Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: «**Розробка нових модулів та оптимізація моделі** 1 **з демонстрацією особливостей проектування за методом RUP, мова програмування PHP**

Виконав(ла): студент(ка) VI курсу, групи СПм-61

спеціальності

121 – Інженерія програмного забезпечення
(шифр і назва спеціальності)

	(підпис)	Дзись Н.Т. (прізвище та ініціали)
Керівник	(підпис)	Цуприк Г.Б. (прізвище та ініціали)
Нормоконтроль	(підпис)	Стоянов Ю.М. (прізвище та ініціали)
Завідувач кафедри	(підпис)	Петрик М.Р. (прізвище та ініціали)
Рецензент	(підпис)	Михайлишин М.С. (прізвище та ініціали)

Тернопіль 2021

АНОТАЦІЯ

Дзись Н.Т. Розробка нових модулів та оптимізація моделі з демонстрацією особливостей проектування за методом RUP, мова програмування PHP. – Рукопис.

Кваліфікаційна робота на здобуття освітнього ступеня «магістр» за спеціальністю 121 – Інженерія програмного забезпечення, Тернопільський національний технічний університет ім.Івана.Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії,група СПм-61, місто Тернопіль, 2021 рік. Пояснювальна записка до атестаційної роботи на здобуття освітнього ступеню «магістр» містить: с., рис., табл., додатків.

Метою кваліфікаційної роботи є вдосконалення через оптимізацію моделі та розробка нових модулів з обґрунтуванням та демонстрацією особливостей проектування з врахуванням принципів раціонального уніфікованого підходу.

Практичне застосування – корисність ідеї полягає в тому, щоби показати, що дотримання строгого підходу до розподілу як завдань так і відповідальності в середині команди-розробника програмного забезпечення ґарантуватиме можливість реалізації всіх запланованих етапів створення програмного продукту точно по обумовленим термінам та не виходячи за розмір передбаченого бюджету передбаченого на створення якісного програмного продукту, який відповідатиме в повній мірі вимогам замовника.

Технічні вимоги – для розробки запропоновано до використання систему керування вмістомCMS, мова програмування PHP.

Ключові слова: ПРЕДМЕТНА ОБЛАСТЬ, ОБ'ЄКТ, МОДЕЛЬ, ОПТИМІЗАЦІЯ, МОДУЛЬ, ПРОЄКТУВАННЯ, ОЦІНКА ЯКОСТІ, ОХОРОНА ПРАЦІ,БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ

ABSTRACT

Dzys N.P. Development of new modules and optimization of the model with demonstration of Software Design features using RUP method, programming language is PHP. – Manuscript.

Qualifying work is for master's degree in specialty 121 — Software Engineering. – Ternopil Ivan Pul'ui National Technical University, Faculty of Computer Information Systems and Software Engineering, Software Engineering Department, group number SPm-61 // Ternopil', 2021. Qualifying work contains pages, figures, tables and supplements.

The arm of the qualification work is to improve through model optimization and development of new modules with justification and demonstration of software design features taking into account the principles of rational unified approach.

Practical application – the usefulness of the idea is to show that adherence to a strict approach to the distribution of both tasks and responsibilities within the software development group will guarantee the possibility of implementing all planned stages of software development exactly on time and without exceeding the budget. provided for the creation of a quality software product that will fully meet the requirements of the customer.

Methods of developing technology based on CMS content management system and the PHP programming language.

Keywords: SUBJECT AREA, OBJECT, MODEL, OPTIMIZATION, MODULE, SOFTWARE DESIGN, QUALITY ESTIMATION, LABOUR PROTECTION, LIFE AND HEALTH PROTECTION

ЗМІСТ

ВСТУП

Актуальність ролі інформаційних технологій в сучасному світі є безумовною. Сьогодні престиж, сучасність та успішність фірми в тому числі визначається використанням та оптимізацією робочого процесу через перехід у віртуальний світ. Сьогодні це як ніколи актуально в силу того, що ми є сучасниками нових реалій і цілком нового життя в світі суцільних жорстких пандемічних обмежень. Стан речей сьогодні і довго перспективні тенденції красномовно свідчать про те, що Світ в сфері економіки та відповідної діяльності немає ніяких шансів на виживання без переходу у віртуальну його частину. І якщо ще кілька місяців тому можна було говорити та планувати таку діяльність частково, то сьогодні мова йде про повну переорієнтацію, а для тих хто не погоджується – мова йде про економічний занепад чи виживання в прямому сенсі.

З огляду на ситуацію ніби все зрозуміло і як варіант не просто «втриматися на плаву», а й розвиватись отримуючи новий досвід хоч і вимушено проте є кілька важливих моментів на які варта звернути увагу. Галузь знань 12 – Інформаційні технології сьогодні – це тренд. Чи не кожен випускник школи чи коледжу хоче бути програмістом і відповідно до статистичних даних 2021 року більше п’ятдесяти відсотків планують і вступають за цим напрямком. З них близько половини працює по спеціальності і вважає себе і часто небезпідставно спеціалістом високого класу. Але, часто буває так що програміст чи навіть команда працюють в стилі найбільш знайомому їм не беручи до уваги передовий досвід та вдосконалення цієї динамічної галузі новинки в якій з’являються ледь не щодня. Хоча я вважаю, що це не правильно оскільки Інженерія програмного забезпечення це одна з небагатьох спеціальностей робота в якій завжди (ще задовго до будь-яких обмежень) власне і передбачає віддалену роботу, тому реалії лише незначною мірою повпливали на неї. Однак незважаючи на цей факт, чомусь далеко не завжди у своїй роботі спеціалісти цього виду діяльності

раціонально та уніфіковано підходять до своїх обов’язків часом частково нехтуючи формальною стороною питання.

Таким чином раціональний уніфікований підхід, скорочено RUP, передбачає дотримання строгого підходу до розподілу як завдань так і відповідальності в середині команди-розробника програмного забезпечення. Відповідно до його парадигм ґарантується можливість реалізації всіх запланованих етапів створення програмного продукту точно по обумовленим термінам та не виходячи за розмір передбаченого бюджету передбаченого на створення якісного програмного продукту, який відповідатиме в повній мірі вимогам замовника.

Також варта наголосити, що раціональний уніфікований підхід – це вже момент оптимізації та вдосконалення, модулі, які розробляються з його врахуванням будуть працювати якісніше по всьому функціоналу як окремо так і програми в цілому. Особливості проєктування якісного програмного продукту за RUP підвищить продуктивність роботи всієї команди і надасть корисний додатковий досвід з можливістю його накопичення (оскільки при роботі над створенням кожного нового проєкту навики будуть лише вдосконалюватись) використовуючи різноманітні інформаційні джерела, шаблони та настанови по використанню інструментальних засобів на всіх критично важливих етапах роботи, впродовж всього життєвого циклу створення та супроводу програмного продукту.

1 ОСНОВНА ЧАСТИНА

1.1 Опис моделі предметної області

Сьогодні, в складний, безприцендентний час пандемічних обмежень, коли вихід з дому щоразу наражає на небезпеку кожного мешканця планети Земля є лише один інструмент, який дає можливість вільно, без будь яких зусиль і що найважливіше сьогодні, безпечно отримати, оприлюднювати, ділитись та повідомляти інформацію будь-якого напрямку діяльності чи то наукова, чи ділова, чи пізнавальна, чи розважальна. Всесвітня мережа позв'язує напевно усі відомі, великі, важливі чи то наукові чи урядові організації всього світу, такі як:університети, бізнес-центри, інформаційні агентства, видавництва, тощо. Сьогодні ми користуємось ефективним інструментом, за допомогою якого вдалось створити величезне сховище даних, зібраних з усіх галузей людського вміння та знання, віртуальних бібліотек, архівів, стрічок новин, які містять в собі величезні об'єми і текстової, і графічної, і аудіо і відео інформації.

Сьогодні ми вже не можемо диференціювати Глобальну мережу від сучасної цивілізації, невід'ємною частиною якої вона є. У будь якій зі сфер життєдіяльності чи то освіта, чи торгівля, чи зв'язок, чи послуги, завдяки йому створено (і цей процес продовжується та триває неперервно) нові форми спілкування, навчання, комерційної діяльності та розваг. Сучасне покоління – є інтернетозалежним, це абсолютно новий та не подібний на попередні соціально-культурний феномен. Для дітей сьогодні інтернет – це невід'ємна частина їх віртуального життя, яке вони часто (і це величезна проблема) переносять в реальне життя, необдумано наражаючи себе на небезпеку, оскільки не розуміють що це два різних світи зі своїми правилами та законами, особливостями, специфікою і навіть хворобами. Без перебільшення вже можна сказати, що людство вступило до нового інформаційного етапу свого розвитку, в якому мережеві технології без перебільшення відіграють величезну роль, та в якому інтернет це вже реалії звичного, комфортного, та зручного життя, якість

якого оцінюється в тому числі і можливістю доступу до інтернету. Хоча справедливо зазначити, що виникненню його передувала ідея створення глобального сховища для інформації в поєднанні з універсальним засобом для її поширення.

Про те, що робота з інформацією, даними є архіважливим питанням складність якого зумовлена в тому числі об'ємами говорилось вже достатньо давно. Ще тоді, коли такі технології були недоступні не лише широкому загалу, а й вузькопрофільним спеціалістам американські вчені Веннівер Буш (англійською Vannevar Bush, часом також Венівар Буш, Ванневар Буш, який народився у 1890 [1] і Теодор Голм Неельсон (англійською Theodor Holm Nelson, який народився у 1937 році) [2,3] працювали над пошуком способів (варіантів) оптимізації розумової діяльності людини результатом якої є великі об'єми даних, та розробкою так званої «електронної літератури». Їх метою була автоматизація та пришвидшення рутинного процесу пошуку і опрацювання потрібної інформації. Веннівер Буш навіть придумав декілька пристроїв гіпотетичного толку, якими організовуються асоціативні зв'язки представлені у картотеці даних, а Теодор Голм Неельсон розробив теорію "документарію всесвіту", в якому всі знання, які людство накопичело, представлені у вигляді єдиної інформаційної системи, яку пронизують мільярди перехресних посилань. Однак, на той період часу про роботи цих вчених можна було говорити швидше за все в філософському сенсі, ніж про їх практичний характер. Однак запропоновані ними ідеї поклали основу того, що сьогодні називається гіпертекстом.

Веннівером Бушем було чимало зроблено з метою зацікавлення наукою військовими. Момент щедрого фінансування досліджень в галузі кібернетики безумовно сприяло ришвидшенню розвитку. Значна роль в формуванні теоретичної бази в майбутньому глобальної інформаційної системи однозначно належить Норберту Вінеру [5,6]. Його блискучі семінари в Массачусетському технологічному інституті (MIT) дозволили залучити до комп'ютерної галузі чимало талановитих молодих людей.

В кінці 1950-х років міністерством оборони США заснувано Аґентство по перспективним дослідницьким проєктам ARPA (AdvancedResearchProjects Agency), діяльність якого полягала в комп'ютерному моделюванні подій військового і політичного характеру. Талановитим організатором та вченим-комп'ютерником Джозефом Карлом Робнетт Ліклайдером (англійською Joseph Carl Robnett Licklider, який народився в 1915) [7], керівництво ARPA було переконане зосередити зусилля на розвитку комп'ютерного зв'язку та мереж. У своїй роботі ним фактично була розроблена структура майбутньої всієї глобальної мережі [8,9,10].

Починаючи із 1960-х років комп'ютерні мережі почали дуже бурхливо розвиватись. Виникла величезна кількість фірм, яким створювалось програмне забезпечення та обладнання для локальної мережі різних установ та організацій. Та природно були і проблеми, зокрема виникло питання до сумісності.

В результаті AdvancedResearchProjects Agency отримало задачу вирішити проблеми. Цими питаннями займались Поль Берен [2], Лоуренс Гілман Робертс [11,12] і Вінтон Ґрей Серф [2], якими було розроблено і застосовано методи, які лягли в основу подальшого розвитку мережевих технологій, зокрема сюди варта віднести: пакетну комутацію, динамічну маршрутизацію повідомлень для розподіленої мережі, застосування універсального мережевого протоколу (набір правил, з врахуванням яких організується і, що важливо, передається інформація). Це призвело до створення мережі ARPANET, яка і була прототипом для майбутнього Інтернету. Саме тому 1969 рік традиційно вважають роком виникнення мережі інтернет [13,14,15].

В 1976 році Вінтоном Ґрей Серфом було розроблено універсальний протокол передачі даних TCP/IP(англійською TransmissionControlProtocol/ Internet protocol). Назва IP означало просто міжмережевий протокол. Саме його вважають стандартом між мережевих комунікацій [2].

У 1980-х роках Інтернет не був поширений серед населення зовсім і з ним працювати та ним користувались лише фахівці.

А вже 1990 році програмістом CERN (Європейський центр ядерних досліджень) у Женеві Тімом Бернерс-Лі було створено систему, якою реалізується ідея єдиного гіпертекстового простору. Бернерс-Лі назвав свій проект WWW - World Wide Web, тобто “Всесвітня павутина” [16,17].

Але справді «народної» популярності Інтернет набув лише після виходу “Мозаїки” (Mosaic) – графічний браузер, який розробив у 1992 році Марк Андресен (англійською Marc Andreesen) – співробітник Іллінойського університету. На рисунку 1.1 представлена динаміка розвитку мережі з серпня 1995 по квітень 2014 року.

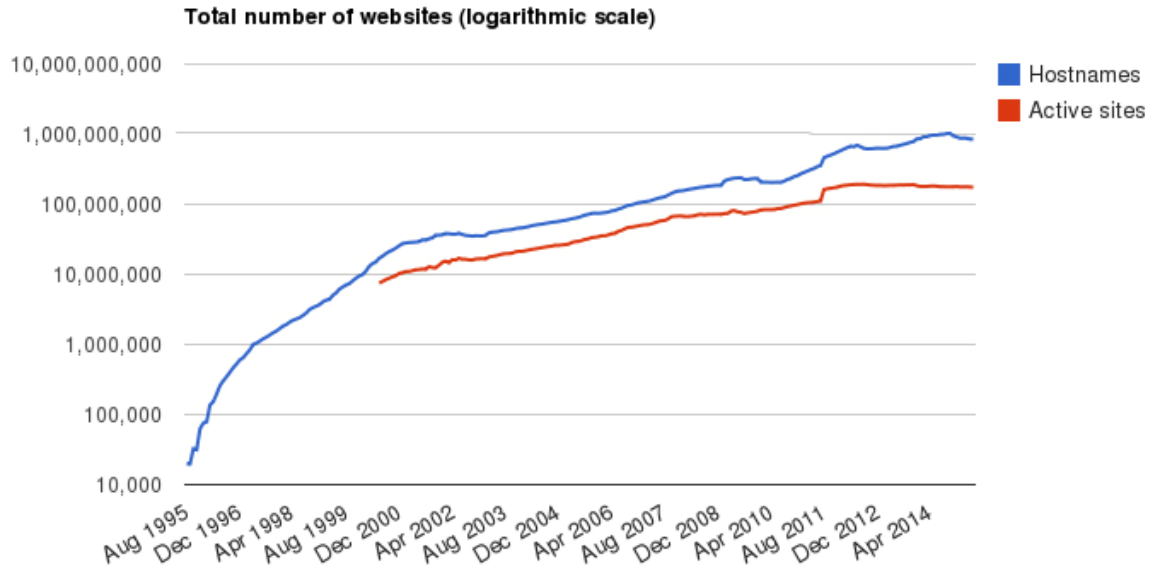

Рисунок 1.1 – Лінія тренду інтернет-ресурсів в визначений період, ресурс кампанії “Netcraft Ltd” (серпень 1995 року – квітень 2014 року)

Аналізуючи загальнодоступну інформацію можу стверджувати, що Всесвітня мережа, сьогодні,її розвиток відбувається експоненційною кривою, а це означає, що приблизно кожних два рики відбувається кількісних показників (сюди можна віднести число користувачів, число під’єднаного комп’ютерного обладнання, обсяги інформації і трафік, кількість інформаційних ресурсів). Аналізуючи цю тенденцію, роблю висновки про те, що ці показники будуть лише рости, адже важливість її в життєдіяльності кожної людини (чи більшості людей) лише зростає.

