МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ВЕБСАЙТУ ФІРМИ-ПРОВАЙДЕРА ІНТЕРНЕТ-ПОСЛУГ»**

Виконав: студент 4 курсу, групи 6.1219-2пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
М.С. Гривенко
(ініціали та прізвище)
Керівник доцент кафедри програмної інженерії, доцент, к.т.н. Лимаренко Ю.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри електроніки, інформаційних систем та програмного забезпечення ІННІ ЗНУ, доцент, к.т.н. Заяц В.І.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ 07 ” 02 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Гривенку Микиті Сергійовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка вебсайту фірми-провайдера Інтернет-послуг

керівник роботи Лимаренко Юлія Олексіївна, к.т.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 26 » січня 2023 року № 102-с

2. Строк подання студентом роботи 07.06.2023 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.
3. Вихідні дані до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Дослідження та аналіз ПЗ.
2. Проєктування вебсайту.
3. Розробка сайту.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою докладу

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 07.02.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи та постановка	08.02.2023	
	задачі.		
2.	Збір вихідних даних, обробка методичних	02.03.2023	
	та теоретичних джерел.		
3.	Розробка першого розділу.	23.03.2023	
4.	Розробка другого розділу.	20.04.2023	
5.	Розробка третього розділу.	15.05.2023	
6.	Оформлення та нормоконтроль	01.06.2023	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	22.06.2023	

Студент _____ М.С. Гривенко
(підпис) (ініціали та прізвище)

Керівник роботи _____ Ю.О. Лимаренко
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка вебсайту фірми-провайдера Інтернет-послуг»: 55 с., 44 рис., 15 джерел.

ІНТЕРНЕТ-ПРОВАЙДЕР, ТЕЛЕГРАМ БОТ, CSS, HTML, PYTHON.

Об’єкт дослідження – основні засоби розробки сучасних вебсайтів.

Мета роботи: розробка вебсайту інтернет-провайдера “GreeNet”.

Метод дослідження – аналітичний.

Для розробки сайту та написання дипломної роботи були вивчені та розглянуті засоби для створення Web-сайтів.

Під час виконання завдання були розглянуті основи створення сайтів, вивчена теорія, що пізніше була реалізована на практиці у вигляді створеного програмного продукту. Для створення сайту було використано HTML, CSS, JavaScript, для створення телеграм боту Python та бібліотека aiogram.

Результатом даної роботи є розробка веб-сайту інтернет-провайдера “GreeNet” і телеграм боту, для зворотнього зв’язку з користувачами.

SUMMARY

Bachelor’s qualifying paper «Development of the Internet Provider’s Website»: 55 pages, 44 figures, 15 references.

INTERNET-PROVIDER, TELEGRAM BOT, CSS, HTML, PYTHON.

The object of the study is the main means of developing modern websites.

The aim of the study is the website of the Internet provider «GreeNet».

The method of research is analytical.

For the development of the website and the writing of the thesis, the tools for creating websites were studied and considered.

During the execution of the task, the basics of website creation were considered, the theory was studied, which was later implemented in practice and in the form of a created software product. HTML, CSS, JavaScript were used to create the site, Python and the aiogram library were used to create bot telegrams.

The result of this work is the development of the website of the Internet provider «GreeNet» and a Telegram bot for user feedback.

ЗМІСТ

ВСТУП

Створення вебсайтів у нас час дуже популярне і з часом створювати сайти стає легше, та кожен може створити свій сайт за лічені години.

На данний момент існує багато онлайн конструкторів, таких як: Wix, Google AdSense, Flexbe, uKit, та багато іншого. Але ці сайти не будуть мати такого великого функціоналу, адже вони шаблонні і не мають такої різнобарвної палітри. Тому для розробки великих і серйозних сайтів використовуються стандартизована мова гіпертекстової розмітки HTML, формальна мова опису зонішнього виду документа CSS, мова програмування JavaScript, C-подібна скриптова мова загального призначення PHP, декларована мова програмування SQL, об’єктно-орієнтована мова програмування Python, та багато інших.

Мови програмування з кожним роком стають більш простішими, а у інтернеті з’являються все більше способів вивчити їх з легкістю. Тому майже кожна людина може створити свій сайт, за допомогою мови програмування.

Об’єкт дослідження – процес пошуку та перегляду сайті інтернет провайдерів.

Предмет дослідження – вебсайт інтернет-провайдера “GreeNet”.

Мета дослідження – розробка вебсайту інтернет-провайдера “GreeNet”.

Для реалізації поставленої мети було сформульовано наступні завдання:

– проаналізувати вебсайти інтернет провайдерів;
– дослідити та обрати засоби розробки;
– розробити проєкт сайту інтернет-провайдера “GreeNet”;
– розробити сайт відповідно до проєкту.

Розроблена система задовольняє всім вимогам, що були зазначені на етапі постановки завдання.

Кваліфікаційна робота складається зі вступу, трьох розділів.

Перший розділ містить основні відомості про вебсайт інтернет-

провайдера, а саме розглянуто сайти-аналоги та сформульовано основні вимоги до розроблюваного сайту, досліджено інструменти розробки.

Другий розділ містить технічне завдання, опис структури вебсайтів, діаграму варіантів використання, макети для створювання сайту.

Третій розділ містить опис розробленого вебсайту та телеграм боту, особливості реалізації, опис вебсторінок розробленої інформаційної системи та функціонал телеграм боту.

1 ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис об’єкту автоматизації

Основною метою проєктування є – створення сайту інтернет-провайдера “GreeNet”. Сайт повинен являти собою візитівку інтернет провайдера, за допомогою якого користувачі зможуть дізнатись більше про нього.

Інтерфейс повинен бути зрозумілим, простим та привабливим для зовнішнього вигляду, але щоб користувачі різних вмінь та навичок користування комп’ютером могли з легкістю орієнтуватись на ньому.

1.2Огляд аналогів

Для створення сайту-візитки інтернет провайдера, необхідно дослідити аналоги які є популярними у місті. Можна виділити два досить відомих інтернет провайдера у місті – Kyivstar та DiaNet.

Нас цікавить оформлення та наявний функціонал. Головна сторінка сайту Kyivstar представлено на рисунку 1.1.

