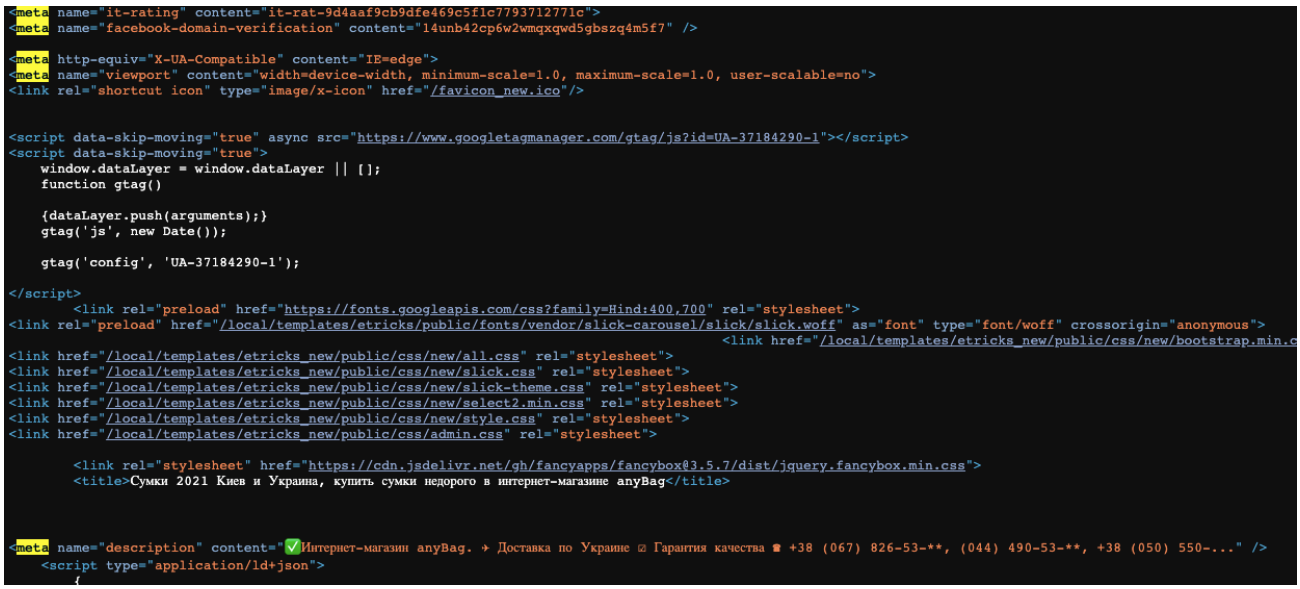


Рис. 3.8. Приклад meta даних

4. Якісні посилання на сайт

Веб-сторінка, яка містить багато вмісту, авторитетна, неупереджена та допомагає відвідувачам дізнатися більше про те, що їх цікавить, найімовірніше залучатиме посилання з інших веб-сайтів, що покращує пошукову оптимізацію.

Потрібно завжди використовувати описові посилання, зв’язуючи ключові слова — це не тільки покращує пошукову оптимізацію, але й додає цінності для ваших читачів, у тому числі людей з обмеженими можливостями або тих, хто використовує програми зчитування з екрана.

5. Теги alt

Необхідно описувати свої зображення та відео за допомогою тегів alt або альтернативних текстових описів. Вони дозволяють пошуковим системам знайти вашу сторінку, що має вирішальне значення — особливо для тих, хто використовує лише текстові браузери або програми зчитування з екрана.

```
<div id="header">
    <div class="container">
        <ul class="header-btns">
            <div class="header-logo">
                <a class="logo" >
                    <img
                            src="/local/templates/etricks_new/public/images/vendor/etricks/anybag-logo.png"
                            alt="Интернет-магазин сумок anyBag"
                            title="Интернет-магазин сумок anyBag"
                    />
                </a>
                <div class="logo-text">Магазин модных сумок и аксессуаров</div>
            </div>
            <li class="header-search-open xs-hide">
                <div class="header-btns-icon">
                    <span class="fa fa-search"></span>
                </div>
            </li>
            <li class="header-phone-open xs-hide">
                <div class="header-btns-icon">
                    <span class="fa fa-phone"></span>
                </div>
            </li>
            <li class="header-account header-user-options-open dropdown default-dropdown ">
                <a
```