На рисунку 1.2 представлені зміни чисельності користувачів в Україні 2000–2010 роки. Судячи з представленої тенденції, навіть немає змісту шукати та оновлювати дані, оскільки все говорить про те, що картина буде мінятись лише в сторону підвищення показників і вже нічого в світі не зможе цього зминити. А розуміючи на реалії сьогодення, активне життя в різних напрямках все впевненіше переходить у віртуальний світ.

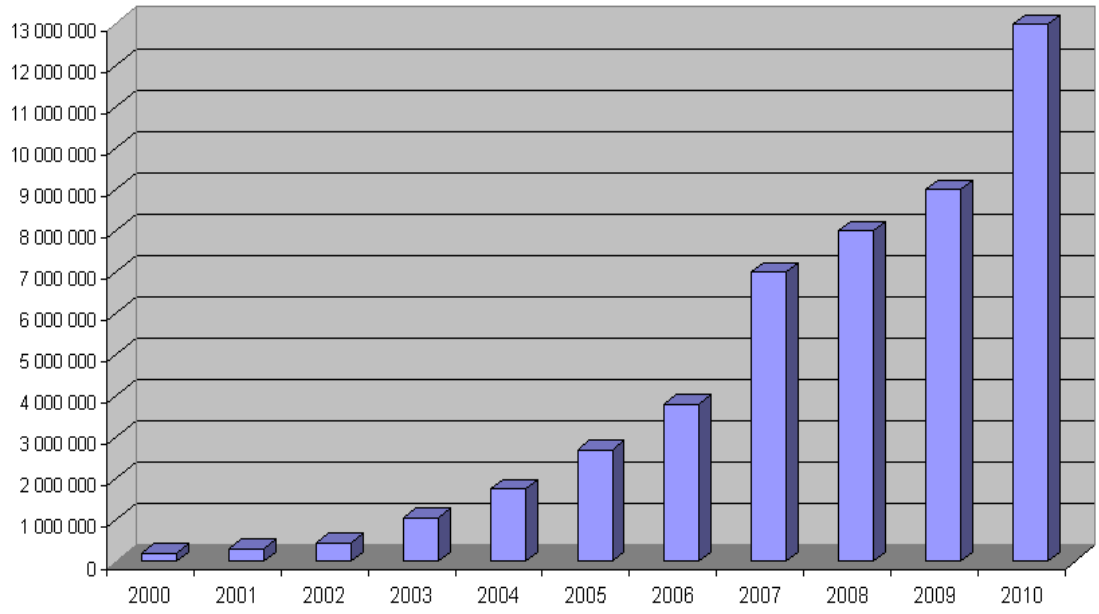

Рисунок 1.2 – Зміни чисельності користувачів Всесвітньої мережі за даними по Україні станом на 2000–2010 роки

Таким чином, підсумовуючи все вище сказане, можу з упевненсстю стверджувати, що більшість процесів мережевого проєктування, адміністрування й обслуговування повністю перейдуть до автоматизованого вигляду.

1.1.2 Аналіз та класифікація web-сайтів різних видів та типів

Якщо брати до уваги програмне забезпечення, можна сказати, що веб-сервером є сервер, який зазвичай може містити декілька компонентів, за

допомогою яких можна контролювати доступ веб-користувачів до розміщених на сервері файлів. Як правило це по мінімуму має бути HTTP-сервер, який є частиною програмного забезпечення, яке може «розуміти» URL- веб-адреси та HTTP є протоколом, який використовується браузером користувача задля перегляду веб-сторінок.

Безпосередній доступ користувачів до необхідної інформації, зазвичай, знаходиться на комп'ютерних веб-серверах, і реалізувати його можливо лише через встановлене спеціалізоване програмне забезпечення. Організація значної частини цієї інформації має вигляд веб-сайтів зі своїм ім'ям та адресою в Internet-і.

В загальному, можна сказати, що веб-сайтом є інформація, представлення якої відбувається в деякому вигляді, та яку розташовано на веб-сервері та зі своїм ім'я та адресою кожної. Для отримання можливості перегляду веб-сайтів користувачеві потрібно скористатися програми- браузерами. В залежності від імені та адреси сайту у рядку «Адреса» висяться відповідні відомості і тоді відповідна інформація браузером завантажуватиметься у відповідному вікні.

Веб-сторінка є інформаційним ресурсом, який складено з позв'язаних між собою веб-сторінок (гіпертекстових документів), які розташовано на веб-сервері та який має індивідуальну адресу.

Кількість веб-сторінок та їх типи визначаються обсягом і характером матеріалів сайту.

Перше найбільш загальне поділ можна провести по загальному робочому призначенням сайту – це або інформаційний сайт, або сервісний сайт. У реальному житті в чистому вигляді вони майже не існують, а представлені комбінаціями в різних пропорціях можливостей одного й іншого типу сайтів. Ці комбінації дуже численні і саме вони і складають звичні для нас і часто звучать назви сайтів: сайт візитка, інтернет магазин, портал і т.д. Ці види сайтів ми розглянемо трохи нижче. Поки ж ще одне загальне поділ сайтів, але вже за способом управління інформацією на сайті. Це статичні сайти і динамічні сайти.

Сайти можуть бути незмінними чи т.з.статичними, такими які мають набір постійних та незмінюваних сторінок на яких за допомогою коду інформація

жорстко закріплена. Проте, слід зазначити, що зміни таки можна вносити, але для цього варта скористуватись послугамии фахівців у веб-технологіях, що є суттєвим недоліком таких сайтів, оскільки в кінцевому рахунку їх собівартість може зрости в рази із-за ціни послух в даному сегменті IT-ринку. Такі сайти є бюджетним рішенням для замовника.

Основною є перевагою динамічного сайту є вбудована система управлінням його контентом (англійською Content management system, скорочено CMS), до якого відносять його вміст, функції, модулі та сервіси. Тип створення сторінок такого сайту є динамічним, і це є їх однією з основних переваг, оскільки оновлення інформації відбувається по мірі необхідності та без запрошення відповідних фахівців як це відбувається для сайтів статичного типу. Публікація та приховування інформації може здійснюватись кожного разу при виникненні такої потреби.

Звичайно лише умовно, проте існує поділ представлених у вільному доступу сайтів за завданнями, які вирішується за допомогою них та способами їх вирішення. Також хочу наголосити, що для реально працюючих сайтів характерною є комбінація типів. Отже, сайти можна умовно поділити наступним чином:

–сайти візитка,

–промоційний сайт,

–сайт корпоративного призначення,

–інтернет-вітрина,

–інтернет-магазин,

–тематичний сайт,

–блог,

–інтернет-портал,

–каталог сайтів,

–поштова система,

–пошукова система,

–форум,

–фото-, відео-, аудіо-, обмінник,

–дошка оголошень,

–соціальна мережа.

Вибір певного виду сайту повністю залежить від потреби замовника та його фінансових можливостей, але сьогодні він є невід'ємною частиною для успішного ділового життя. Сайт допоможе розвинути діяльність не лише початківцям, але й точно не буде зайвим для розширення діяльнjсті та осучаснювання вже фірм із іменем. Однак ще раз хочу наголосити, що ефективним такий прогрманий продук буде лише в разі точного розуміння потреби.

1.1.3 Обґрунтування актуальності розроблюваної моделі

Загально відомо, що віртуальним магазином, інтернет-магазином є магазин, з розміщеною в Інтернеті "вітриною" та за домомогою якого можна замовити, придбати, оплатити будь-що з представленого в наявності переліку так само в мережі. Таким чином повністю закриваються всі проблемні моменти, характерні сьогодні: безпека, швидкість, відсутність можливості інфікування перебуваючи в приміщеннях, і подібне. Сьогодні велика кількість людей вже зацікавлена такою можливістю, а всі інші в недалекому майбутньому також, в силу обставин, будуть змушені освоїти цю можливість та протестувати всі її переваги та недоліки. Зазвичай така форма реалізації товарів – це цілий тип чи навіть різновид сайтів основною функцією яких є вигідне та зручне, впорядковане представлення товарів як в малому так і середньому бізнесі. Через такі магазини є можливість реалізувати практично будь-які схеми торгівлі, зокрема сюди можна віднести торгівлю зі складу, торгівлю під замовлення, торгівлю для приватних осіб, торгівлю з та поміж організаціями, торгівля цифровими товарами, торгівля послугами та інформацією тощо. Як правило сайти такого тип мають якісь свої

правила та першочерговість представлення інформації. Важливим також є спосіб представлення, який в тому числі залежить від цільової та вікової аудиторії потенційних користувачів даного програмного продукту. В першу чергу важливо вдало розмістити і представити якнайдокладніше каталог товарів, їх вартість, варіантів оплати (при отриманні та огляді чи по передоплаті) та доставки (поштова чи кур'єром), можливості розстрочки та індивідуальних замовлень, і чим оптимальніше та вдаліше буде враховано всі особливості, тим зручніше користувачеві буде здійснити таке замовлення і, що надзвичайно важливо, повернутися за чимось повторно.

Таким чином, підсумовуючи хочу перелічити основні переваги такого виду програмного продукту:

1 –робочий графік 24/7, впродовж всього календарного року, без будь яких вихідних чи перерв, не роблячи перерви на святкові дні, без врахування людського фактору (лікарняні, відпустки), мінімізуючи можливість безпосереднього контакту представника продавця та покупця;

2 –можливість здійснення покупки з будь-якої та в будь-якій точці світу, без врахування часових поясів та необхідності безпосередньо перебувати на місці;

3 –повністю автономна робота;

4 –відсутність будь-якого обмеження на площу, оскільки вона віртуальна і дозволяє розташування необмеженої кількості та видів товарів чи послуг;

5 –можливість заробити через здачу в оренду своїх торгових площ, хоч вони і є віртуальними;

6 –можливість спілкування з потенційними клієнтами, які можуть знаходитись в будь-якій точці планети, в режимі on-line та без врахування часових поясів;

7 –терміни та занальна вартість реалізації діяльності такого виду значно нижчі, аніж класичного стаціонарного магазину;

8 –значна економія часу, коштів та сил в силу уникнення необхідності оформлення через державні установи (дозволи та ліцензії). Відсутність численних

перевірок пожежних інспекторів, санепідемстанції, податкової, служби з охорони праці тощо;

9 –розвиток сучасних технологій та можливостей для створення мобільних інструментів з можливістю доступу до Всесітньої мережі (доступ без необхідності за діяння ноутбука чи стаціонарного комп'ютера через звичайний мобільний телефон);

10 – можливість представлення товарів незважаючи на об'єм та їх розмір , яке навіть неможливо порівняти чи прирівняти до будь-якого з існуючих магазинів.

В принципі такий перелік ще можна продовжити, але вважаю, що і цього є достатньо і є підтвердженням актуальності такої розробки, проте з використанням особливостей раціонального уніфікованого підходу та враховуючи специфіку та вимоги сьогодення.

1.1.4 Обґрунтування вибору уніфікованої мови моделювання

Уніфікована мова моделювання (UML) – це універсальна мова візуального моделювання систем. Хоча найчастіше UML асоціюється з моделюванням ОО програмних систем, вона має набагато ширший спектр застосування, дякуючи своїй розширюваності [18].

UML увібрав кращі новітні технічні прийоми моделювання і розробки програмного забезпечення. По суті, мова UML була задумана так, щоб її можна було реалізувати її ж інструментальними засобами. Фактично, це визнання того, що великі сучасні програмні системи, як правило потребують інструментальної підтримки. UML-діаграми легко сприймаються і при цьому без зусиль генеруються комп'ютерами.

Важливо розуміти, що UML не пропонує будь-якої методології моделювання. Звісно, деякі методологічні аспекти самі собою розуміються елементами-складовими моделі UML, але сам UML представляє з себе лише візуальний синтаксис, який можна використати для створення моделей.

UML не прив'язаний до якоїсь конкретної методології чи життєвого циклу. На справді, він може використовуватись зі всіма існуючими методологіями. UP використовує UML в якості базового синтакмиму візуального моделювання. Відповідно, UP можна розглядати як пріоритетний метод для UML, оскільки він найкраще адаптований до неї. Однак, сам UML здатен надати (і надає) підтримку візуального моделювання іншим методам. Конкретним прикладом метода, який використовує UML в якості візуального синтаксису є метод OPEN(Object-oriented Process, Enviroment, and Notation).

Незмінна ціль UML та UP – сприяти обєднанню всього найкращого в досвіді розробки програмного забезпечення останнього десятиліття. Для цього UML і UP уніфікують досвід попередніх мов віртуального моделювання і процесів розробки програмного забезпечення найбільш оптимальним чином.

Аналізуючи відкриті джерела, можна зробити висновки, що до 1994 року в світі IT царював хаос. Існувало кілька конкуруючих мов і методологій візуального модулювання, кожна з власними перевагами й недоліками, сторонниками і противниками.

В 1994 році Буч і Рамбо об'єднуються в компанії Rational Corporation для роботи над UML , і незабаром UML стає відкритим промисловим стандартом. В 1997 році Група управління об'єктами (OMG) приймає мову UML і народжується перша відкрита мова візуального моделювання яка задовольняє усі стандарти [20].

У 2000 році з'являється версія UML 1.4 як суттєве розширення UML, досягнуте додаванням семантики дій. Вона описує поведінку набору елементарних дій, які можуть бути реалізовані конкретними мовами дій. В 2005 році, була завершена специфікація UML 2.0. Тепер UML - повністю сформована мова моделювання. В UML 2.0 з'явилось багато нових візуальних синтаксичних

структур. Фактично самі глибок зміни стосуються мета моделі UML, з якою розробними моделей напряму не матимуть справи. Метамодель UML – це модель мови UML, яка виражена в підмножині UML. Вона строго визначає синтаксис і семантику всіх елементів моделювання UML. Загалом, філософію UML, яка склалася у ці часи можна визначити так: вона бере найкраще з того, що було до неї, інтегрує і використовує в якості основи [21].

1.1.5 Обґрунтування вибору та переваги при застосуванні уніфікованого процесу

Уніфікований процес (UP) – це методологія [22]. Вона вказує на виконавців, дії та артефакти, які необхідно використати, чи створити для моделювання програмної системи.

Процес розробки ПЗ(Software Engeneering Process), визначає хто, що, коли і як в розробці ПЗ, тобто процес в якому вимоги користувача перетворюються в ПЗ.

Уніфікований процес розробки ПЗ – це SEP від авторів UML [22]. Зазвичай його називають Уніфікованим процесом, або UP. Тобто сам UML – це наглядна частина проекту, а UP – це процес. По суті, UP – введений в практику і перевірений метод розробки програмного забезпечення, який об’єднує в собі найкращі якості своїх попередників.

Датою народження UP можна вважати 1967 рік, коли компанія Ericcson [23] почала використовувати метод-предтечу компонентно-орієнтованої розробки. Принципом було те що, менші блоки зв’язувались між собою, утворюючи блоки більшого розміру, які вже складали всю систему.

Наступним великим кроком в розвитку моделювання ПЗ був вихід у світ мови специфікації і опису (SDL) у 1980 році [23]. В 1992 р. він був розширений і став об’єктно-орієнтованою мовою з класами і наслідуванням. Ця мова була розроблена для відображення поведінки телекомунікаційних систем.

Батьком UP часто називають Айвара Джекобсона [24], який у 1987 році засновує компанію Objectory AB, і розробляє процес розробки ПЗ, названий Objectory. Можливо, самим важливим нововведенням того часу було те, що сам по собі Objectory розглядався як система. Коли в 1995 р. компанія Rational купила Objectory AB, Джекобсон зайнявся об'єднанням процесу Objectory з розробками компанії Rational.

Rational Oblectory Process – результат об'єднання підходів до розробки програмного забезпечення обох корпорацій. Було удосконалено реалізацію, тестування, управління проектом, розгортання, конфігураці.

У 1998 році виходить RUP(Rational Unified Process) – результат злиття різних компаній у Rational, та спільної розробки. З тих пір, світ побачили багато версій RUP, кожна з яких краща за попередню [25].