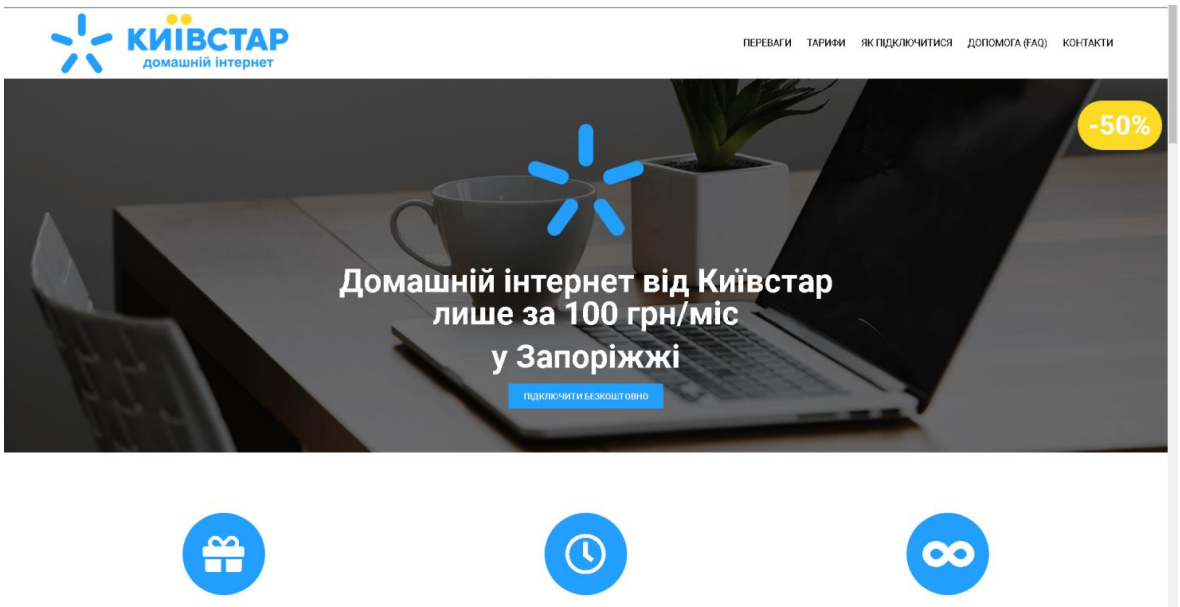

Рисунок 1.1 – Шапка сайту Kyivstar

Функціонал наявний на головній сторінці сайту “Kyivstar” наведено на рисунку 1.2.

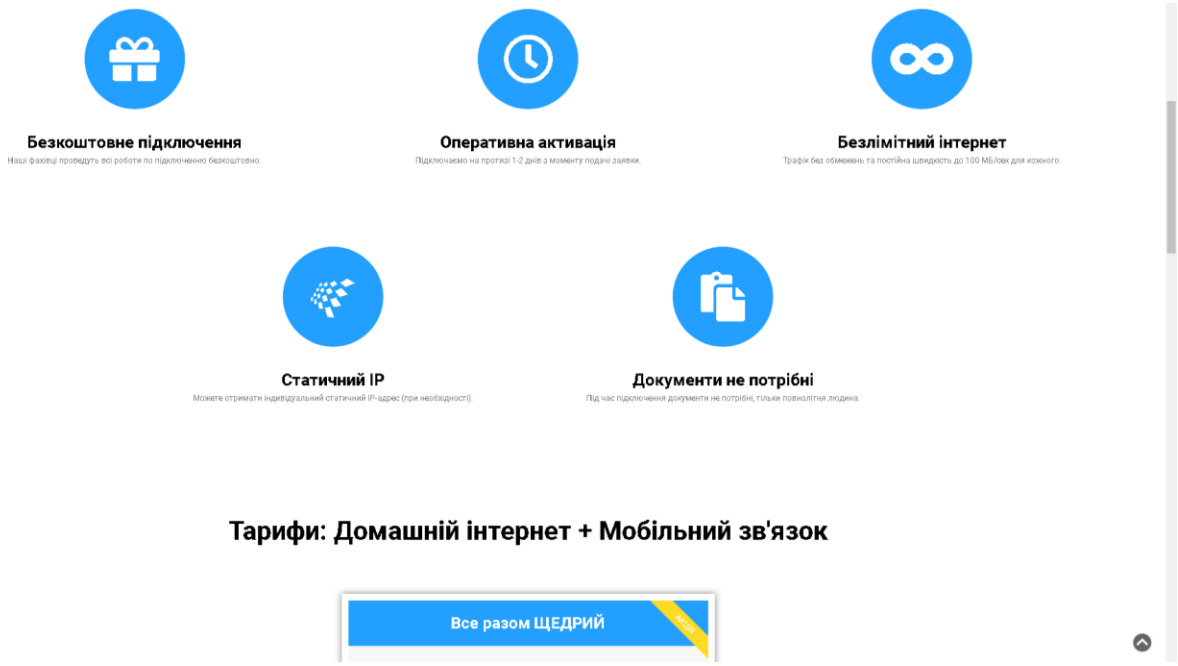

Рисунок 1.2 – Додаткові розділи сайту Kyivstar

Тарифікації та інструкції підключення представлені на рисунку 1.3.

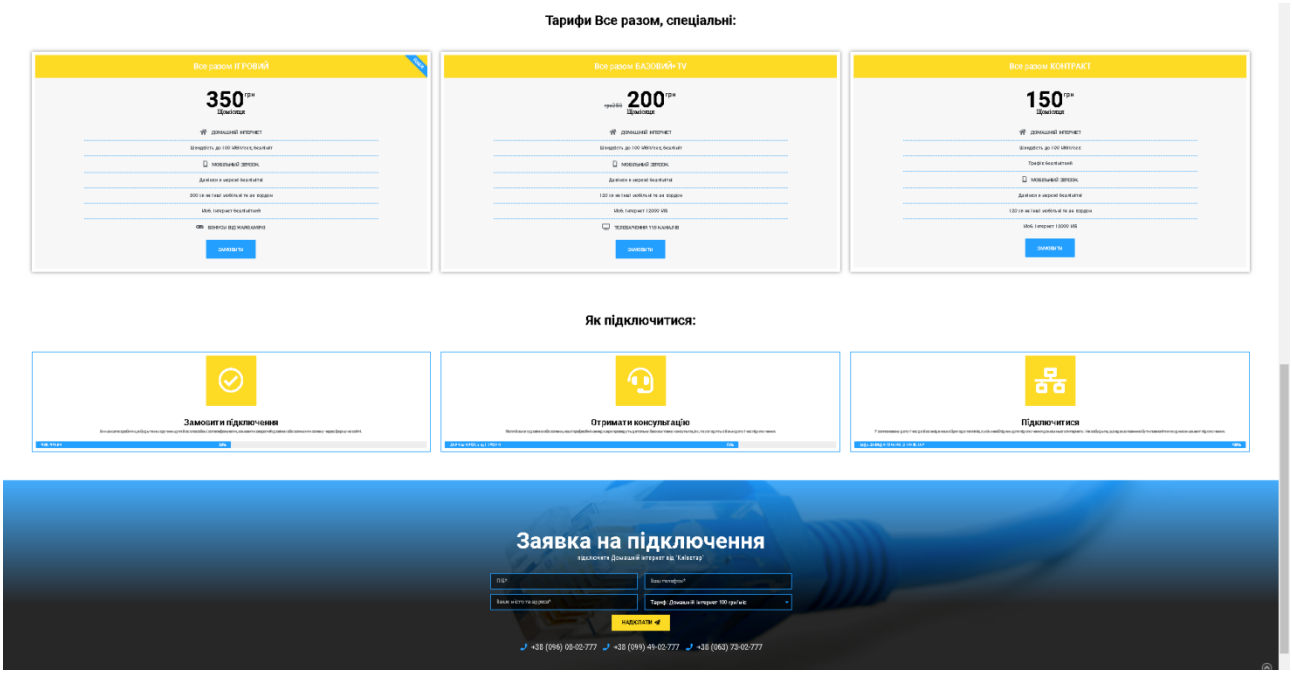

Рисунок 1.3 – Тарифікація інтернет провайдера Kyivstar

Тепер розглянемо головну сторінку інтернет провайдера “DiaNet”. Головна сторінка представлена на рисунку 1.4.