Рис 3.8. Приклад alt в тезі img

ВИСНОВКИ

У дипломній роботі вирішено такі завдання:

- Аналіз системи Bitrix
- Підготовка технічного завдання
- Реалізація файлової структури проекту
- Вибір програмного забезпечення серверу

Використання мови програмування PHP в складі системи 1C-Bitrix дає такі переваги:

- Синтаксис PHP інтуїтивно зрозумілий
- Кросплатформеність
- Наявність документації
- Наявність активної спільноти
- Легка масштабованість

У описаній системі керуванні будуть реалізовані всі поставленні завдання та цілі. Що може свідчити про високу ефективність використання даної системи.

Розроблений веб-сайт дозволяє автоматизувати процеси продажу товарів використовуючи функції, які до цього були розробленні.

В другому розділі представлена розробка системи керування інтернет-магазином на базі 1C-Bitrix, та детально проаналізован архітектурні рішення, також представленні деякі теоретичні відомості про програмні засоби, які використані під час розробки системи керування контентом.

В третьому розділі відображено процес тестування сайту за багатьма критеріями та представлені графічні розрахунки за деякими етапами тестування. Головне, що можна виділити; тестування – є найважливішим етапом циклу

розробки програмного продукту. Основна ціль тестування – пошук багів (помилок).

Основними принципами тестування - є кроки, які демонструють користувачеві, наскільки проект є корисним, правильним та логічно побудованим, чи добре сприймається користувачем і чи правильно робить весь реалізований функціонал додатку. Також в третьому розділі розглянуто основні правила SEO – оптимізація, та проведено SEO аналіз використовуючи спеціалізовані автоматичні сервіси аналізу.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ

1. Диаграмма компонентов (component diagram) [Електронний ресурс] – Режим доступу до ресурсу: http://khpiiip.mipk.kharkiv.edu/library/case/leon/gl10/gl10.html.

2. Діаграма компонентів [Електронний ресурс] // wikipedia – Режим доступу до ресурсу: https://uk.wikipedia.org.

3. Документация [Електронний ресурс] // 1С Битрикс – Режим доступу до ресурсу: https://dev.1c-bitrix.ru/docs/.

4. Барысов Р. Структура файлов [Електронний ресурс] / Роберт Барысов // Bitrix. – 2010. – Режим доступу до ресурсу: https://dev.1c-bitrix.ru/learning/course/.

5. Farhad M. Energy Efficiency in Data Centers and Clouds [Електронний ресурс] / Mehdipour Farhad – Режим доступу до ресурсу: https://www.sciencedirect.com/topics/computer-science/big-data-processing.

6. Amazon Web Services [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Amazon_Web_Services.

7. https://habr.com/ru/company/otus/blog/484048/ [Електронний ресурс]. – 2016.

8.PHP.SU - MySQL для начинающих [Онлайновий] // PHP.SU. - http://www.php.su/articles/.

9. - Myers, Glenford J.The art of software testing / Glenford J. Myers, Corey Sandler, Tom Badgett. — 3rd ed. - John Wiley & Sons, Inc., Hoboken, New Jersey, 2012. - 240 p.

10. - Bird C. The Art and Science of Analyzing Software Data / Christian Bird, Tim Menzies, Thomas Zimmermann - Morgan Kaufmann, Waltham, MA 02451, 2015. - 660 p.

11. Roger Lee (Ed.) - Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing. / Springer Nature Switzerland AG 2020. - 262 p.

12. - Ajay Kumar Jena, Himansu Das, Durga Prasad Mohapatra (Editors) - Automated Software Testing:Foundations, Applications and Challenges. / Springer Nature Singapore Pte Ltd. 2020. - 165 p.