UP базується на трьох аксіомах. Він є:

1 –Керованим вимогами та ризиками;
2 –Архітектурно-центричним;
3 –Ітеративним та інкрементним.

Ризик – ще один керуючий механізм UP, оскільки якщо ви не будете атакувати ризики то вони атакуватимуть вас. UP вирішує цю проблему, заклавши аналіз ризиків в основу створення ПЗ.

Розвиток якісної архітектури системи – ще одна аксіома UP. Очевидно, що якісна розроблена архітектура забезпечить створення працездатної системи, а не просто спішно скомпонованого набору коду.

Врешті, UP є ітеративним та інкрементним. Ітеративний аспект означає, що проект розбивається на невеликі під проекти(ітерації), які забезпечують функціональність системи по частинах чи інкрементам, створюючи повнофункціональної системи. Фактично до ключових робочих потоків UP, таких як аналіз, ми повертаємось по декілька раз протягом проекту.

Щоб зрозуміти UP, треба зрозуміти ітерації. Ідея, по суті проста: розбити великий проект на декілька менших «міні-проектів», якими легче керувати і

успішно виконати. Кожний з такий «міні-проєктів» і є ітерацією. Основний момент: кожна ітерація включає всі елементи звичайного проєкту:

1 –Планування;

2 –Аналіз і проектування;

3 –Конструювання;

4 –Інтеграція і тестування;

5 –Версія для внутрішнього чи зовнішнього використання.

В кожні ітерації п’ять основних потоків визначають, що повинно бути зроблено і які навички для цього необхідні. Наряду з п’ятьма основними потоками, будуть присутні й інші, такі як планування, оцінка, і все що характерно для цієї конкретної ітерації.

До п’яти основних робочих потоків відносяться:

1 –Визначення вимог – збір даних про те, що повинна робити система;

2 –Аналіз вимог – уточнення і структурування вимог;

3 –Проєктування – реалізація вимог в архітектурі системи;

4 –Реалізація – конструювання програмного забезпечення;

5 –Тестування – перевірка, чи відповідає реалізація всім вимогам.

Кожна ітерація формує базову версію. Це версія для внутрішнього користування набору розглянутих і затверджених артефактів, на даній ітерації.

Кожна базова версія:

- представляє базу для подальшої розробки та розгляду;
- може змінюватись тільки через формальні процедури управління конфігурацією та намірами.

Життєвий цикл про’кту розподілений на чотири фази: “Початок”, “Уточнення”, “Конструювання” та “Впровадження”, кожна з яких має свої контрольні точки. В кожній фазі може буди одна чи більше ітерацій, в кожній ітерації виконується п’ять основних і скільки завгодно додаткових робочих

потоків. Точне число ітерацій у фазі залежить від розміру проєкту, але кожна ітерація повинна тривати не більше двох-трьох місяців.

UP складається з послідовності чотирьох фаз, кожна з яких завершується важливою контрольною точкою:

1. Початок – цілі життєвого циклу;
2. Уточнення – архітектура життєвого циклу;
3. Конструювання – базова функціональність;
4. Впровадження – випуск продукції.

На рисунку 1.3 представлено графік фаз та дисциплін RUP.

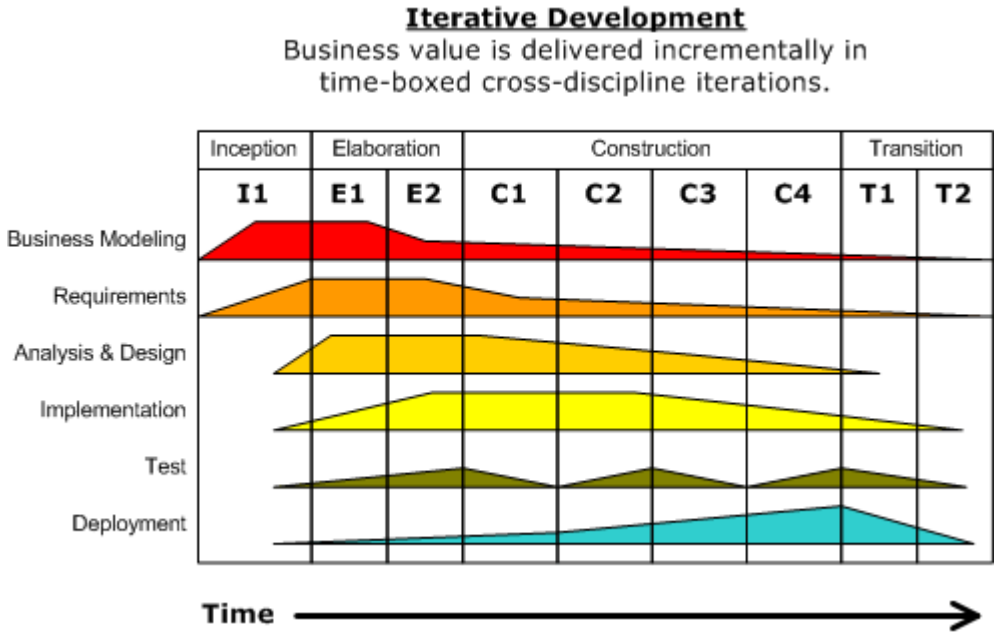

Рисунок 1.3 – Фази та дисципліни RUP [26]

Ціль фази “Початок” – зсунути проєкт з “мертвої точки”. Початок включає:

- Обґрунтування здійснимості – може включати розробку технічного прототипу з метою перевірки правильності технологічних рішень чи концептуального прототипу для перевірки бізнес-вимог;
- Розробка економічного обґрунтування для демонстрації того, що проект забезпечить виражену в кількісному відношення економічну вигоду;
- Визначення основних вимог для створення предметної області системи;

- Виявлення найбільш загрозливих ризиків.

Цілі фази “Уточнення” можна описати наступним чином:

- Створення виконуваної базової версії архітектури;
- Деталізація оцінки ризиків;
- Визначення атрибутів якості;
- Виявлення варіантів використання, які складають до 80% функціональних вимог;
- Створення детального плану фази Конструювання;
- Формулювання пропозиції, яка включає ресурси, час, обладнання, штат та вартість.

Основна ціль фази “Уточнення” – створення базової версії архітектури. Загальна увага у фазі Уточнення направлена на робочі потоки визначення вимог, аналізу і проєктування.

Ціль фази “Конструювання” – завершити визначення вимог, аналіз та проєктування і розвинути виконувану базову версію архітектури, створену в фазі Уточнення, в завершену систему. Головна проблема Уточнення – підтримка цілісності архітектури системи. Дуже часто при встановленні термінів поставки, і переході до написання коду починається відкидання формальностей, що призводить до порушенню архітектурного представлення, низької якості кінцевої системи і високим затратам на обслуговування. Основну увагу на цьому етапі концентрують на реалізації проєкту.

Впровадження починається коли завершено бета-тестування і система повністю розгорнута. Сюди відноситься усунення всіх дефектів, знайдених під час бета-тестування, і підготовка до масового випуску програмного забезпечення на всі користувацькі сайти. Цілі можна охарактеризувати так:

- Виправлення дефектів;
- Підготовка користувацьких сайтів під нове ПЗ;

- Настройка роботоздатності програмного забезпечення по користувацьких сайтах;
- Зміна програмного забезпечення у випадку виникнення непередбачуваних проблем;
- Створення посібників для користувачів, та іншої документації;
- Представлення користувачам консультацій;
- Проведення після проєктного аналізу.

Головна увага концентрується на робочих потоках реалізації і тестування. Для виправлення всіх помилок проектування, виявлених при бета-тестуванні, виконується суттєвий об’єм проектування. Фази виявлення вимог та аналізу повинні бути майже не задіяними. В противному випадку з проектом не все в порядку.

1.2 Визначення вимог до системи

1.2.1 Робочий потік визначення вимог системи

Вимогу можна визначити як «детальний опис того, що повинно бути реалізоване».

Вимоги – це основа всіх систем. Це формування того, що повинна робити система. Це, по суті, тільки виклад того, що система повинна робити, але не того *як* вона це буде робити.

Формат формування вимог:

<id> <система> shall <дія>

UML рекомендує дуже простий формат формування вимог. Кожна вимога має унікальний ідентифікатор, ключове слово shall і опис дії. Перевага такої структури в спрощенні завдання синтаксичного розбору вимог для інструментів управління вимогами.

Існує два основних типи вимог:

- Функціональні вимоги – яку поведінку повинна пропонувати система;
- Не функціональні вимоги – особлива властивість чи обмеження, яке накладається на систему.

Функціональні вимоги – це формування того, що повинна робити система, опис призначення системи.

Не функціональні вимоги – обмеження, яке накладається на систему.

У кожної вимоги може бути ряд атрибутів, які фіксують додаткову інформацію про вимогу.

Кожен атрибут вимоги має описане імя та значення. Наприклад, у вимоги може бути атрибут dueDate (дата оплати), значенням якого є дата виконання даної вимоги. Конкретний набір атрибутів залежить від природи і потреб проєкту і може змінюватись в залежності від типу вимоги. Напевно самим розповсюдженим атрибутом є priority (пріоритет). Його значення визначає пріоритет вимоги відносно інших вимог.

Головне, при визначенні атрибутів вимог – зробити їх максимально простими. Якщо атрибут не приносить користі, то не потрібно його використовувати.

Вимоги виходять з контексту модульованої системи. В цей контекст входять безпосередні користувачі системи, інші зацікавлені сторони (наприклад керівники, спеціалісти обслуговування, встановлювальними), інші системи, з якими взаємодіє дана система, апаратні прилади, з якими взаємодіє дана система, правові та регулятивні обмеження, технічні обмеження, фінансові цілі.

Як правило, розробка вимог починається з документу, який описує в загальних рисах те, що збирається робити система, і які послуги вона буде надавати ряду зацікавлених сторін. Призначення такого документу – визначити найбільш важливі цілі системи з точки зору зацікавлених сторін.

Проведення інтерв’ю з зацікавленими сторонами – самий прямий спосіб збору вимог. Зазвичай більш повну інформацію можна отримати при проведенні інтерв’ю один на один.

Анкети можуть бути хорошим доповненням до інтерв'ю. Вони добре підходять для отримання відповідей на конкретні закриті питаня, можна виділити з інтерв'ю ключові питання, об'єднати їх в анкету і розкинути на більш широку аудиторію.

Семінар – один з найефективніших способів збору інформації, яка відноситься до вимог.

Визначення вимог – це ітеративний процес, в якому вимоги виявляються по мірі уточнення розуміння потреб зацікавлених сторін.

У процесі визначення вимог було визначено такі функціональні вимоги:

- сайт повинен надавати покупцям можливість перегляду товару;
- сайт повинен надавати покупцям можливість додавання товару до кошика;
- сайт повинен надавати покупцям можливість купівлі товару(-ів);
- сайт повинен надавати покупцям декілька способів оплати товару;
- сайт повинен забезпечити можливість пошуку товарів;
- сайт повинен включати в себе інформацію про товари;
- сайт повинен включати в себе рекламну інформацію унизу сторінки;
- сайт повинен включати верхнє горизонтальне меню з категоріями товару;
- сайт повинен містити посилання на нові товари, популярні та лідери продажів на головній сторінці.

Не функціональні вимоги:

- швидкість відкривання сторінок не повинна перевищувати 1 с.;
- вести взаємодію з клієнтом тільки по захищених каналах зв'язку;
- підтримка багатомовності;
- підтримка багатовалютності;
- можливість організації доставки.

1.2.2 Моделювання варіантів використання

Моделювання варіантів використання – одна з форм визначення вимог. Зазвичай цей процес проходить наступним чином:

- визначаються границі потенційної інформаційної системи.
- визначаються актори;
- визначаються варіанти використання;

Робота триває доти (через повторенні попередніх кроків) допоки варіанти використання, актори та границі розроблюваної системи не буде стабілізовано.

Границею системи можна назвати прямокутником, який окреслює варіанти використання для визначення краю чи границі системи. В UML 2 цю границю називають контекстом системи (subject).

Актори – ролі, які виконують люди чи сутності, якими використовується система.

Варіанти використання – те, що актори можуть робити із системою.

Відносини – відносини між акторами і варіантами використання.

Розглянемо дані для виявлення акторів та варіантів використання:

- Бізнес-модель – може бути представлена в розпорядження розробників моделі системи, але це не завжди виконується.
- Модель вимог – ця вимога – хороший вихідний матеріал для процесу моделювання варіантів використання.
- Списоком можливостей називають ще набором потенціальних вимог, котрі можуть представлятися в різних формах, зокрема як загальний список(vision) проєкту чи щось-небудь подібне.

Актори – це ролі, які виконують ті сутності, що напряму взаємодіять із системою.

Для кожного актора зазвичай потрібно додавати супровідний опис (короткий опис), яким роз’яснюється що чи хто є даним актором та що він із себе представляє. В позначенні акторів, класів можуть представлятись і акторові

атрибути та події. Важливо усвідомлювати, що актори завжди є зовнішніми якщо ми говоримо стосовно системи.

Варіанти використання – це щось, що мусить виконуватися системою, за побажанням актора.

Варіанти використання завжди описуються та ініціюються актором з його авторової точки зору.

Ідентифікацію варіантів використання найкраще розпочинати з акторових списків, після чого роздивитись, як кожний актор «планує» скористатися системою. Кожному варіанту використання повинно бути присвоєно коротке описове ім’я.

Моделювання варіантів використання – ітеративний процес, виконується шляхом поетапного уточнення. Все починається з імені варіанту використання, деталі додаються пізніше.

Глосарій проєкту – можливо, один з найважливіших артефактів проекту. Глосарій забезпечує словник ключових термінів і визначень. Він повинен бути зрозумілим усім учасникам проекту, включаючи всі зацікавлені сторони.

Також глосарій повинен включати всі синоніми й омоніми.

Синоніми – різні слова, які означають одне і те ж. Потрібно вибрати одне і притримуватись його.

Омоніми – це слова, одинакові за звучанням, проте різні за значенням. В цьому випадку завжди виникають проблеми в спілкуванні, оскільки сторони розмовляють різними мовами, при цьому впевнені що говорять про одне і те ж. Спосіб вирішення – вибрати одне значення для одного терміна, а для інших омонімів, ввести нові терміни.

Зазвичай уживають такий шаблон для специфікації варіантів використання:

- Ім’я варіанту використання;
- ID варіанту використання;
- Короткий опис – абзац, в якому викладена мета проекту;
- Актори, які задіяні в варіанті використання;

- Передумови – умови, які повинні вконуватись, щоб варіант використання міг здійснитись.
- Основний потік – кроки виконання варіанту використання;
- Постумови – умови, які повинні виконатись по закінченню варіанту використання;
- Альтернативні потоки – список альтернативних основному потоку подій.

Узагальнення акторів можна ужити тоді, коли два чи більше актори одинаково взаємодіють та поводяться в деяких аспектах. Тоді можна вказати нового абстрактного актора з окремими властивостями, який буде відображати узагальнення, вживаючи зв’язок наслідування.

Узагальнення варіантів використання використовується коли один чи більше варіант використання насправді є спеціалізацією більшого варіанту використання. Як і у випадку з акторами, цей прийом варто застосовувати лише коли він спрощує модель варіантів використання.

В узагальненні варіантів використання їхні дочірні варіанти використання представляють більш спеціалізовані форми їх батьків. Потомки можуть наслідувати можливості батьківського варіанту використання, вводити нові можливості, міняти унаслідувані можливості. Дочірній варіант використання автоматично наслідує всі можливості свого батька.

1.2.3 Модель варіантів використання системи користувачами

Відношення «include» виносить кроки, спільні для кількох варіантів використання, в окремий варіанти використання, який потім включається в решту. Включаючий варіант використання називається базовим а той який включається – включаємим. Включаємий варіант використання надає свою поведінку своєму базовому варіанту використання. В базовому варіанті використання необхідно

точно вказати місце, де повинно бути включено поведінку включаємого варіант використання. Базовий варіанти використання вважається незавершеним без усіх його включаємих варіантів використання.