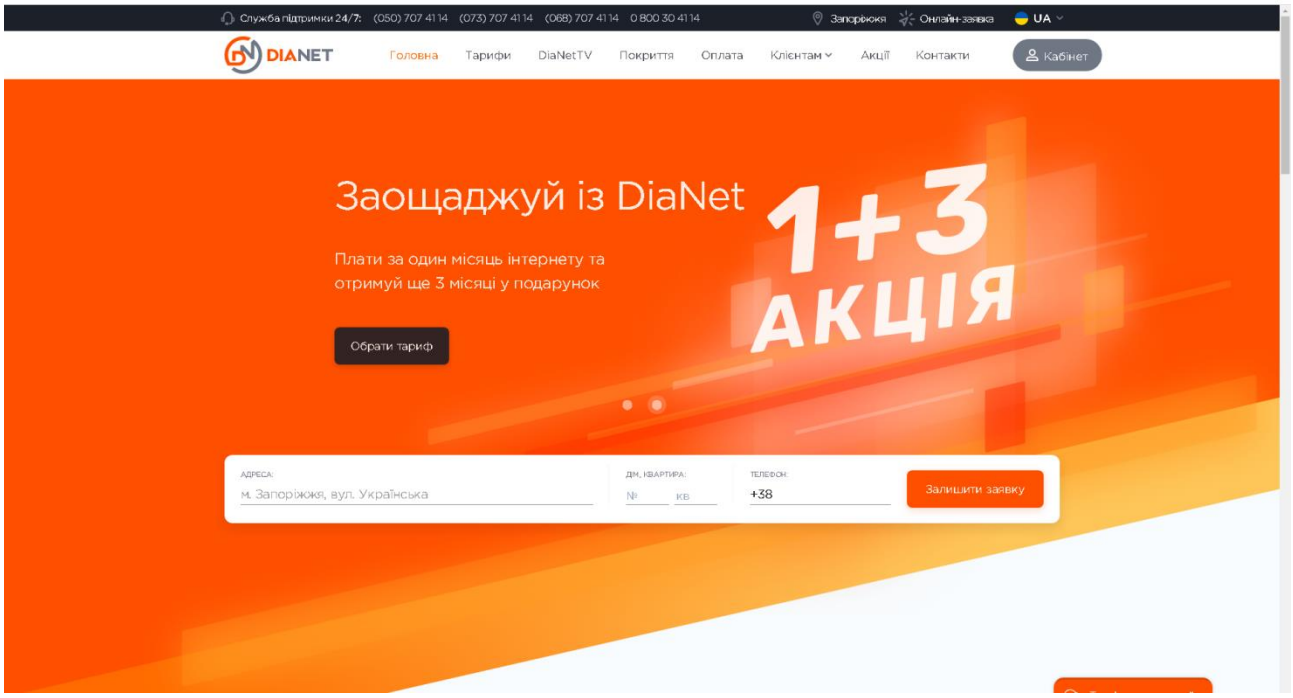

Рисунок 1.4 – Шапка сайту DiaNet

Інформація розміщена на головній сторінці представлена на рисунку 1.5.

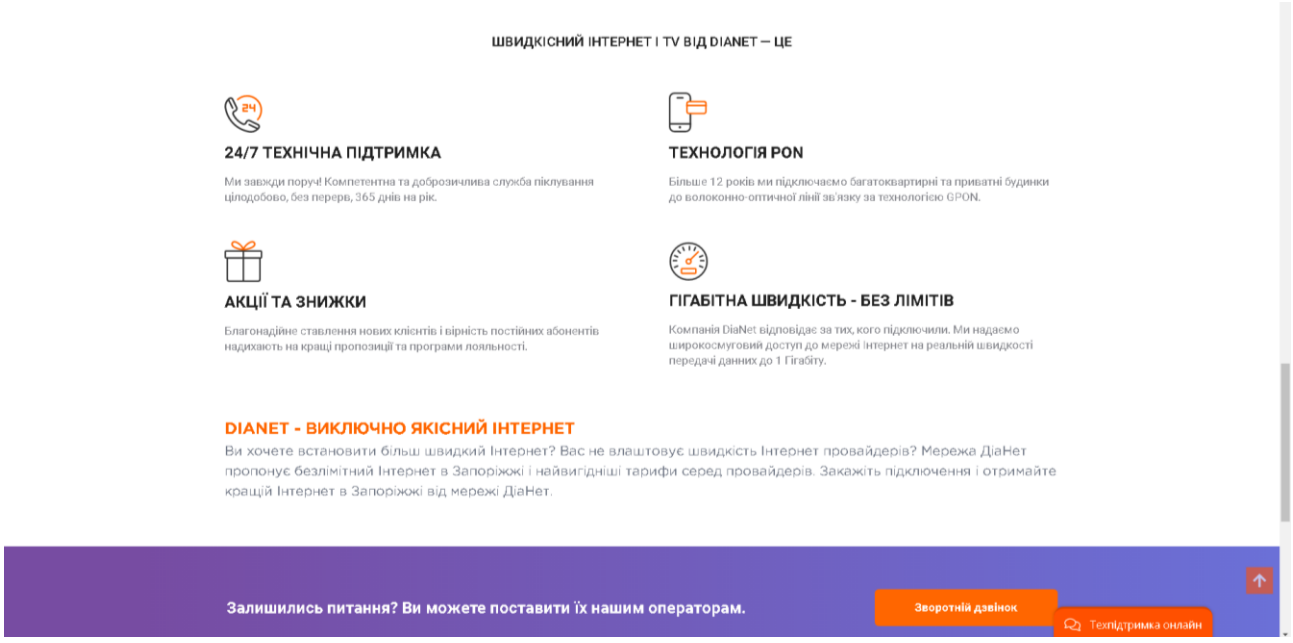

Рисунок 1.5 – Інформація сайту DiaNet

Розділ з тарифікацією розміщений на головній сторінці представлено на рисунку 1.6.

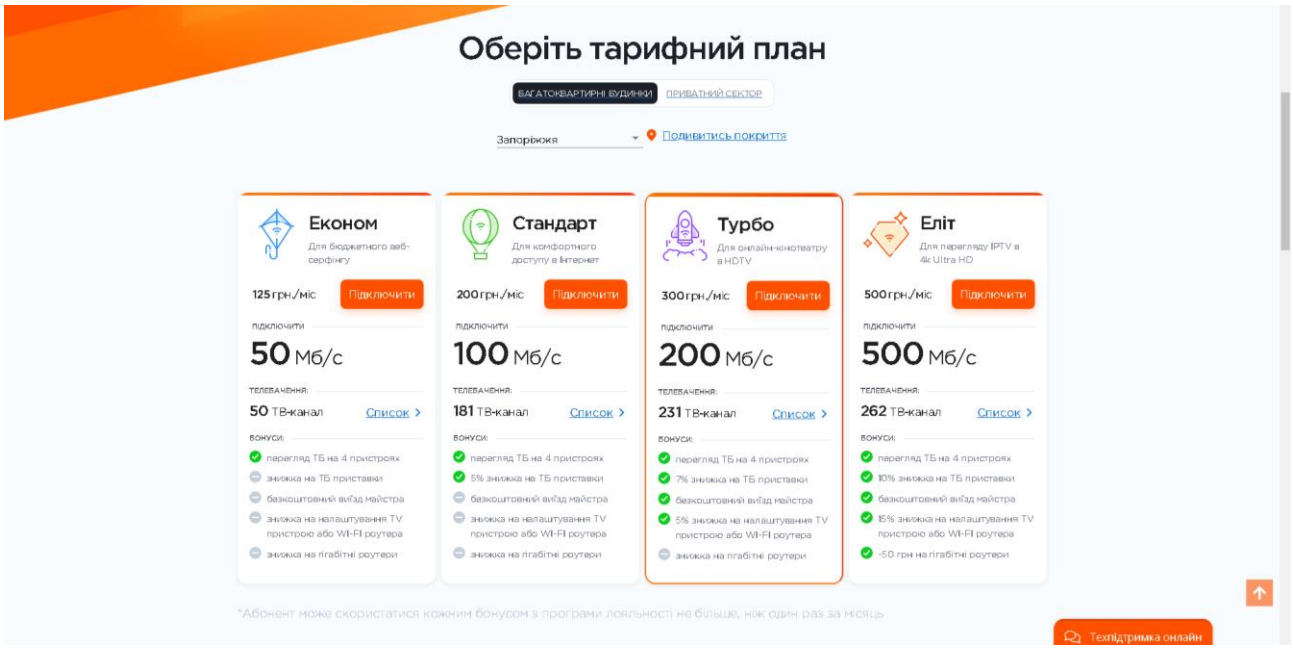

Рисунок 1.6 – Тарифікація провайдера DiaNet

1.2.1 Переваги сайтів

Розглянувши дизайн та функціонал сайтів, можна виділити наступні переваги.

Переваги сайту Kyivstar:

– дизайн розроблений в одному стилі;
– функціональність сайту;
– легкість в роботі з сайтом;
– зрозумілість знаходження об'єктів на сайті.

Переваги сайту Dianet:

– різнобарвний дизайн;
– наявність карти міста, з місцями покриття;
– детальна інформація про тарифікацію;
– зворотній зв'язок з операторами на сайті та у різних месенджерах

таких як Viber, Telegram.

1.2.2 Недоліки сайтів

Недоліки сайту Kyivstar:

– дизайн розроблений в одному стилі, не такий цікавий як різнобарвний у Dianet;
– вся інформація знаходиться на основній сторінці;
– зворотній зв'язок на сайті лише через бота.

Недоліки сайту Dianet:

– уся інформація розміщена на головній сторінці;
– інші сторінки просто розбивають інформацію що наявна на головній.

1.3 Постановка завдання

За результатами аналізу аналогічних вебсайтів було сформульовано наступні задачі:

– організація зрозумілого і простого в управлінні користувацького інтерфейсу;
– забезпечення інформативних сторінок;
– створення привабливого зовнішнього вигляду.

1.4 Формування вимог до сайту

З урахуванням аналогічних сайтів, можна сформулювати такі вимоги до сайту:

– меню, розміщене в шапці сайту;

– слайдер, для акцентування уваги користувачів на інформації;
– кнопка переходу до зовнішнього ресурсу особового кабінету інтернет провайдера;
– кнопка переходу до телеграм боту для зворотнього зв’язку з користувачами.

1.5 Засоби розробки сайтів

У наш час існує дуже багато мов для створення сайтів, опишемо найпопулярніші, такі як: HTML, CSS, JavaScript, PHP, SQL,Python.

1.5.1 Html

HTML (від англ. HyperText Markup Language – «мова гіпертекстової розмітки») – стандартизований мову розмітки вебсторінок у Всесвітній павутині. Код HTML інтерпретується браузерами; отримана в результаті інтерпретації сторінка відображається на екрані монітора комп'ютера або мобільного пристрою.

Мова HTML до 5-ї версії визначався як додаток SGML (стандартної узагальненої мови розмітки за стандартом ISO 8879). Специфікації HTML5 формулюються в термінах DOM (об'єктній моделі документа).

Суворим варіантом HTML є XHTML, він успадковує синтаксис XML і є додатком мови XML в області розмітки гіпертексту.

HTML-сторінки, як правило, відкриваються браузерами обмінюючись з сервером інформацією по протоколу HTTP або HTTPS, у вигляді простого тексту або з використанням шифрування.

Мова гіпертекстової розмітки HTML був розроблений британським вченим Тімом Бернерс-Лі приблизно в 1986-1991 роках в стінах ЦЕРНу в

Женеві в Швейцарії. HTML створювався як мова для обміну науковою і технічною документацією, придатний для використання людьми, які не є фахівцями в області верстки. HTML успішно справлявся з проблемою складності SGML шляхом визначення невеликого набору структурних і семантичних елементів - дескрипторів. Дескриптори також часто називають «тегами». За допомогою HTML можна легко створити відносно простий, але красиво оформлений документ. Крім спрощення структури документа, в HTML внесена підтримка гіпертексту. Мультимедійні можливості були додані пізніше.

Першим загальнодоступним описом HTML був документ «Теги HTML», вперше згаданий в Інтернеті Тімом Бернерс-Лі в кінці 1991 року. У ньому описуються 18 елементів, що становлять початковий, відносно простий дизайн HTML. За винятком тега гіперпосилання, на них сильно вплинув SGMLguid, внутрішній формат документації, заснований на стандартному узагальненому мовою розмітки (SGML), в CERN. Одинадцять з цих елементів все ще існують в HTML 4.

Спочатку мова HTML був задуманий і створений як засіб структурування та форматування документів без їх прив'язки до засобів відтворення (відображення). В ідеалі, текст з розміткою HTML повинен був без стилістичних та структурних спотворень відтворюватися на обладнанні з різною технічною оснащеністю (кольоровий екран сучасного комп'ютера, монохромний екран органайзера, обмежений за розмірами екран мобільного телефону або пристрою і програми голосового відтворення текстів). Однак сучасне застосування HTML дуже далеко від його початкової задачі. Наприклад, тег <table> призначений для створення в документах таблиць, але іноді використовується і для оформлення розміщення елементів на сторінці. З плином часу основна ідея платформонезавісимость мови HTML була принесена в жертву сучасним потребам в мультимедійному і графічному оформленні.

HTML – тегів мову розмітки документів. Будь-який документ на мові HTML являє собою набір елементів, причому початок і кінець кожного елемента позначається спеціальними позначками – тегами. Елементи можуть

бути порожніми, тобто не містять ніякого тексту та інших даних. В цьому випадку зазвичай не вказується закриває тег (наприклад, тег розриву рядків
 – одиночний і закривати його не потрібно). Крім того, елементи можуть мати атрибути, що визначають будь-які їх властивості (наприклад, атрибут href = "у посилання).

1.5.2 Css

CSS (/ siːɛsɛs / англ. Cascading Style Sheets «каскадні таблиці стилів») – формальна мова опису зовнішнього вигляду документа (вебсторінки), написаного з використанням мови розмітки (найчастіше HTML або XHTML). Також може застосовуватися до будь-яких XML-документах, наприклад, до SVG або XUL.

CSS використовується творцями вебсторінок для завдання кольорів, шрифтів, стилів, розташування окремих блоків і інших аспектів представлення зовнішнього вигляду цих вебсторінок. Основною метою розробки CSS було відділення опису логічної структури вебсторінки (яке проводиться за допомогою HTML або інших мов розмітки) від опису зовнішнього вигляду цієї вебсторінки (яке тепер проводиться за допомогою формального мови CSS). Такий поділ може збільшити доступність документа, надати велику гнучкість і можливість управління його поданням, а також зменшити складність і повторюваність в структурному вмісті.