Відношення «extend» - спосіб введення нової поведінки в існуючий варіант використання. Базовий варіант використання представляє набір точок розширення(extention points) – точок входу, в які можна додати нову поведінку. А розширяючий варіант використання представляє ряд сегментів вставки , які можна ввести в базовий варіант використання в місця, вказані точками входу. Відношення «extend» можна використати для того, щоб точно вказати які саме точки розширення базового варіанту використання підлягають розширенню.

Розширяючі варіанти використанняи зазвичай не є повними варіантами використання, тому, як правило їх екземпляр не може бути створено. Зазвичай, вони включають всього один чи декілька фрагментів поведінки – сегменти вставки. Відношення «extend» визначає точку розширення в базовому варіанті використання, в який буде введено сегмент вставки. Тут працюють наступні правила:

1. Відношення «extend» повинно визначати одну чи декілька точок розширення базового варіанту використання. В іншому випадку вважається що відношення «extend» відноситься до всіх точок розширення.

2. В розширяючому варіанті використання повинно бути стільки ж сегментів вставки, скільки точок розширення визначено в відношення «extend».

3. Два розширяючих варіанти використання можуть розширяти один базовий варіант використання в одній і тій же точці розширення.

4. В розширяючому варіанті використання може бути декілька сегментів вставки. Це корисно в тих випадках, коли не вдається повністю реалізувати розширення в одному сегменті через те, що необхідно повернутись і щось зробити в основному потоці базового варіанту використання.

Додаткові можливості застосовуються тільки тоді, коли вони спрощують модель варіантів використання.Враховуючи досвід моделювання варіантів використання в різних компаніях, можна зробити наступні висновки:

1. Зазвичай зацікавлені сторони після недовгого тренування і навчання можуть без зусиль розібратись в акторах та варіантах використання;

2. Зацікавленим сторонам важче сприймати узагальнення акторів;

3. Широке використання «include» погіршує розуміння моделі варіантів використання;

4. У зацікавлених сторін виникають великі труднощі з розумінням відношення «extend» навіть після детальних пояснень;

5. Як не дивно, багато розробників об'єктних моделей невірно розуміють семантику відношення «extend»;

Узагальнення варіантів використання варто уживати, тільки якщо в системі використовуються абстрактні батьківські варіанти використання, в іншому випадку це сильно ускладнює дочірні варіанти використання.

Під час проектування визначено таких акторів:

- покупець;
- оператор сайту;

На рисунку 1.4 представлена розроблена Модель-діаграма варіантів використання користувачами системи.

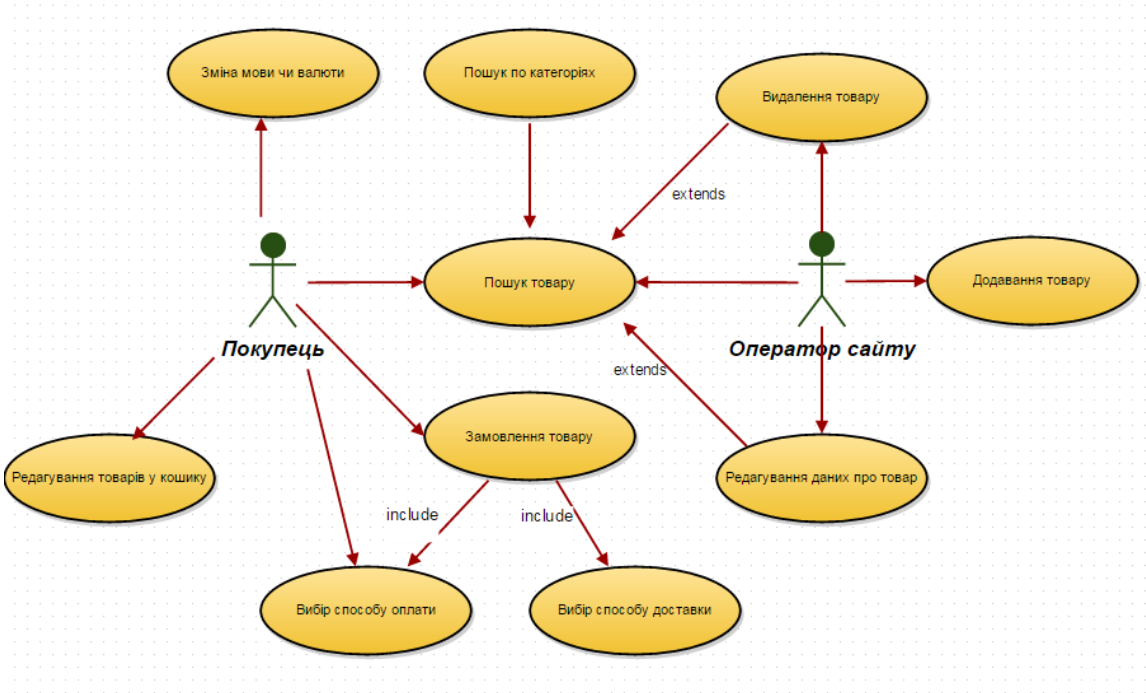

Рисунок 1.4 – Модель-діаграма варіантів використання користувачами системи

1.3 Проєктування архітектури програмного забезпечення

1.3.1 Робочий потік проектування

Проєктування – основна діяльність при моделюванні в останній частині фази Уточнення і першій частині фази Конструювання. Аналіз та проектування в деякій степені можуть проходити паралельно. Одна команда повинна провести артефакт від аналізу до проєктування. Об’єктно-орієнтовані проектувальники повинні основну увагу відводити найголовнішим питанням проектування, таким як архітектурі розподілених компонентів – політики і стандарти повинні вводитись для вирішення тактично важливих питань проєктування.

Проєктна модель включає:

- проектні підсистеми;
- проектні класи;
- інтерфейси;
- реалізації варіантів використання – проектні;
- діаграму розгортання(в першому наближенні).

Відношення відображення існують між: проєктною та аналітичною моделями, одною та більше проектними підсистемами і пакетом аналізу. Варто підтримувати дві окремі моделі, аналітичну і проектну, якщо система велика, складна, стратегічно важлива, піддається частим змінам з довгим строком служби, та для її розробки потрібні зовнішні ресурси.

Діяльність UP Проектування архітектури – це ітеративний процес, який проходить в кінці фази Уточнення та на початку Конструювання. В ході процесу створюються та описуються в загальних рисах артефакти, які в подальшому конкретизуються.

Проєктні класи – це класи, настільки повно описані, що можуть бути реалізовані.

Існує два джерела проектних класів: предметна область та уточнення класів аналізу. Один клас аналізу може перетворитися в нуль, один чи більше проєктних класів.

1.3.2 Наслідування

Наслідування повинно використовуватись тільки тоді, коли між двома класами існує чітке відношення «є», або з метою повторного використання коду.

Недоліки:

1. Це сама строга зі всіх можливих форм зв’язаності між двома класами; в рамках ієрархії наслідування інкапсуляція слабенька, що призводить до проблеми «ламкості базового класу» - зміни базового класу передаються вниз по ієрархії.
2. Дуже негнучка форма відношень в більшості мов програмування – відношення встановлюється під час компіляції і залишається незмінним під час виконання.

Наслідування повинно застосовуватись, коли необхідно унаслідувати деяку реалізацію. Реалізація інтерфейсу повинна застосовуватись, коли необхідно визначити контракт.

З всіх широко використовуваних об’єктно-орієнтованих мов програмування на теперішній час шаблони підтримують тільки C++ і Java. Шаблони дозволяють «параметризувати» тип – шаблон створюється шляхом визначення типу за допомогою формальних параметрів, і екземпляри шаблону створюються через зв’язування параметрів з конкретними значеннями.

Уточнення аналітичних відносин до відносин рівня проектування включає:

- уточнення асоціацій до агрегації або композитної агрегації в відповідних випадках;
- реалізацію класів-асоціацій;

- реалізацію асоціацій один-до-багатьох;
- реалізацію асоціацій багато-до-одного;
- реалізацію асоціацій багато-до-багатьох;
- реалізацію двонаправлених асоціацій;
- введення можливості навігації;
- введення кратності на обох кінцях асоціації;
- введення імені ролі на обох кінцях асоціації, чи принаймні на цільовому кінці;
- використання структурованих класифікаторів.

Агрегація и композиція це відношення типу цілісність, де об’єкти одного класу грають роль цілого, але агрегата, а об’єкти іншого класу грають роль частин. Ціле використовує сервіси частин, частини обслуговують запити цілого, ціле є домінуючою стороною відносин і частина грає більш пасивну роль.

Ці відношення транзитивні – якщо С є частиною В і В є частиною А, тоді С є частиною А.

Ці відношення асиметричні: ціле ніколи не може бути частиною самого себе; цикли в схемі агрегації недопустимі.

Композиція це строга форма агрегації де:

- частини одночасно належать тільки одному композиту;
- композит володіє винятковою відповідальністю за керування всіма своїми
- частинами – за їх створення та ліквідування;
- композит також може звільняти частини, якщо відповідальність за них бере на себе інший об’єкт;
- якщо композит знищується, він повинен знищити всі свої частини і передати відповідальність за них якому-небудь іншому об’єкту;
- кожна частина належить тільки одному композиту, тому можливі тільки ієрархії композиції – мереж композиції не існує.

1.3.3 Проєктування інтерфейсів та компонентів системи

Інтерфейси забезпечують можливість проєктувати програмне забезпечення як контракт, а не як конкретну реалізацію.

Діяльність UP проєктування підсистеми полягає в розділі системи на підсистеми – по можливості максимально незалежні частини. Посередниками у взаємодії систем виступають інтерфейси.

Інтерфейс визначає іменований набір відкритих властивостей. Інтерфейси визначають опис функціональності від її реалізації, та можуть бути прикріплені до класів, підсистем, компонентів і будь-якого іншого класифікатора і визначає пропонуємі ним сервіси. Якщо класифікатор в підсистемі реалізує відкритий інтерфейс, підсистема чи компонент також реалізує відкритий інтерфейс. Все, що реалізує інтерфейс, погоджується притримуватись контракту, визначеного набором операцій, що описані в інтерфейсі.

Під час проектування реалізації об’єднуються конкретні класи. Для збереження простоти (не гнучкості), необхідно проектувати реалізацію.

Проєктування контракту:

- клас об’єднується з інтерфейсом, у якого можу бути багато можливих реалізацій;
- для забезпечення гнучкості (що, скоріш за все підвищить складність), необхідно проектувати контракт.

Надаваний інтерфейс – інтерфейс, надаваний класифікатором, який реалізує інтерфейс. Також класифікатору необхідний інший класифікатор який теж буде реалізувати цей інтерфейс. Залежність відображається на інтерфейс, представлений у вигляді класу, або використовується роз’єм зборки. Роз’єм зборки – об’єднує надаваний і необхідний інтерфейси.

Компонент – модульна частина системи, що інкапсулює її вміст. Реалізація компонента замінювана в рамках його оточення та може містити атрибути й операції а також брати участь у відносинах. Компоненти, як і класи, також мають

свою внутрішню структуру. Зовнішня поведінка компонента повністю визначається його надаваними і необхідними інтерфейсами. Компоненти являють собою один чи більше артефактів.

Компонентно-орієнтована розробка займається побудоваю програмного забезпечення з частин, які підключаються щоб надати компонентам можливість підключення застосовуються інтерфейси. Проектування інтерфейсу дозволяє використовувати багато різних реалізацій багатьма різними компонентами.

Компоненти можуть являти собою фізичну сутність або логічну сутність.

Стандартні стереотипи компонентів:

- «buildComponent» – компонент, який визначає набір сутностей для організаційних чи системних цілей розробки;
- «entity» – компонент постійної інформації, який представляє бізнес поняття;
- «implementation» – компонент, який не містить власної специфікації; він є реалізацією окремого компоненту, визначеного стереотипом «specification», з яким має відношення залежності;
- «specification» – класифікатор, який визначає набір об’єктів без опису їх фізичної реалізації – наприклад, компонент, позначений типом «specification» має тільки представляємі і вимагаємі інтерфейси, але не має реалізуючих класифікаторів;
- «process» – орієнтований на транзакції компонент;
- «service» – функціональний компонент, який вираховує значення, не має стану;
- «subsystem» – елемент ієрархічної декомпозиції великих систем.

На рисунку 1.6 приведено приклад діаграми класів для системи купівлі та запису інформації про купівлю.

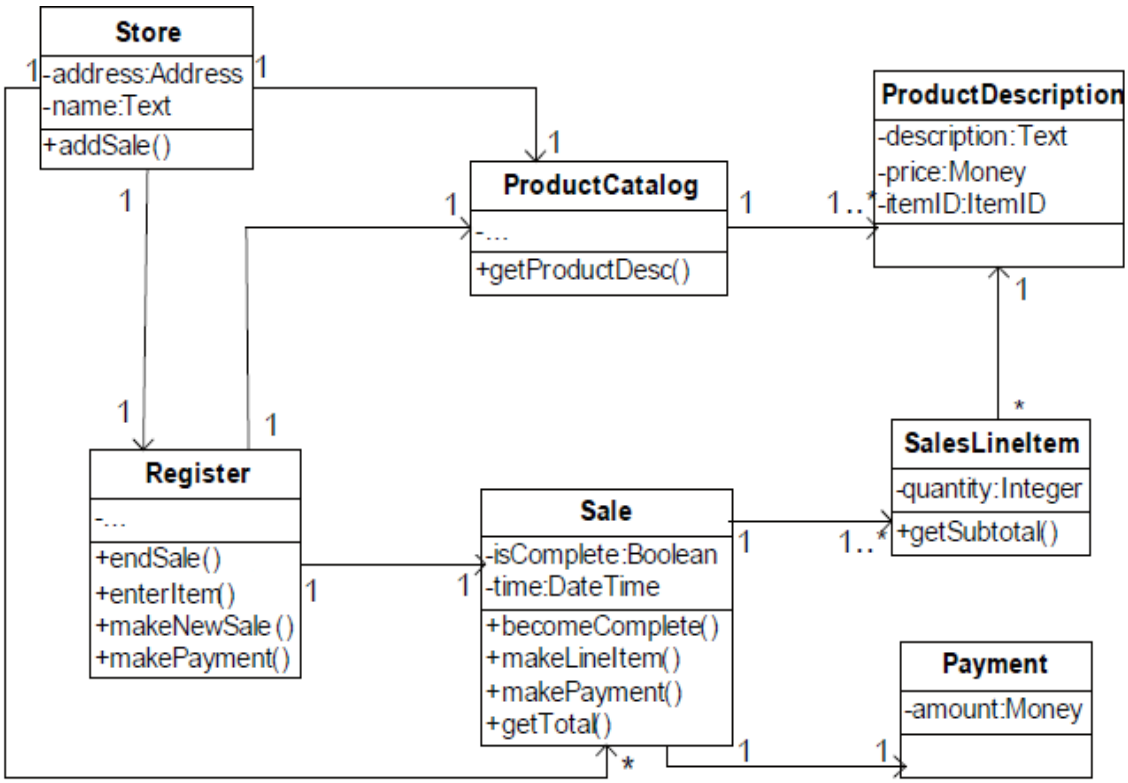

Рисунок 1.6 Діаграма класів системи придбання одиниць

1.3.4 Реалізація варіантів використання на етапі проєктування

Реалізація варіантів використання на етапі проєктування – це насправді розширення реалізації варіантів використання, створеного при аналізі. Діяльність UP Проектування полягає у виявленні проектних класів, інтерфейсів та компонентів, взаємодія яких забезпечує поведінку, описану варіантом використання.

Проєктні реалізації варіантів використання – це взаємодія проектних об’єктів і класів, направлена на реалізацію варіантів використання.

Вона включає проектні діаграми взаємодії – уточнені аналітичні діаграми взаємодії – уточнені аналітичні діаграми взаємодії та проектні діаграми класів – уточнені аналітичні діаграми класів.

Діаграми взаємодії можуть використовуватись в проєктування для моделювання центральних механізмів, таких як збереження об’єктів; ці механізми можуть охоплювати багато варіантів використанняі

Діаграми послідовностей бувають наступних типів:

- par – всі операнди виконуються паралельно;
- critical – операнд виконується автоматично без переривань.