Крім того, CSS дозволяє представляти один і той же документ в різних стилях або методах виведення, таких як екранне уявлення, друковане видання, читання голосом

До появи CSS оформлення вебсторінок здійснювалося виключно засобами HTML, безпосередньо всередині вмісту документа. Однак з появою CSS стало можливим принципове розділення змісту і представлення документа. За рахунок цього нововведення стало можливим легке застосування єдиного

стилю оформлення для маси схожих документів, а також швидка зміна цього оформлення.

1.5.3 JavaScript

JavaScript – мультіпарадігменна мова програмування. Підтримує об'єктно-орієнтована, імперативний і функціональний стилі. Є реалізацією стандарту ECMAScript. JavaScript зазвичай використовується як вбудований мова для програмного доступу до об'єктів додатків.

JavaScript зазвичай використовується як вбудований мова для програмного доступу до об'єктів додатків. Найбільш широке застосування знаходить в браузерах як мова сценаріїв для додавання інтерактивності вебсторінок. Основні архітектурні риси: динамічна типізація, слабка типізація, автоматичне керування пам'яттю, прототипне програмування, функції як об'єкти першого класу.

JavaScript є об'єктно-орієнтованою мовою, але що використовується в мові прототипирование обумовлює відмінності в роботі з об'єктами в порівнянні з традиційними клас-орієнтованими мовами. Крім того, JavaScript має ряд властивостей, властивих функціональним мовам, – функції як об'єкти першого класу, об'єкти як списки, каррінг, анонімні функції, замикання – що додає мові додаткову гнучкість.

Незважаючи на схожий з Сі синтаксис, JavaScript в порівнянні з мовою Сі має корінні відмінності:

– об'єкти з можливістю інтроспекції;
– функції як об'єкти першого класу;
– автоматичне приведення типів;
– автоматичне прибирання сміття;
– анонімні функції.

У мові відсутні такі корисні речі, як:

Стандартна бібліотека: зокрема, відсутній інтерфейс програмування додатків по роботі з файловою системою, управління потоками введення-виведення, базових типів для бінарних даних; стандартні інтерфейси до вебсерверів і баз даних; система управління пакетами, яка б відстежувала залежності і автоматично встановлювала їх.

1.5.4 Php

PHP – скриптова мова загального призначення, інтенсивно застосовується для розробки вебдодатків. В даний час підтримується переважною більшістю хостинг-провайдерів і є одним з лідерів серед мов, що застосовуються для створення динамічних вебсайтів.

В області вебпрограмування, зокрема серверної частини, PHP – один з популярних сценарних мов (разом з JSP, Perl і мовами, використовуваними в ASP.NET). Популярність в області побудови вебсайтів визначається наявністю великого набору вбудованих засобів і додаткових модулів для розробки вебдодатків. Основні з них:

- автоматичне вилучення POST- і GET-параметрів, а також змінних оточення вебсервера в зумовлені масиви;
- взаємодія з великою кількістю різних систем управління базами даних через додаткові модулі (MySQL, MySQLi, SQLite, PostgreSQL, Oracle Database (OCI8), Microsoft SQL Server, Sybase, ODBC, mSQL, IBM DB2, Cloudscape і Apache Derby, Informix, Ovrimos SQL, Lotus Notes, DB ++, DBM, dBase, DBX, FrontBase, FilePro, Ingres II, SESAM, Firebird і InterBase, Paradox File Access, MaxDB, інтерфейс PDO, Redis);
- автоматизована відправка HTTP-заголовків;
- робота з HTTP-авторизацією;
- робота з cookies і сесіями;

– робота з локальними і віддаленими файлами, сокетами;
– обробка файлів, що завантажуються на сервер;
– робота з XForms.

Станом на 2010-і роки використовується сотнями тисяч розробників; згідно з рейтингом корпорації TIOBE, що базується на даних пошукових систем, в травні 2020 року PHP знаходився на 6 місці серед мов програмування. Входить в LAMP – поширений набір програмного забезпечення для створення та хостингу вебсайтів (Linux, Apache, MySQL, PHP).Серед сайтів, які використовують PHP - Facebook, Wikipedia, Yahoo !, Baidu.

1.5.5 Sql

SQL – декларативна мова програмування, застосовуваний для створення, модифікації та управління даними в реляційній базі даних, керованої відповідною системою управління базами даних.

Спочатку SQL був основним способом роботи користувача з базою даних і дозволяв виконувати наступний набір операцій:

– створення в базі даних нової таблиці;
– додавання в таблицю нових записів;
– зміна записів;
– видалення записів;
– вибірка записів з однієї або декількох таблиць (відповідно до заданого умовою);
– зміна структур таблиць.

Згодом SQL ускладнився – збагатився новими конструкціями, забезпечив можливість опису та управління новими збереженими об'єктами (наприклад, індекси, уявлення, тригери і процедури) – і став набувати рис, властиві мовам програмування. При всіх своїх змінах SQL залишається найпоширенішим лінгвістичним засобом для взаємодії прикладного програмного забезпечення з

базами даних. У той же час сучасні СУБД, а також інформаційні системи, що використовують СУБД, надають користувачеві розвинені засоби візуального побудови запитів.

1.5.6 Python

Python – інтерпретована обєктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду.

Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.

Python підтримує динамічну типізацію, тобто, тип змінної визначається лише під час виконання. З базових типів слід зазначити підтримку цілих чисел довільної довжини і комплексних чисел. Python має багату бібліотеку для роботи з рядками, зокрема, кодованими в юнікоді. Система класів підтримує множинне успадкування і метапрограмування. Будь-який тип, включаючи базові, входить до системи класів, й за необхідності можливе успадкування навіть від базових типів.

Переваги. Кілька дизайнів сторінки для різних пристроїв перегляду. Наприклад, на екрані дизайн буде розрахований на велику ширину, під час друку меню не виводитиметься, а на КПК і стільниковому телефоні меню буде слідувати за вмістом.

Зменшення часу завантаження сторінок сайту за рахунок перенесення правил представлення даних в окремий CSS-файл. В цьому випадку браузер завантажує тільки структуру документа і дані, що зберігаються на сторінці, а представлення цих даних завантажується браузером тільки один раз і може бути закешовану.

Простота подальшої зміни дизайну. Не потрібно правити кожну сторінку, а досить лише змінити CSS-файл. Додаткові можливості оформлення. Наприклад, за допомогою CSS-верстки можна зробити блок тексту, який решта тексту буде обтікати (наприклад для меню) або зробити так, щоб меню було завжди видно при прокручуванні сторінки.

Незалежність від конкретної СУБД. Недивлячись на наявність діалектів і відмінностей в синтаксисі, в більшості своїй тексти SQL-запитів, що містять DDL і DML, можуть бути досить легко перенесені з однієї СУБД в іншу. Існують системи, розробники яких спочатку орієнтувалися на застосування щонайменше кількох СУБД (наприклад: система електронного документообігу Documentum може працювати як з Oracle Database, так і з Microsoft SQL Server і DB2). Природно, що при застосуванні деяких специфічних для реалізації можливостей такої переносимості добитися вже дуже важко.

Наявність стандартів і набору тестів для виявлення сумісності і відповідності конкретній реалізації SQL загальноприйнятому стандарту тільки сприяє «стабілізації» мови. Правда, варто звернути увагу, що сам по собі стандарт місцями занадто формалізований і роздутий в розмірах (наприклад, базова частина стандарту SQL: 2003 складається з більш ніж 1300 сторінок тексту).