Комунікаційні діаграми: порядковий номер додається міткою для позначення потоку керування.

Діаграми діяльності:

- розгалуження;
- об’єднання.

Діаграми взаємодій підсистем показують взаємодії між різними частинами системи на високому рівні абстракції:

- в них можуть входити актори, підсистеми, компоненти та класи;
- частини підсистем (наприклад, представляємі інтерфейси) можуть відображатися в прямокутниках, які прилягають знизу до піктограми підсистеми.

Тимчасові діаграми – моделюють тимчасові обмеження:

- дуже корисні для моделювання апаратних систем реального часу та встроєних систем;
- час збільшується протягом горизонтальної осі зліва направо;
- лінія життя, стан і умови розташовуються по вертикалі;
- переходи між станами та умовами відображаються у вигляді графіку;
- можна показати тимчасові обмеження і події;
- компактна форма тимчасової діаграми робить акцент на відносному часі.

На рисунку 1.7 приведено діаграму станів для операції пошуку та редагування товару оператором сайту.

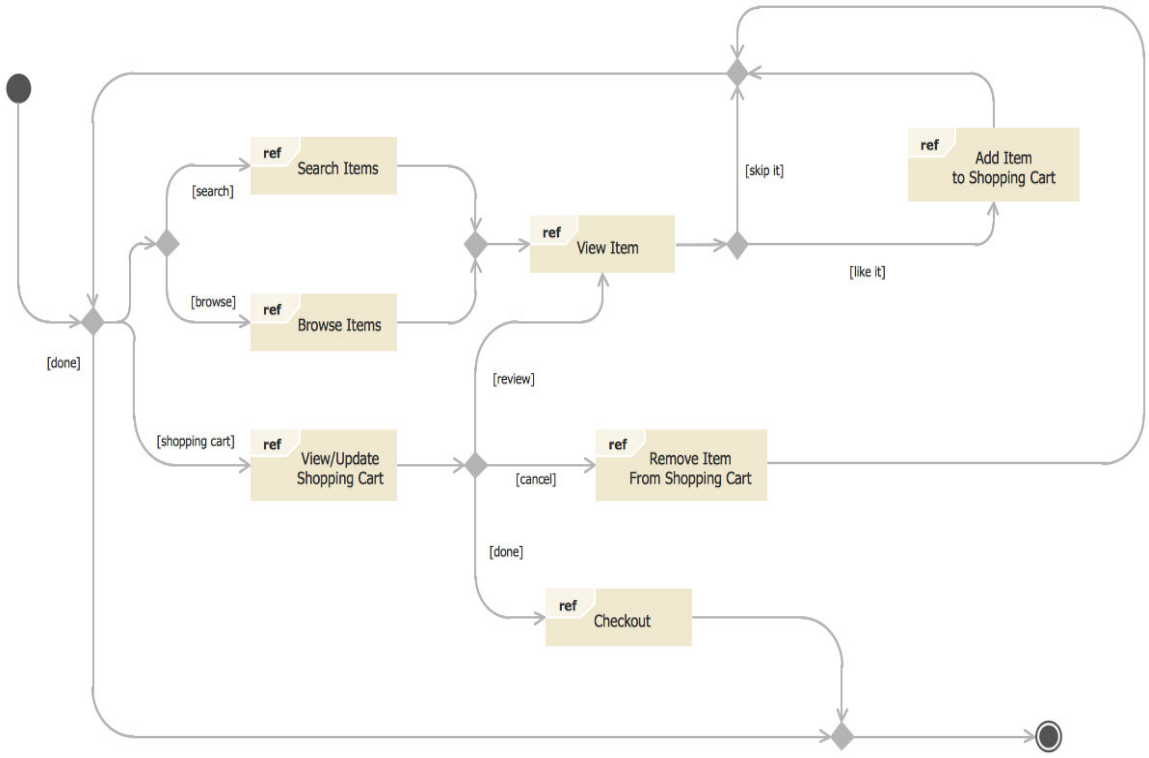

Рисунок 1.7 – Діаграма станів операції пошуку та редагування товарів

1.4 Обґрунтування вибору засобів для реалізації розробки

1.4.1 Мова розмітки гіпертекстових посилань

Відомо що з метою обмінювання інформацією у Всесвітній мережі зазвичай використовують протоколи прикладного рівня, які можуть реалізувати якийсь з прикладних сервісів (пересилка файлу, -ів, обмін гіпертекстовими даними, поштова пересилка та інше). В цілому ж варта зазначити, що одним із наймолодших та найпопулярніших сервісів Всесвітньої мережі, тенденція розвитку яких призвела і до шаленого зростання і безпосередньо її популярності, хоча сьогодні це вже не просто популярність, а спосіб життя, є WorldWideWeb (скорочено англійськоюWWW), яку було засновано на основі протоколу передачі даних прикладного рівня (англ. HyperText Transfer Protocol —

скорочено HTTP) [28]. Однією, проте принциповою відмінністю гіпертекстових документів, представлених в WorldWideWeb, на відміну від традиційно відомих гіпертекстових документів є використовувані в них зв'язки, які не обмежуються не просто до одного документу, а й навіть не обмежуються одним комп'ютером. Власне мовою розмітки гіпертексту (англійською HyperText Markup Language, скорочено HTML). Саме завдяки її можна домогтись отримання широких можливостей для форматування і структурної розмітки документів, створення зв'язу, -ів між документами, засобів для включення інформації графічного та мультимедійного типу. Видимими HyperText Markup Language документи стають за рахунок функціонування браузера, який є спеціалізованою програмою. Найпоширенішими сьогодні є браузери MozillaFirefox (безкоштовний браузер, що підтримується Mozilla) та InternetExplorer – графічний веб-браузер (скорочено MSIE чи IE; /aɪ iː/) від відомої компанії Microsoft. MozillaFirefox реалізації є доступними фактично для усіх для усіх актуальний популярних платформ програмного та апаратного типів, а реалізації MSIE чи IE; /aɪ iː/ є доступними для усіх Windows-платформ, Macintosh та кількох Unix-систем комерційного характеру.

Таким чином в тезі<html> знаходиться текст усього документу, який містить розбито) на заголовок (<head>) та тіло (<body>), тобто складається з двох частин.

Теги заголовку можуть містити різні параметри, наприклад, назва документу, тобто ті парамтри, які при відображенн документу використовує браузер. Якщо говорити про <body>, то тут міститься інформація, яку безпосередньо призначена для користувача, він її бачить.

Коли виникає необхідність посилання до другого (іншого) документу, то це реалізується з використанням тегу..., у якому URL є повною чи відносною адресою документу. Тоді, при наявності ув'язненого в тег <a>, тексту, його як правило виокремлюють з використанням підкреслення чи зміною кольору, а після активізації мишкою на таке посилання через браузер відкривається документ, з адресою якого співпадає адреса вказана в href-

параметрі. Для можливості вставляти графічні зображення використовується тег <img src="URL">.

1.4.2 Переваги та недоліки розширювальної мови розмітки Extensible Markup Language [29]

Відповідним фахівцям web-технологій вже зрозуміло, що передача даних в Всесвітній мережі по існуючим стандартом вже достатньо утруднена із-за їх недостатності. В свій час HTML-формат, який свого часу сприймайся як прорив в напрямку представлення(відображення) вмісту для інтернет-вузлів, вже не задовільняє сучасних вимог, адже його застосування не представляє засобів, щоби ефективно описати та управляти передавальними даними., оскільки за допомогою нього можна отримати лише опис того, яким чином потрібно відобразити дані перед кінцевим користувачем на його екрані. Також проблемою для компаній, які працюють в цьому напрямку є й потреба в сумісному використанні різноманітних компонент, убезпечування їхньої взаємодії, надання можливості обмінюватися між ними даними.

Таким чином відсутність стандарту надання засобів для можливості інтелектуально займатися пошуком даних, обміном даних, адаптивною обробкою їх (даних) отримання стало достатньо серйозною проблемою, вирішити яку виявилось можливим лише з 1998 року, коли міжнародна організація (консорціум) W3C затвердила мову схем для документів XML (англійською eXtensible Markup Language) [29], основне призначення якої полягало в опису структурованих даних в текстовому форматі. Такий формат було створено (розроблено) як засіб передачі та збереження даних, і він має очевидну схожість з HTML.

Таким чином, нижче сформулюю можливосты XML опису та передачі наступних видів структурованих даних, а саме:

- окремих документів;
- метаданих, якими описуються вміст вузлів Всесвітньої мереді;
- об’єктів, в який містяться дані та методи по роботі з ними (елементів для управлінняACTIVEX чи об’єктів Java);
- окремих записів (для прикладу, результатів виконання запитів до БД);
- різних web-посилань як до інформаційних так до людських ресурсів Всесвітньої мереді (наприклад, адрес електронної пошти, гіпертекстових посилань тощо).

Мова XML є легкою для читання і досить простою для розуміння. Для спеціаліста, кий працював із HTML, вивчити принцип складання XML-документи не займе надто багато часу. Однією з відмінностей між HTML та XML, є можливість використовувати для XML пари тегів необмеженого набору. А кожна з таких пар (тегів) дає можливість представити що означають пов’язані в ній дані, а не як вони мають виглядати. В цьому також і суть поняття принципу розширюваності XML мови.

Якщо узагальнити і говорити в широкому сенсі то документи XML мають відповідати нижче переліченим вимогам, а саме:

- для кожного відкриваючого тега, яким визначається певна частина даних, обов’язковою умовою є наявність такого який закриватиме. Мається на увазі, що і це є ще однією відмінністю від HTML, не можна допускати пропущення закриваючих тег;
- в XML вкладка тег є строго контрольованою і важливим є порядок проходження як відкриваючих і закриваючих тегів;
- врахування в XML регістрів символів;
- в XML усю повністю інформацію, яку розташовано між відкриваючими і закриваючими тегами, розглядають у вигляді даних. Саме тому враховують усі символи форматування (всю кількість пропусків, перекладів рядків, не можна ігнорувати і табуляції, як, для прикладу, це допускається в HTML).

Крім вище наведеного для XML характерною є наявність набору

зарезервованих символів, які необхідно задіювати в документі XML, але специфічно. Деякі із спеціалістів сприймають XML як представлення нової технології інтеграції для програмних компонент.

Таким чином, до основними переваг використання XML можна віднести:

- інтеграцію даних від різних джерел. Є можливість використання XML з метою об'єднування на середньому рівні 3-рівневих web-систем та баз даних;
- локальну обробку даних. Можливість для отриманих у XML-форматі даних здійснювати їх розбір, обробку, відображення безпосередньо на стороні клієнтова без необхідності в додаткових звернення до серверної сторони;
- перегляду та маніпуляції даними в різноманітних аспектах. Можливість обробки та видимості для клієнта різними способами отриманих даних;
- можливе часткове оновлення для даних. При використання та роботі з XML є можливість оновлення лише тієї частини структурованих даних, до якої було внесено зміни, тобто не міняти всю структуру повністю.

Таким чином, вище наведений аналіз найбільш значимих, на мою думку, переваг, дають можливість вважати та сприймати XML в якості незамінного інструменту, за допомогою якого реалізується можливість розробки саме гнучких засобів метою яких є можливість пошуку інформації по базах даних, потужних трирівневих web-додатків, та таких додатків, функціоналом яких передбачено підтримку транзакцій. Якщо говорити іншими словами, то при допомозі XML реалізується можливість формування запитів до баз даних різноманітних типів структур. Це дозволить здійснити можливість пошуку інформації в багато численних, досить часто мало- чи не- сумісних одна з одною базах даних. При використанні XML для середнього рівня 3-рівневих web-додатків реалізується можливість ефективного обміну даними як між клієнтами так і між серверами робота яких пов'язана з системи призначеними для електронної комерції.

Також варта згадати і можливість, при використанні мови XML в якості засобу, метою якого є опис граматики інших мов, а також контролю за

правильністю укладання документів.

Інструменти ж призначені для опрацювання даних, які було отримано в форматі XML, можна розробляти з використанням середовищ Visual Basic,Java чи C++.

1.4.3 Аналіз особливостей застасування каскадних таблиць стилів сторінок CSS

Спеціальною мовою, яку використовують в метою відображування сторінок, написання яких виконано за допомогою мов для розмітки даних називають каскадні таблиці стилів сторінок (англійською Cascading Style Sheets, скорочено CSS) [30]. Найчастіше Cascading Style Sheets можуть використовувати з метою для візуалізації презентування сторінок, які було написано мовою розмітки гіпертекстів (HTML) та розширюваної мови розмітки гіпертексту (XHTML), проте для других видів документів виду XML, CSS також можливо застосовувати.

Cascading Style Sheets специфікації було створено та продовжують свій розвиток при активній участі Консорціуму мережі-інтернет.

Варта зазначити, що каскадним таблицям стилів сторінок притаманні відмінні один від одного рівні і профіля. Кожен наступний Cascading Style Sheets рівень може створюватись на базі попередніх, при цьому додається нова функціональність Також створення наступного рівня може відбуватись через розширення уже існуючих функцій. В свою чергу рівні прийнято позначати наступним чином, зокрема: CSS1,CSS2,CSS3. Профілі є сукупностями правил для Cascading Style Sheets для одного чи кількох рівнів які створено як для пристроїв певного окремого типу так і для інтерфейсів. Для прикладу, варта навести, що існують профіля Cascading Style Sheets і для принтерів, і для мобільних пристроїв і для інших засобів.

Підсумовуючи вище сказане, варта сказати, що Cascading Style Sheets (CSS) (верстка каскадного чи блочного типів) з'явилась як альтернатива чи навіть заміна для верстки web-сторінок табличного типу. Головною є перевагою верстки блокового типу є розприділення змісту даних (сторінок), так само як і їх візуального представлення через презентацію.

1.4.4 Аналіз скрипкової мови програмування PHP

Скриптова мова програмування PHP (англійською Hypertext Preprocessor) мова, яка досить широко використовується та є мовою сценаріїв з загальним призначенням, вихідний код якоє є відкритим.

Історія скрипкової мови почалась в далекому 1994 році десь восени. Саме тоді данський програміст Расмус Лердорф (Rasmus Lerdorf) розпочав свою роботу в напрямку, який через деякий час і став PHP [31]. При цьому єдиною його на той час ціллю було визначити хто ж читатиме його власне резюме. Вже тоді винахідливий програміст як незалежний підрядчик для пошуку роботи робив розсилку своїм потенційним роботодавцям свої короткі резюме з використанням URL-посилання на його повноцінну розширену версію. З метою стеження за відвідувачами, в нього виникла ідея створення CGI-скрипта з використанням Perl, який він вставляв в якості спеціального тегу в HTML код відповідної сторінки, і таким чином накопичував інформацію про його відвідувачів. Просто для того, щоби справити позитивне враження на своїх потенційних роботодавців, Расмус Лердорф дав дозвіл кожному відвідувачу його сторінки переглядати накопичену статистичну інформацію відвідувань.

Власне сам код призначений для збирання статистичних даних було прийняте рішення назвати наступним чинам: «PHP-Tools for Personal Home Page». В решті решт знайшлися люди яких зацікавив такий інструмент і як вони могли б його отримати для свого використання. Це спонукало Расмуса Лердорфа

запропонувати «винахід» третім особам. Проте, того часу ще не існувало руху, який сьогодні називається Open Source. І вже наприкінці 1995 року Лердорфом було відкрито для широкого загалу перший список розсилки по PHP. Ціль яку він вбачав полягала в наданні можливості обміну ідеями, кодом та виправлення помилок.

Скриптова мова програмування PHP є пре-процесором для мови розмітки гіпертекстів HTML. Приклад його роботи представлено на схемі яку зображено на рисунку 1.8.