Декларативність. За допомогою SQL програміст описує тільки те, які дані потрібно витягнути або модифікувати. Те, яким чином це зробити, вирішує

СУБД безпосередньо при обробці SQL-запиту. Однак не варто думати, що це повністю універсальний принцип - програміст описує набір даних для вибірки або модифікації, проте йому при цьому корисно уявляти, як СУБД збиратиме текст його запиту. Чим складніше сконструйований запит, тим більше він допускає варіантів написання, різних за швидкістю виконання, але однакових за підсумковим набору даних.

Недоліки. Різна відображення верстки в різних браузерах (особливо застарілих), які по-різному інтерпретують одні й ті ж дані CSS.

Найпоширеніша необхідність на практиці виправляти не тільки один CSS-файл, але і теги HTML, які складним і коханим способом пов'язані з селекторами CSS, що іноді зводить нанівець простоту застосування єдиних файлів стилів і значно збільшує час редагування та тестування.

Невідповідність реляційної моделі даних. Творці реляційної моделі даних Едгар Кодд, Крістофер Дейт і їх прихильники вказують на те, що SQL не є істинно реляційних мовою. Зокрема, вони вказують на такі дефекти SQL з точки зору реляційної теорії:

- допущення рядків-дублікатів в таблицях і результати вибірок, що в рамках реляційної моделі даних неможливо і неприпустимо;
- підтримка невизначених значень (NULL), яка створює фактично багатозначну логіку;
- значимість порядку стовпців, можливість посилань на стовпці за номерами (в реляційної моделі стовпці повинні бути рівноправні);
- допущення стовпців без імені, що дублюються імен стовпців.

В опублікованому Крістофером Дейтом і Х'ю Дарвені Третьому маніфесті вони викладають принципи СУБД наступного покоління і пропонують мову Tutorial D, який є справді реляційним.

Складність. Хоча SQL і замислювався як засіб роботи кінцевого користувача, пізніше він став настільки складним, що перетворився в інструмент програміста.

Відхилення від стандартів. Не дивлячись на наявність міжнародного

стандарту ANSI SQL-92, багато розробників СУБД вносять зміни в мову SQL, застосовуваний в розроблюваної СУБД, тим самим відступаючи від стандарту. Таким чином з'являються специфічні для кожної конкретної СУБД діалекти мови SQL.

Складність роботи з ієрархічними структурами. Раніше діалекти SQL більшості СУБД не пропонували способу маніпуляції деревоподібними структурами. Деякі постачальники СУБД пропонували свої рішення (наприклад, в Oracle Database використовується вираз CONNECT BY). В даний час в ANSI стандартизована рекурсивна конструкція WITH з діалекту SQL DB2. У Microsoft SQL Server рекурсивні запити (Recursive Common Table Expressions) з'явилися з версії 2005.

Браузери можуть запускати JavaScript поза пісочниці з привілеями, необхідними, наприклад, для створення і видалення файлів. Однак такі привілеї не повинні даватися коду з інтернету.

Неправильне наділення привілеями JavaScript з інтернету служило причиною вразливостей як Internet Explorer, так і Mozilla Firefox.

Microsoft Windows дозволяє файлам з кодом JavaScript запускатися як звичайними програмами без того, щоб бути поміщеними в пісочницю. Це робить можливим створення троянських програм.

Одже розглянувши основні мови для створення сайтів, було обрано CSS, HTML, тому що ці мови найпростіші у використанні та розумінні, достатній функціонал.

В першому розділі було розглянуто об’єкти автоматизації, обґрунтовано її необхідність. Було досліджено аналоги та проаналізовано їх функціональні можливості, та дизайн. Проведена постановка задачі та формулювання вимог до системи і опис засобів розробки.

2 ПРОЄКТУВАННЯ ВЕБСАЙТУ

2.1 Створення технічного завдання

Створення технічного завдання роботи здійснюється за такими етапами:

– технічне завдання (ТЗ);
– вихідний документ для проєктування сайту;
– розробка макета та дизайну;
– конструювання технічної частини (сайту, телеграм боту);
– створення програмного продукту або проведення науково-дослідних робіт (НДР) у відповідності до якого проводиться виготовлення;
– введення в експлуатацію.

2.1.1 Найменування і область застосування

Програмний продукт, що буде розробляться матиме назву «Інтернет-провайдер GreeNet». Програма призначена для перегляду інформації про інтернет провайдера GreeNet.

2.1.2 Підстава для розробки

Сайт розробляється на підставі наказу «Про затвердження тем дипломних робіт студентів освітньо-кваліфікаційного рівня «бакалавр» денної форми навчання Запорізького-національного університету».

2.1.3 Призначення розробки

Дана програма повинна забезпечувати користувача такими можливостями:

– перегляд інформації про провайдера;
– перегляд тарифів інтернет провайдера;
– перегляд спектру послуг які надає провайдер;
– перегляд звітів та документації провайдера;
– перегляд можливості зв’язку з інтернет провайдером або тех. підтримкою.

2.1.4 Вимоги до функціональних характеристик

Програма повинна забезпечити можливість виконання наступних функцій:

– забезпечити коректний перегляд інформації на сторінках;
– здійснення коректного переходу між сторінками;
– можливість коректного скачування файлів з сайту;
– перехід до особового кабінету через сайт.

2.1.5 Вхідні та вихідні дані

Вхідними даними є:

– головна сторінка;
– сторінка тарифікації;
– сторінка послуги;
– сторінка з документацією;
– сторінка техпідтримки;

– сторінка оголошень;
– сторінка гроза.

Вихідними даними є:

– посилання на документацію та можливість її скачати;
– посилання на особовий кабінет.

2.1.6 Етапи розробки сайту

Існує 6 етапів розробки вебсайтів:

– створення технічного завдання;
– проєктування сайту або вебдодатки(збір і аналіз вимог, розробка технічного завдання, проєктування інтерфейсів);
– розробка креативної концепції сайту;
– створення дизайн-концепції сайту;
– створення макетів сторінок;
– створення мультимедіа і FLASH-елементів;
– верстка сторінок і шаблонів;
– розробка серверної частини.

Залежно від поставленої задачі деякі етапи можуть бути відсутні або зв'язані в один.

Створення технічного завдання. Технічне завдання (ТЗ) (англ. Product Requirements Document; PRD) – документ, що встановлює основне призначення, показники якості, техніко-економічні та спеціальні вимоги до виробу, обсягу, стадії розроблення та складу конструкторської документації.

Технічне завдання є вихідним документом для проєктування споруди (архітектурного комплексу), конструювання технічного пристрою (приладу, машини тощо), розробки автоматизованої системи чи інформаційної системи, створення програмного продукту, проведення науково-дослідних робіт (НДР) і дослідно-конструкторських робіт (ДКР).

Технічне завдання на надання послуг встановлює основні вимоги до робіт в певній сфері діяльності, обов'язки замовника і виконавця, показники якості, терміни проведення і звітності та економічні показники послуг. Цей документ використовують окремо або у складі договору(контракту) на виконання робіт.

Розробка креативної концепції сайту. Починається робота зі створення дизайну, зазвичай в графічному редакторі. Дизайнер створює один або кілька варіантів дизайну, відповідно до технічного завдання. При цьому окремо створюється дизайн головної сторінки, і дизайни типових сторінок (наприклад: статті, новини, каталог продукції). Власне «дизайн сторінки» представляє собою графічний файл, листковий малюнок, що складається з найбільш дрібних картинок-шарів елементів загального малюнка [4].

При цьому дизайнер повинен враховувати обмеження стандартів HTML (не створювати дизайн, який потім не зможе бути реалізований стандартними засобами HTML). Виняток становить Flash-дизайн.

Кількість ескізів і порядок їх надання обмовляється з проєкт-менеджером. Також менеджер проєкту здійснює контроль термінів. У великих вебстудіях в процесі бере участь арт-директор, який контролює якість графіки. Етап також закінчується затвердженням ескізу замовником.

Створення дизайн-концепції сайту. Створення концепції дизайну – це самостійний розділ проєктної діяльності, який полягає у виконанні робіт, спрямованих на визначення основних критеріїв, завдань і цілей оформлення (візуалізації) будь-якого об'єкта з урахуванням його смислового (ідейно-тематичного) призначення.

HTML-верстка. Затверджений дизайн передається HTML-верстальник, який «нарізає» графічну картинку на окремі малюнки, з яких згодом складає HTML-сторінку. В результаті утворюється код, який можна переглядати за допомогою браузера.

А типові сторінки згодом будуть використовуватися як шаблони.

Далі готові HTML-файли передають програміст. Програмування сайту може здійснюватися як «з нуля», так і на основі CMS – системи управління

вмістом. У випадку з CMS треба сказати, що сама «CMS» в деякому сенсі це готовий сайт, що складається з замінних частин. «Програміст» – в даному випадку правильно буде називати його просто фахівцем по CMS – повинен замінити стандартний шаблон, що поставляється з CMS, на оригінальний шаблон.

Цей оригінальний шаблон він і повинен створити на основі вихідного «вебдизайну».

При програмуванні сайту фахівця призначаються контрольні точки термінів. Завершальним етапом розробки сайту є тестування. Процес тестування може включати в себе найрізноманітніші перевірки: вид сторінки зі збільшеними шрифтами, при різних розмірах вікна браузера, при відсутності флеш-плеєра і багато інших.

2.2 Діаграма прецедентів

UML (Universal Modeling Language) – універсальна мова моделювання, який був розроблений компанією Rational Software з метою створення найбільш оптимального та універсальної мови для опису як предметної області, так і конкретного завдання в програмуванні. Візуальне моделювання в UML можна уявити як певний процес порівневого спуску від найбільш загальної і абстрактної концептуальної моделі системи до логічної, а потім і до фізичної моделі відповідної системи. Будь-яке завдання, таким чином, моделюється за допомогою деякого набору ієрархічних діаграм, кожна з яких представляє собою деяку проекцію системи [1].

Діаграма (Diagram) – це графічне представлення безлічі елементів.

Найчастіше вона зображується у вигляді зв'язного графа з вершинами (сутностями) і ребрами (відносинами).

Діаграми прецедентів застосовуються для моделювання виду системи з точки зору зовнішнього спостерігача. На діаграмі прецедентів графічно показана сукупність прецедентів та Суб'єктів, а також відносини між ними.

Розглянемо основні елементи діаграми прецедентів. Суб'єкт (actor) – будь-яка сутність, що взаємодіє з системою ззовні або безліч логічно пов'язаних ролей, виконуваних при взаємодії з прецедентами.

Стандартним графічним позначенням суб'єкта на діаграмах є фігурка «чоловічка», під якою записується конкретне ім'я суб'єкта, проте суб'єктом може бути не тільки людина, але і технічне пристроїв о, програма або будь-яка інша система, яка може служити джерелом впливу на моделируемую систему так, як визначить сам розробник (рис. 2.1).

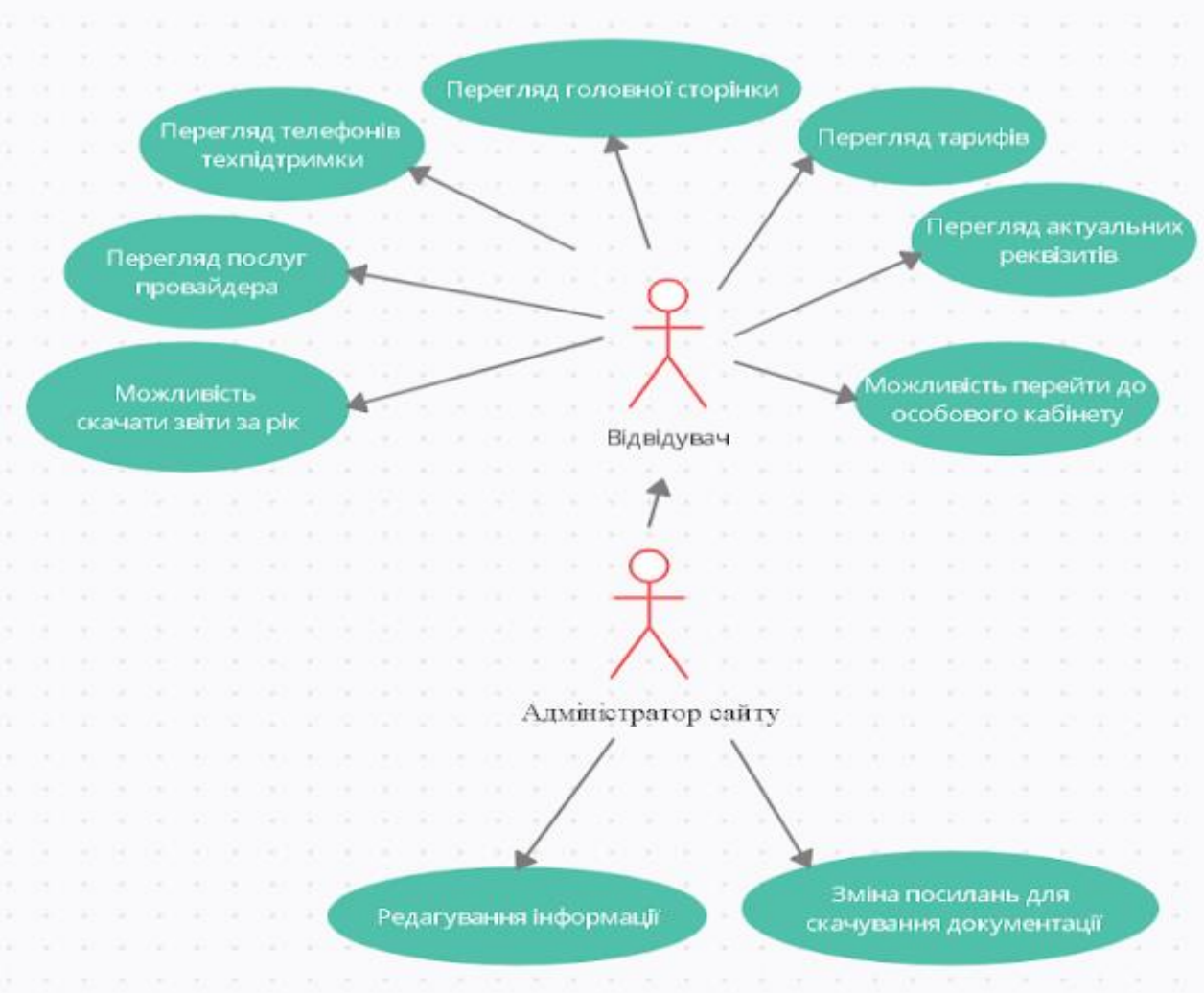

Рисунок 2.1 – Діаграма прецедентів

Прецеденти (use case) – це опис безлічі послідовностей дій (включаючи їх варіанти), які виконуються системою для того, щоб актор отримав результат, який має для нього певне значення. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодію суб'єктів з системою, це одна з найважливіших особливостей розробки прецедентів. Стандартним графічним позначенням прецеденту на діаграмах є еліпс, всередині якого міститься коротка назва прецеденту або ім'я у формі дієслова з пояснювальними словами.