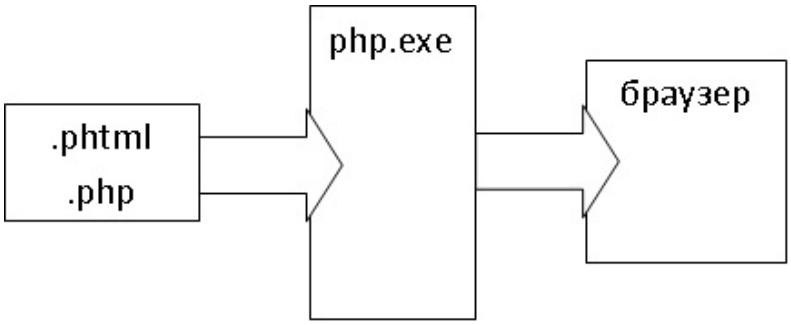

Рисунок 1.8 – Схема роботи скриптової мови програмування PHP

Про PHP можна говорити що вона є мовою серверних скриптів (англійською server scripting language, скорочено SSL), яку вбудовано в HTML, і власне який інтерпретовано та який повинен виконуватись на стороні сервера.

До того моменту, коли файл передається з сервера на браузер, він переглядається препроцесором-інтерпретатором. Щоб це могло відбуватись файли, що обробляються препроцесором, потрібно щоб мали деяке розширення і в більшості випадків це розширення типу .phtml чи .php3(однак ці такі значення можливо замінити) і хоч такої обов’язкової вимоги не існує, код для препроцесора. Перед тим як здійснювати відправку PHP-сторінки, код повинен «програтися» на стороні серверу після чого на брузер буде передано результат в вигляді знову тієї ж таки HTML-сторінки. Однак ця сторінка може суттєво відрізнятись від тієї, яка все ще зберігається на серверній стороні. При цьому, без додаткового жодного опрацювання з веб-сервера на браузер пересилатимутьться усі сторінки звичайного типу, які мають .html або.htm розширення.

Основною відмінністю від скриптів типу CGI, які написано іншими мовами, наприклад Perl чи C є написання програмах типу CGI коду HTML, який виводять, а при використанні PHP еобхідно вбудовувати свою програму-скрипт вже у готову сторінку HTML, але при цьому потрібно використати (<?php і ?>) – як відкриваючий тег так і закриваючий тег.

На відміну від мов клієнтських скриптів, зокремаJavascript/jscript/vbscript, мову PHP називають мовою серверних скриптів. Тут йдеться про те, що скрипт PHP реалізується на стороні серверу, при цьому клієнт отримує лише результат роботи, в той час коли для JavaScript притаманна передача коду в повній мірі на сторону клієнта, йоно машину, і вже лише тоді береться до уваги, виконується браузером.

Зрозуміло і дуже зручно коли роботу PHP представити на конкретному прикладі, як наприклад лістинг 1.1

Лістинг 1.1 – Web-сторінка (з елементами PHP-коду)

```
<html>
<head>
<title>Приклад</title>
</head>
<body>
<?php echo "Привіт, я PHP-програма!"; ?>
</body>
</html>
```

Вже після завершення виконання напропонованого в 1.1 лістингу з’явиться вітальна сторінка.

В загальному хочу перелічити притаманні PHP особливості, до яких варта віднести:

- низьку та плавна навчальну криву;
- розвинену функціональність по роботі з базами даних, рядками чи ж мережевим (-ми) з’єднанням (-ми);
- підтримку операцій пов’язаних з роботою файлових системам, мовою

Java,COM,XML,CORBA,WDDX,Macromedia Flash;

- хорошу кросплатформеність,сумісність з платформами, зкоерма будь-яким видом UNIX,Win32(Nt/95/98/2000), QNX,MACOS(Webten), OSX, Os/2, BEOS;
- хороша сумісність з серверами,зокрема модульомApache(UNIX, Win32),Cgi/fast-cgi,thttpd,fhttpd,phttpd,ISAPI(IIS, Zeus), NSAPI(Netscape iplanet), механізмом сервлетів, таких як Java, Aolserver та модульом Roxen/caudium;
- важливим є те, що для PHP властиво укорочений, у порівнянні з іншими, цикл розробки. Кожних кілька місяців продукуються нова версія, при чому вже з корегуванням знайдених помилок, наявністю додаткових функцій та іншими вдосконаленнями та покращеннями;
- об'єднання розробників є енергійним та достатньо доброзичливим. Достатня кількість безкоштовного коду та програмних прикладів. Забезпечення необхідним ресурсом та підтримкою навіть для новачків;
- можливість легкого створення власних розширень мови;
- синтаксис який нагадує C та є достатньо простим. Програмісти, що мають досвід роботи на C, C++ та Perl та з командними сценаріями труднощів при освоюванні PHP не виникає.

На виборі цієї мови програмування я, кірм вище зазначених її переваг, зупинився ще й тому, що для мене одним з основних є її безкоштовність.

Таким чином, підсумовуючи все вище сказане стосовно обґрунтування вибору мови програмування. Розуміючи, що веб-сторінка це не просто нагромадження якогось тексту і/чи картинок. Надзвичайно важливим є достатній, оптимальний рівень інтерактивності. Ще не так давно, все це намагались реалізувати з використанням CGI-скриптів, які писали з використанням Perl. Але виникли проблеми, пов'язані з поганою масштабованістю. Зокрема, багато часу займає специфіка, коли для кожного нового виклику CGI-скрипту потрібно породжувати новий процес від ядра, на що вистарається багато процесорного часу та затрат оперативної пам'яті. Цей процес можна оптимізувати, скориставшись можливостями PHP, робота як частина веб-серверу.

Крім того, за рахунок подібності синтаксис PHP до C чи Perl, де відсутня строга типізація даних, а також відсутня потреба функцій по виділення та звільнення пам'яті.

Характерною також для програм написаних з використанням PHP, є достатньо легке читання коду та їх розміння в цілому знов ж таки в порівнянні з програмами на Perl.

І нарешті плюсом до безоплатності бонусом маємо кросплатформеність зв'язки PHP-MySQL . Однак справедливим буде зазаначити, що My SQL все ж таки потребує купівлю ліцензії в разі її використанні з комерційною метою. Доступною мовою ,це може значити, що присутня можливість при роботі з Windows, для розробки додатків, призначенням яких є робота з Unix. Також в PHP є передбачено можливість роботи в якості зовнішнього CGI-процесу, чи ж в якості інтерпретатору скриптів, або як модуль, який можна підключати до веб-серверу Apache чи IIS.

І на завершення, так як представлений програмний продукт розроблено спільно, в загальному доступі є достатня кількість документації так само як і списки розсилки, в при необхідності є можливість отримати відповідь на будь-яке запитання причому, при бажання, це можна зробити безкоштовно.

1.4.5 Система управління вмістом PrestaShop

Prestashop [32] являється ще однією безкоштовною платформою призначеною для створення інтернет-магазинів з відкритим вихідним кодом за ліцензією OSL. Дана система є абсолютно безкоштовною і при цьому дужа проста і зручна. Для написання шаблонів вона використовує Smarty, а створити базу даних можна за допомогою бібліотеки MySQL.

Компанія Prestashop була створена в 2007 році і все продовжує розвиватися. Головні офіси знаходяться в Маямі і в Парижі. Дуже велику допомогу в розвитку

цієї платформи грає спів-товариство Prestashop, в яке входить вже більше 500 тис. чоловік. Цю систему використовує більше 140 тисяч інтернет магазинів. Кількість завантажень CMS даної платформи перевищує 1500 на добу. Компанія Prestashop надає як платну так і безкоштовну підтримку з офіційного сайту. Також вона виграє порівняно з іншими в легкості роботи і своїй швидкості.

До функціональності платформи можна віднести такі основні характеристики: оплата, статистика, доставка, каталог, безпека, локалізація, SEO, управління, переклади та модулі.

Оплату товарів в інтернет магазині створеному на Prestashop можна проводити за допомогою банківського переказу, оплати чеком, оплати готівкою при доставці. Так само, ви можете використовувати такі системи оплати через інтернет як WebMoney, PayPal, Moneybookers, Яндекс.Гроші та інші.

Статистика в платформі для створення інтернет бізнесу Prestashop є дуже великою і багатогранною. Можна можете вести статистику відвідувачів, облік товарів, статистику продажів і замовлень, статистику заходів як з пошукових систем так і з інших сайтів, статистику заходів за ключовими словами. Можна створити статистику кращих товарів, категорій і постачальників, статистику доставки, розсилки та інші. Візуалізацію в цій системі можна зробити за допомогою Silverlight, Flash, GD і Google - графіків.

Що стосується доставки, то тут ви можете посилати повідомлення про доставку товару на Email користувача, що надасть можливість відстеження доставки вашого товару в реальному часі.

В каталозі товарів є можливість встановлювати такі стандартні речі як фотографія, ціна з урахуванням податків або без них, використовувати водяні знаки, є можливість для клієнтів залишати коментарі та оцінки вподобаного ним товару. Покупець може сортувати продукти за різними критеріями, дивитись виробники і бренди, які присутні у продажу на вашому сайті, бачити на які товари відбуваються знижки. У вашому каталозі так само буде видно скільки товару залишилося в наявності на складі.

Безпека надається завдяки підтримці SSL-протоколу. У базі даних магазину є шифрування паролів. Так само є Cookies-шифрування.

Prestashop дозволяє перераховувати всі ціни в будь-яку валюту і податки за ставками регіону в якому знаходиться покупець. Так само, можливо синхронізувати валюту за чинним курсом. Таким чином проводиться локалізація інтернет ресурсу.

Платформа Prestashop дуже багатофункціональна саме в управлінні інтернет магазином. Існує можливість додавати будь-які потрібні вам модулі, встановити текстовий редактор WYSIWYG, створювати резервні копії баз даних, проводити автоматичну генерацію htaccess-файлів і robots.txt, управляти шрифтами, записами та інше.

Дана платформа дуже зручна у використанні для великого бізнесу або великого інтернет магазина. Тут є всі необхідні модулі та функції. Створювати з даною системою дуже швидко, просто і зручно.

2 РЕАЛІЗАЦІЯ Й ТЕСТУВАННЯ

2.1 Робочий потік реалізації

Реалізація головним чином стосується створення коду. Однак, об’єктно-орієнтований аналітик чи проектувальник може також бути задіяним для створення моделі реалізації.

Робочий потік реалізації – основний потік фази Конструювання. Головне завдання реалізації – перетворення проектної моделі в виконуваний код.

Моделювання реалізації має важливе значення, якщо планується використовувати модель при прямій розробці (генерації) коду. Модель реалізації – частина проектної моделі.

Артефакти представляють опис реальних сутностей, таких як вихідні файли. Вузли представляють опис обладнання і середовища виконання.

Діаграми розгортання дозволяють моделювати розподілення програмної системи на фізичному обладнанні.

Діяльність UP Реалізація архітектури полягає в визначенні важливих з архітектурної точки зору компонентів і проектування їх на фізичне обладнання.

Діаграма розгортання проектує програмну архітектуру і апаратну архітектуру. В процесі проектування можна створити «перше наближення» діаграми розгортання, визначаючи вузли і екземпляри вузлів та відносин. В процесі реалізації це наближення доповнюється компонентами чи екземплярами компонентів.

Дескрипторна форма діаграми розгортання може використовуватись для моделювання того, які типи обладнання, програмного забезпечення і зв’язків будуть присутні в повністю розгорнутій системі.

Дескрипторна форма описує повний набір можливих сценаріїв

розгортання, та показує:

- Відношення – типи зв’язків між вузами;
- Компоненти – типи компонентів, розгорнутих на конкретних вузлах.

Екземплярна форма діаграми розгортання представляє конкретне розгортання системи на визначених, ідентифікованих частинах обладнання. Описує один невизначений варіант розгортання теми, можливо на конкретному користувацькому сайті. Показує:

- Екземпляри вузлів – конкретні частини обладнання;
- Екземпляри відношень – конкретні відносини між екземплярами вузлів;
- Екземпляри компонентів – конкретні ідентифіковані частини програмного забезпечення, які розгорнуті на екземплярі вузла, наприклад конкретна копія Microsoft Office з унікальним серійним номером.

Артефакт – представляє опис реальної сутності, такої як конкретний виконуваний файл. Екземпляр артефакту – представляє визначений екземпляр конкретного артефакту, наприклад визначену копію конкретного виконуваного файлу, встановлену на конкретному комп’ютері.

2.1.2 Структурна схема розроблюваного сайту

Головне меню містить такі пункти:

- Ноутбуки;
- ПК;
- Планшети;
- Програмне забезпечення;
- Смартфони;
- Носії;

- Аксесуари

Структурна схема сайту зображена на рисунку 2.1

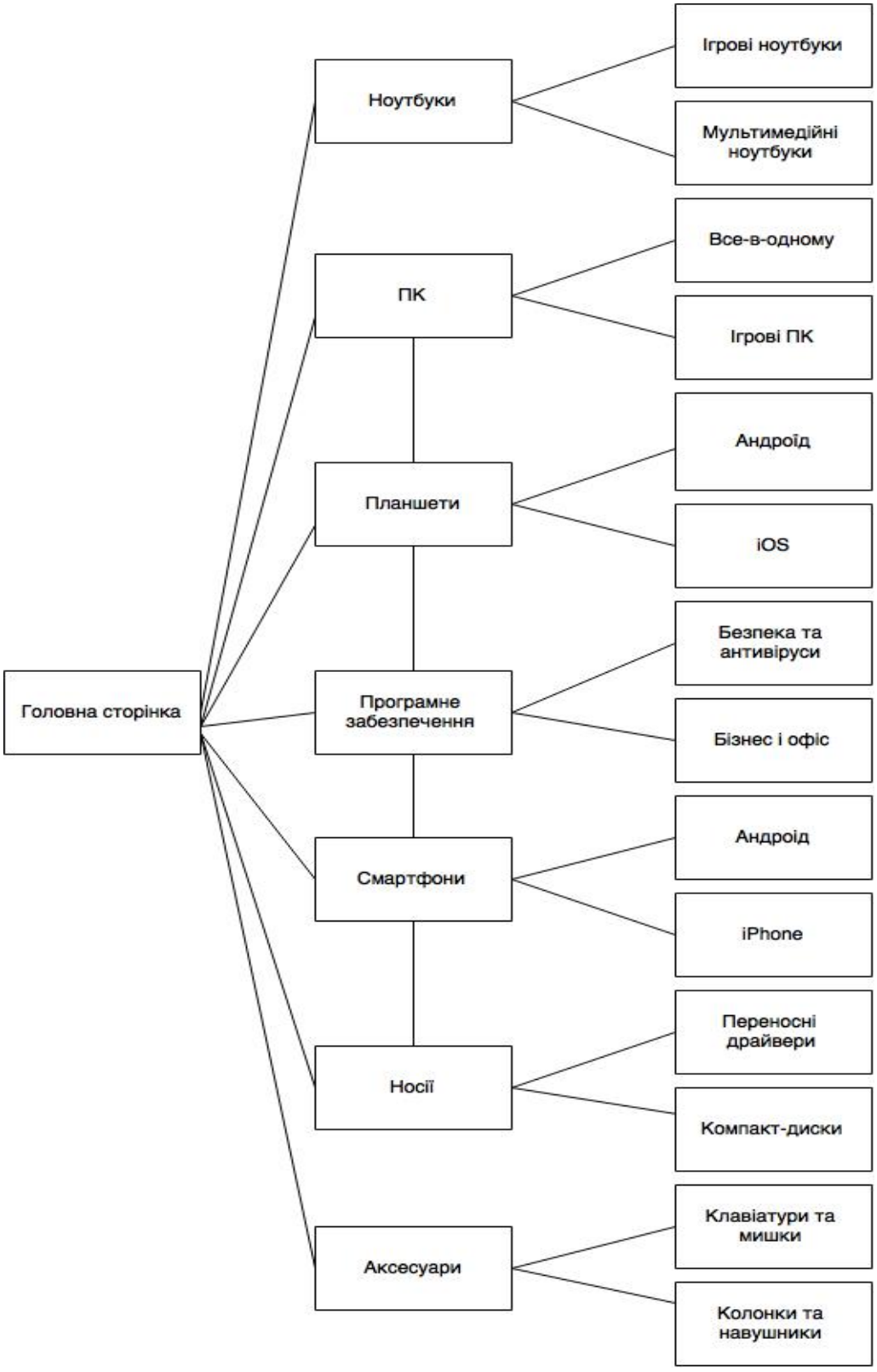

Рисунок 2.1 – Структурна схема сайту

Запропоновано варіант структури меню представлено на зображенні на рисунку 2.2

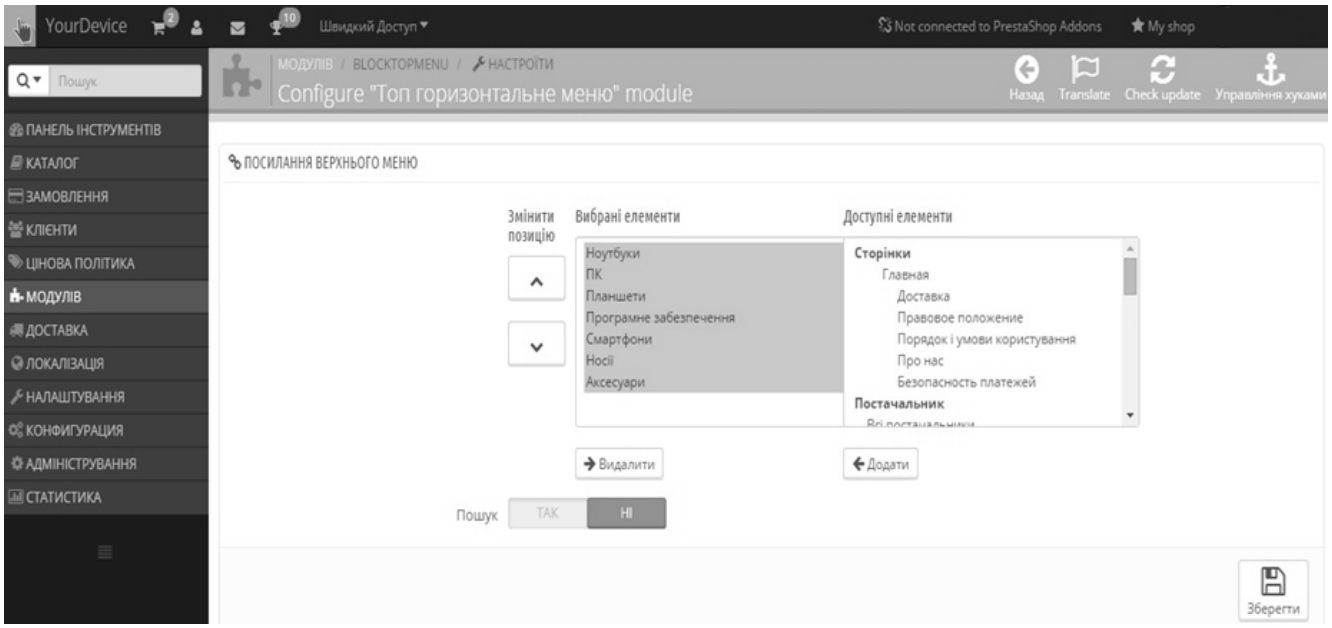

Рисунок 2.2 – Варіант структури меню

Створення та редагування слайд-шоу головної сторінки представлено на рисунку 2.3

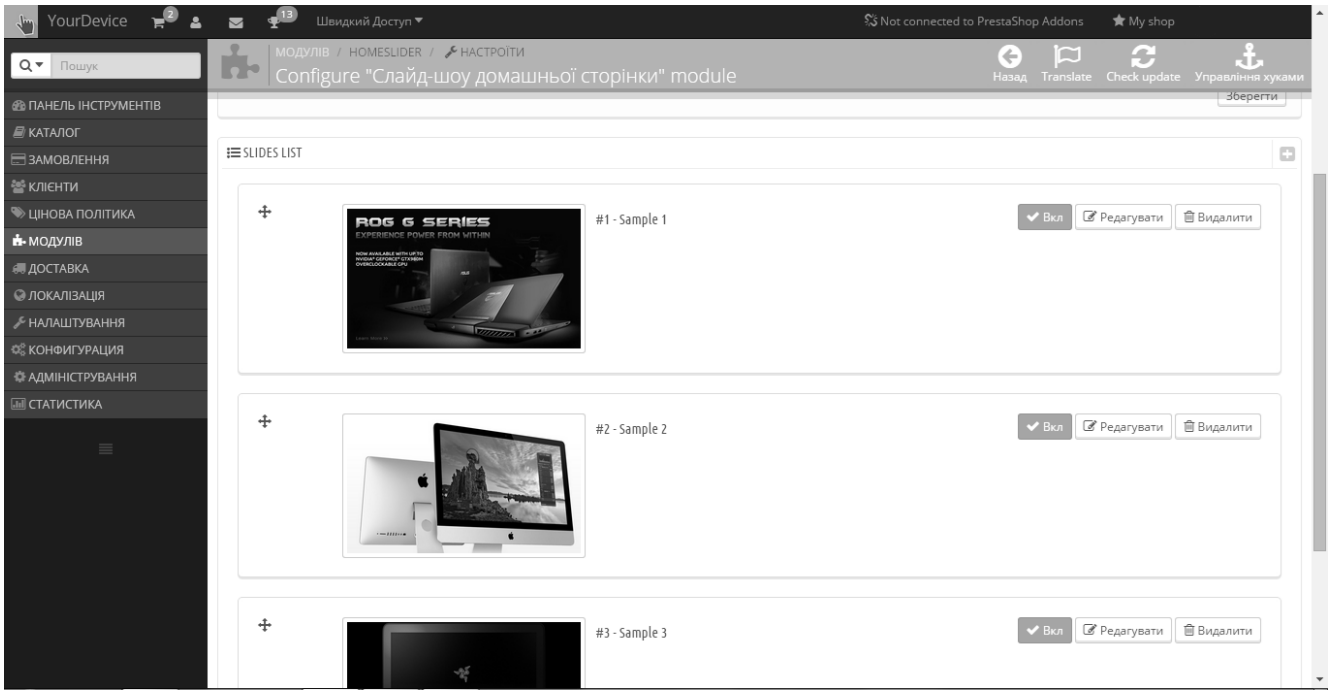

Рисунок 2.3 – Створення слайд-шоу головної сторінки

Додавання та редагування мов представлено на рисунку 2.4

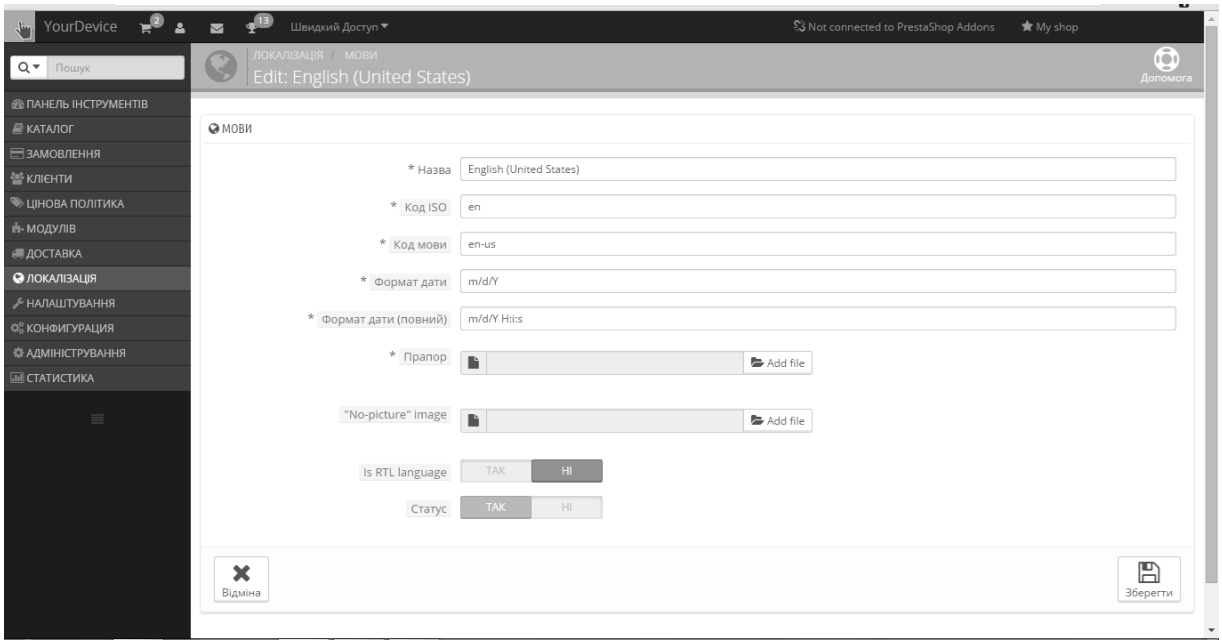

Рисунок 2.5 – Редагування мови відображення сайту

Додавання та редагування нової валюти розрахунку показано на рисунку 2.6

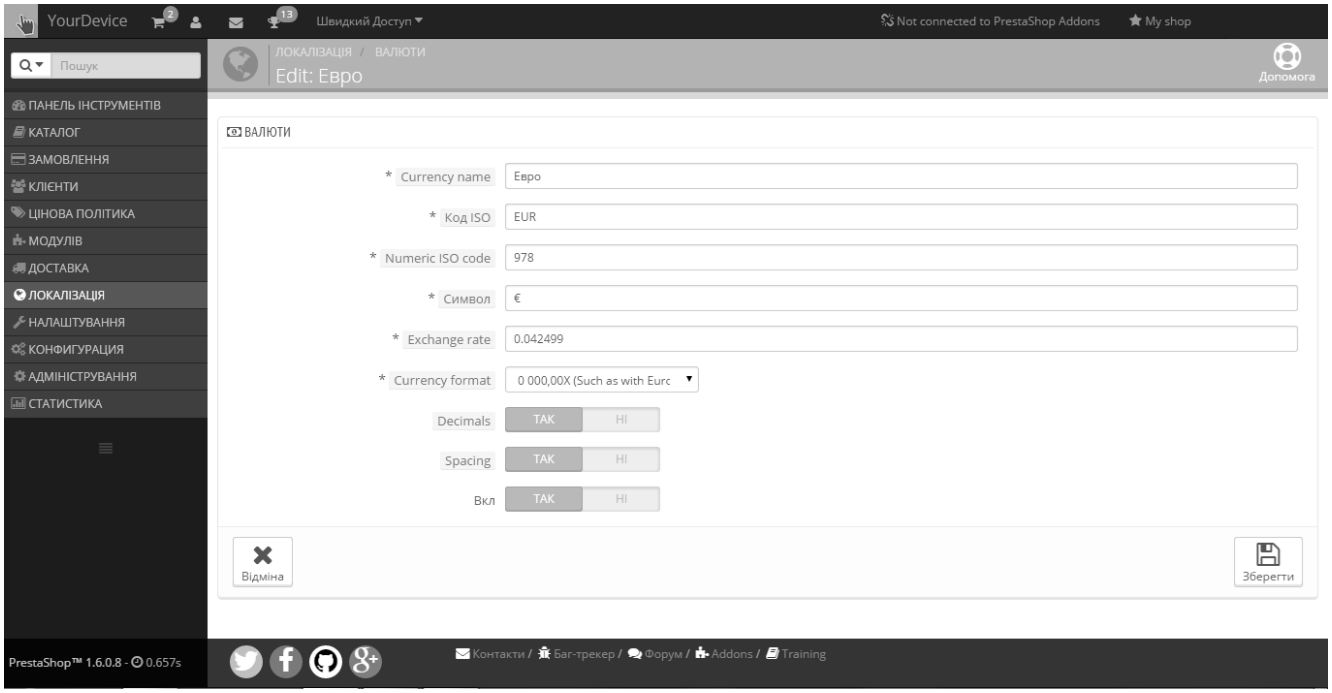

Рисунок 2.6 – Додавання та редагування валюти розрахунку

На кожній сторінці користувач має можливість переглядати та користуватися встановленими плагінами, а також матеріалами котрі викладені на

окремій сторінці. При переході між сторінками список доступних матеріалів залишається незмінним. Це є дуже зручно для користувача якому не потрібно витрачати час для окремого пошуку на кожній сторінці сайту.

Логотип сайту одночасно представляє посилання на головну сторінку, отже із головної сторінки сайту можна перейти на інші сторінки і навпаки. На кожній сторінці є доступ до меню, сторінок і матеріалів. На рисунку 2.7 зображена головна сторінка сайту.

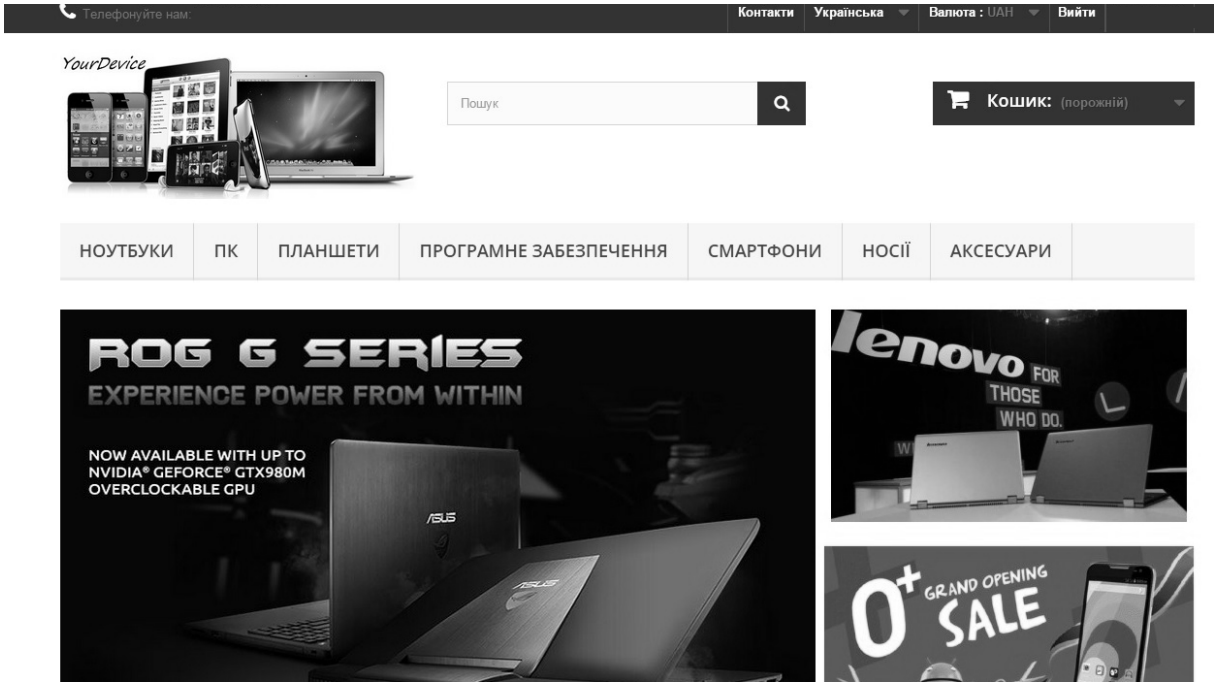

Рисунок 2.7 – Головна сторінка сайту

Додаткові категорії «Нові надходження», «Популярне», та «Лідери продажів» винесені на головну сторінку і показують список товарів відповідно до назви розділу. На рисунку 2.8 приведено приклад відображення цих розділів на сторінці сайту.

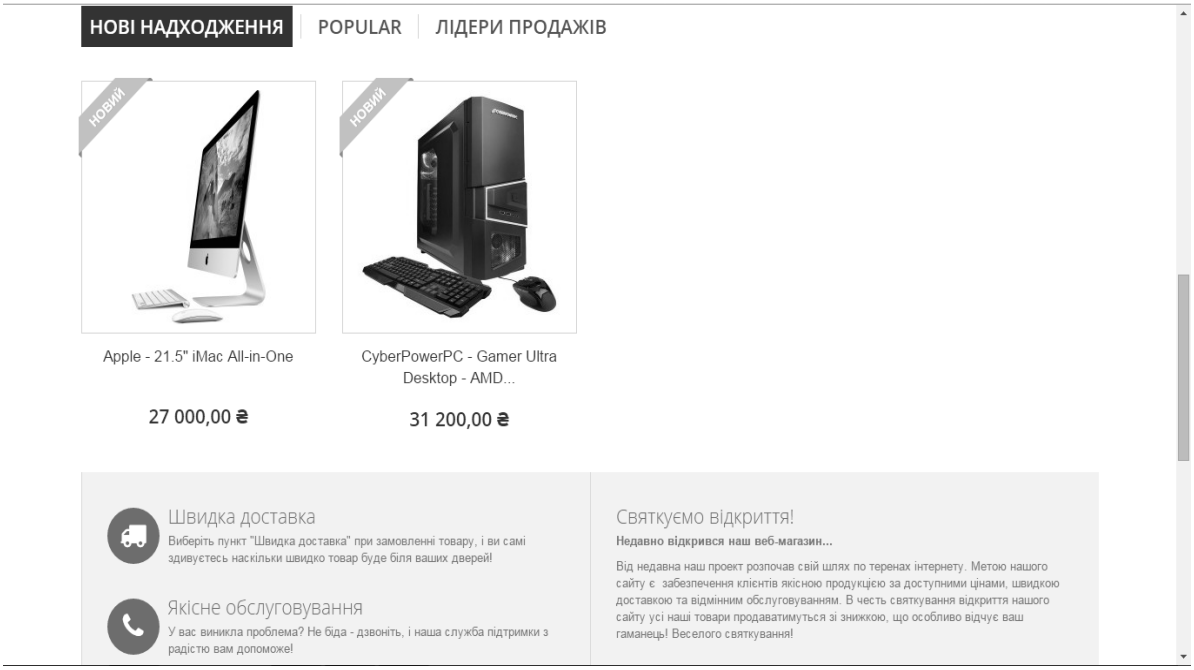

Рисунок 2.8 – Відображення додаткових категорій

Відображення та зовнішній вигляд категорії «Все-в-одному» зображений на рисунку 2.9

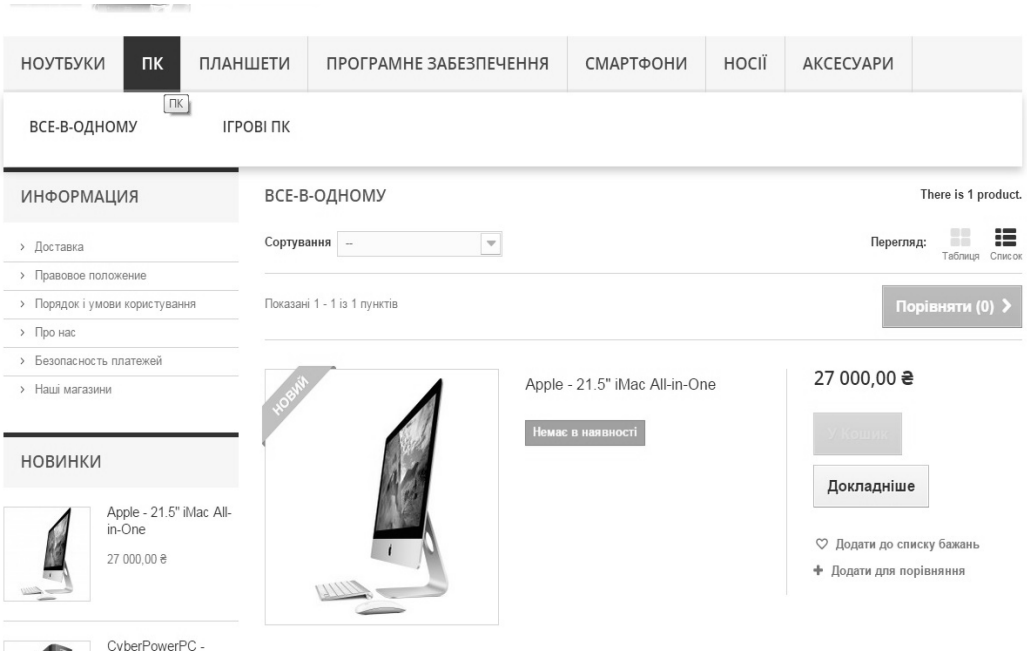

Рисунок 2.9 – Відображення категорії «Все-в-одному»

На рисунку 2.10 зображена головна сторінка товару, яка відображає ціну, характеристики, картинку.

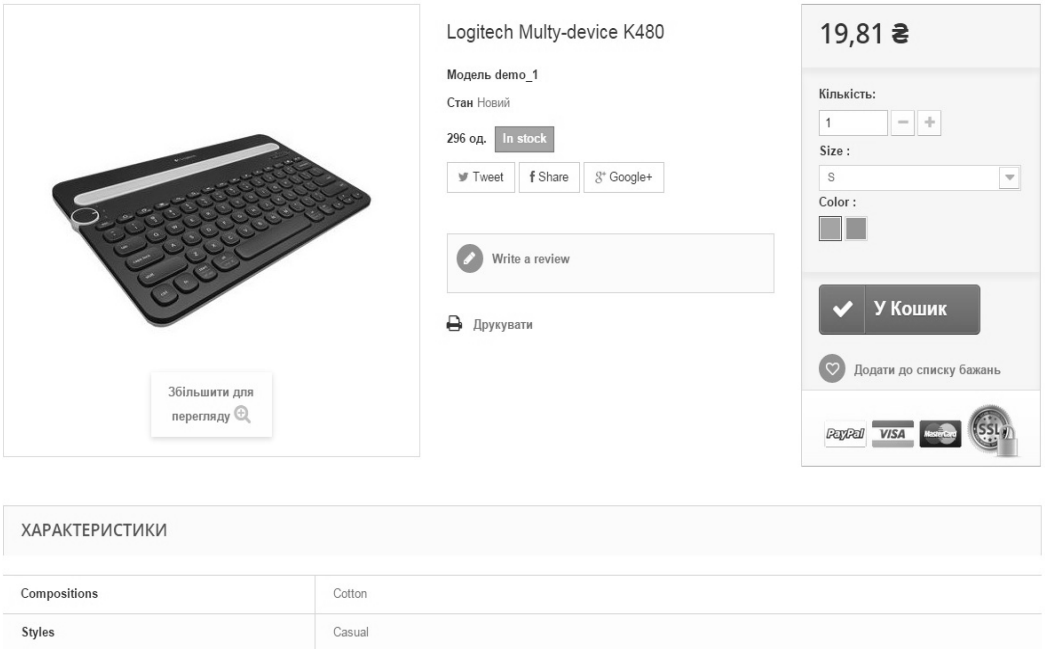

Рисунок 2.10 – Головна сторінка товару

Кошик для купівлі та його відображення приведені на рисунку 2.11

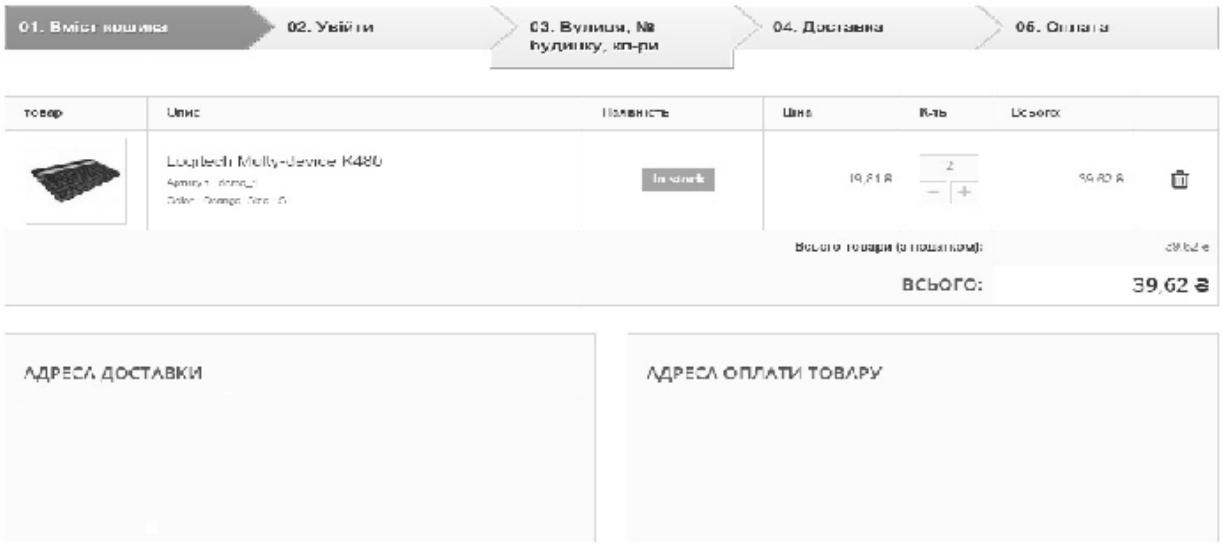

Рисунок 2.11 – Кошик для купівлі товару

На рисунку 2.12 показано вибір способу доставки.

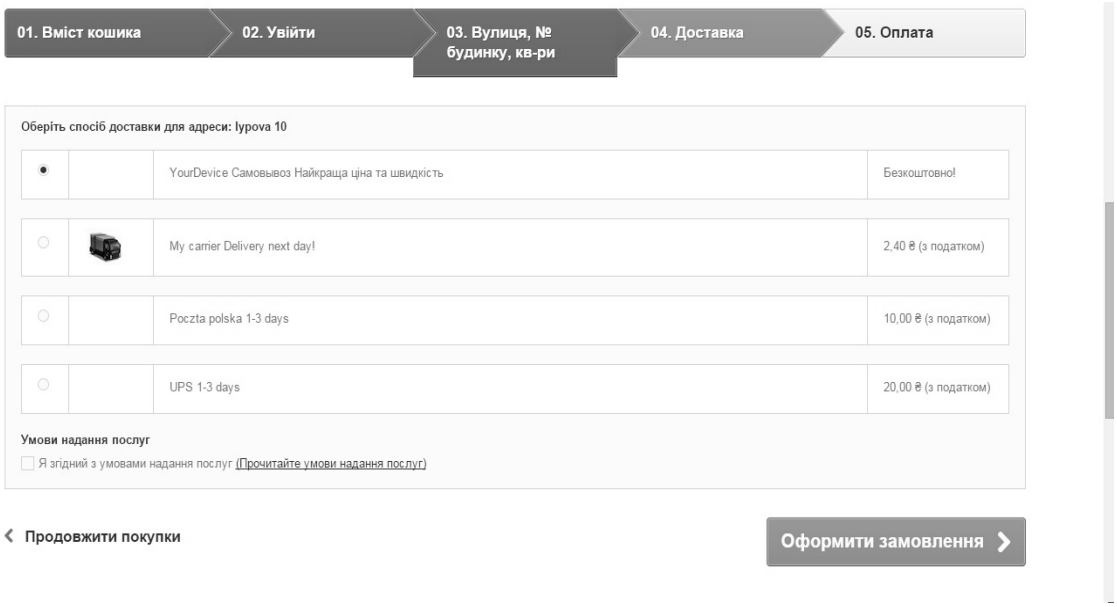

Рисунок 2.12 – Вибір способу доставки

На рисунку 2.13 представлене меню вибору способу оплати

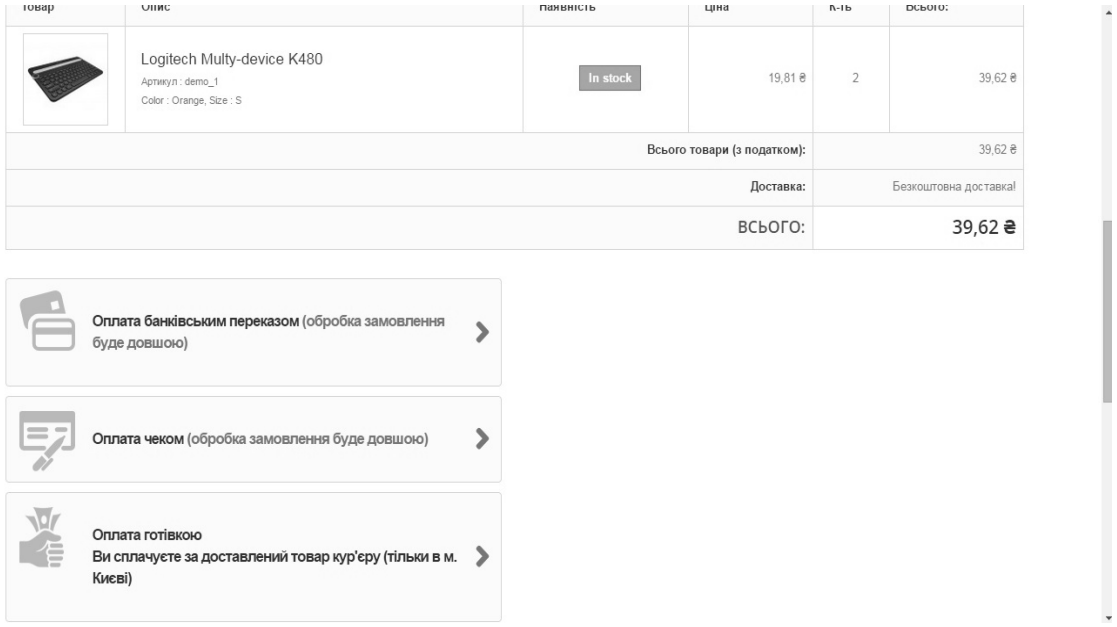

Рисунок 2.13 – Вибір способу оплати

ВИСНОВОК

Під час виконання представленої на розгляд Екзаменаційній комісії кваліфікаційної роботи на тему «Розробка нових модулів та оптимізація моделі з демонстрацією особливостей проектування за методом RUP, мова програмування PHP» я тримав корисний досвід роботи самостійно виконуючи кожен з етапів розробки якісного програмного забезпечення. Також я мав можливість відчути особливості та побувати на місці кожного з членів команди і користуватись та вивчати існуючу на сьогодні базу знань, незалежно від того, на якому етапі роботи я знаходився, чи розробляв вимоги, чи проєктував, чи виконував тестування чи ж управляв проєктом в цілому. А використовуючи раціональний уніфікований підхід зміг забезпечити спільну мову моделювання на всіх етапах роботи, мав можливість узгоджено бачити наперед як створювати програмне забезпечення для отримання оптимального кінцевого результату роботи. А як мову моделювання ,користуючись досвідом загальної бази знань я використав уніфіковану мову моделювання (англ. Unified Modeling Language, скороченоUML), що й передбачено міжнародним стандартом.

Таким чином раціональний уніфікований підхід, дав змогу передбачити дотримання строгого підходу до розподілу як завдань так і відповідальності в середині команди-розробника програмного забезпечення. Відповідно до його парадигм ґарантовано можливість реалізації всіх запланованих етапів створення програмного продукту точно по обумовленим термінам та не виходячи за розмір передбаченого бюджету передбаченого на створення якісного програмного продукту, який відповідатиме в повній мірі вимогам замовника.

Також варта наголосити, що раціональний уніфікований підхід – це вже момент оптимізації та вдосконалення, модулі, які розробляються з його врахуванням будуть працювати якісніше по всьому функціоналу як окремо так і програми в цілому. Особливості проєктування якісного програмного продукту за парадигмами раціонального уніфікованого підходу дасть змогу підвищити

продуктивність роботи всієї команди і надасть корисний додатковий досвід з можливістю його накопичення (оскільки при роботі над створенням кожного нового проєкту навики будуть лише вдосконалюватись) використовуючи різноманітні інформаційні джерела, шаблони та настанови по використанню інструментальних засобів на всіх критично важливих етапах роботи, впродовж всього життєвого циклу створення та супроводу програмного продукту.

Метою роботи було:

- дослідження особливостей мови програмування PHP;
- дослідження принципів проєктування методом з врахуванням парадигм раціонального уніфікованого підходу;
- вивчення системи керування вмістом PrestaShop і закріплення набутих в процесі навчання та практик знать та вмінь, а також особливостей мови програмування PHP, HTML.

Оскільки мети досягнуто, результати виконаної кваліфікаційної роботи готові до представлення на розгляд Екзаменаційної комісії №43, з демонстрацією основних моментів за допомогою презентаційних матеріалів.