Діаграма прецедентів демонструє, що для сайта який проєктується, передбачено 2 види користувачів: відвідувачі і адміністратор.

2.3 Макет сайту

Дизайн-макет сайту – це візуальний образ майбутнього сайту,розроблений з урахуванням технічних можливостей HTML верстки. Такий макет є демонстрацією того, як візуально буде виглядати ваш сайт після верстки і наповнення [2].

Макет представляється у вигляді картинки, яка буде відображена в інтернет браузері, без активних кнопок і інших динамічних елементів.

Специфіка розробки графічного дизайн-макету стосовно сайту представляє з себе поєднання технічних і візуальних параметрів майбутнього сайту. Це опрацювання розташування і розміру елементів сайту з точки зору зручності пошуку і використання інформації на сайті (див. рис. 2.2).

В другому розділі ми розглянули проєктування вебдодатку. Було створено технічне завдання, діаграма прецедентів, макети сайту та описано етапи розробки сайту. Сайт був оформлений в одному стилі (шаблоні), задля приємного користування.

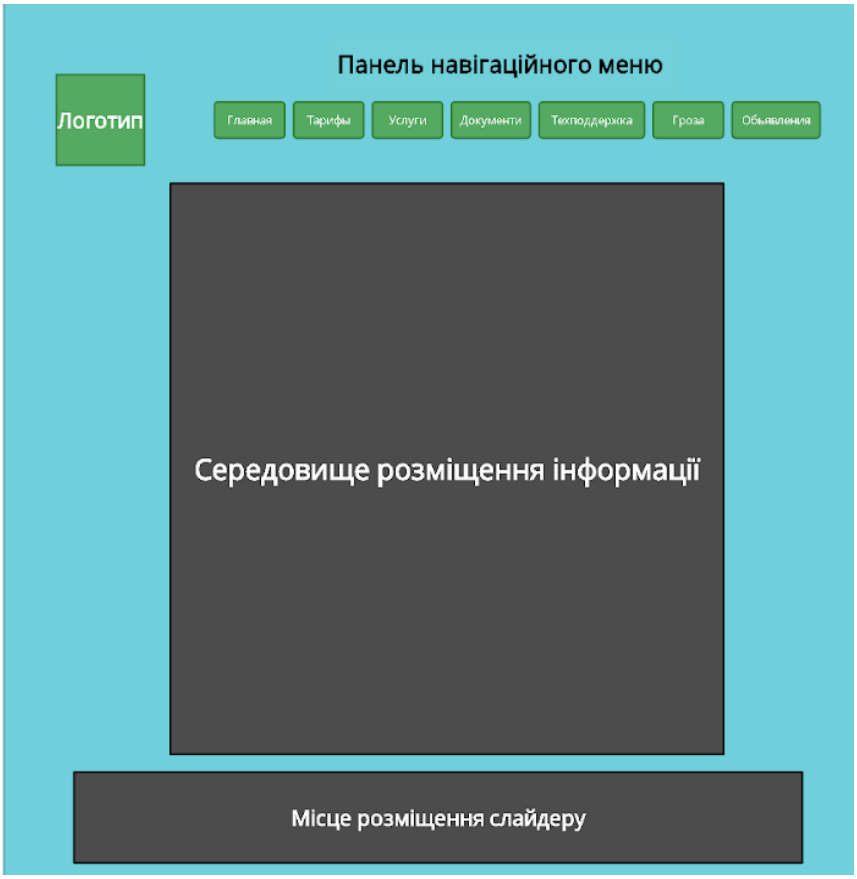

Рисунок 2.2 – Макет сайту

3 РОЗРОБКА ВЕБСАЙТУ ТА ТЕЛЕГРАМ БОТУ

3.1 Інструменти для створення дизайну

У наш час дизайн сайту це найголовніша складова, бо користувачі цінують зовнішній вигляд, саме картинку.

В інтернеті можна побачити багато прикладів, та кожен сайт це сугубо стиль замовника або розробника, те як саме він бачить та як йому подобається.

Макет сайту можна створювати за допомогою будь яких засобів де можна малювати, або наявні геометричні фігури. Як приклад инструмент Creately. Creately - це інструмент візуальної спільної роботи SaaS з можливостями побудови діаграм і дизайну, розроблений Cinergix. Creately має дві версії: хмарну онлайн-версію і завантажується офлайн-версію для ПК, сумісну з Windows, Mac і Linux.

Наприклад, переходячи до онлайн версії, користувача зустрічає меню з вибором шаблонів для тої роботи, яку ви збираєтесь виконати (рис. 3.1).

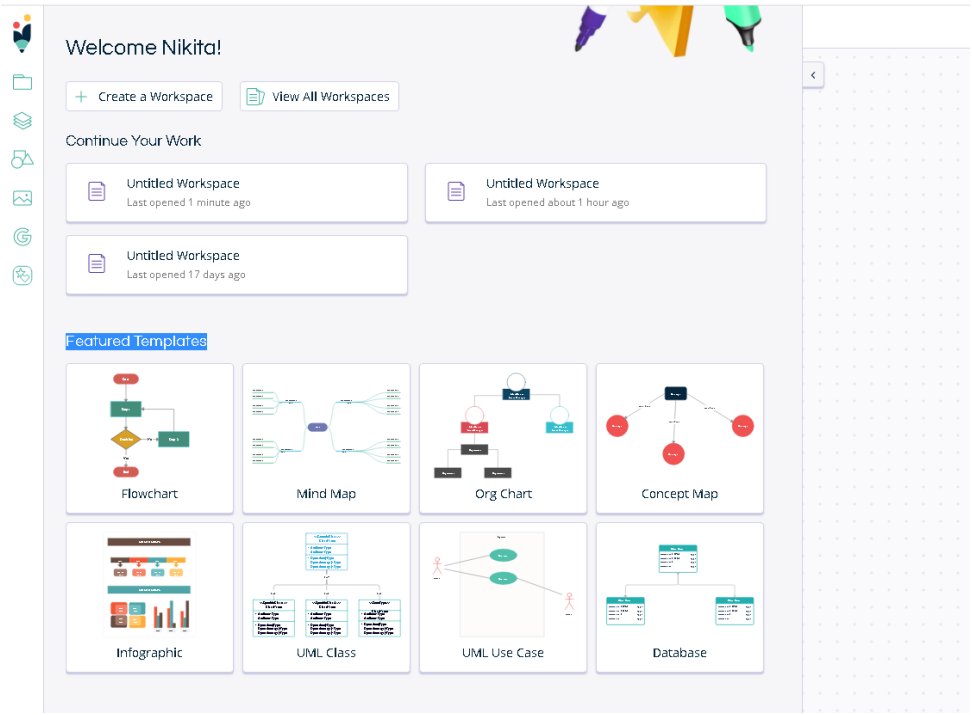

Рисунок 3.1 – Початок роботи з Creately

Далі після вибору шаблону або продовження роботи на чистому середовищі, можна побачити блоки, лінії, ввод тексту, та знайти потрібні вам блоки (рис. 3.2).

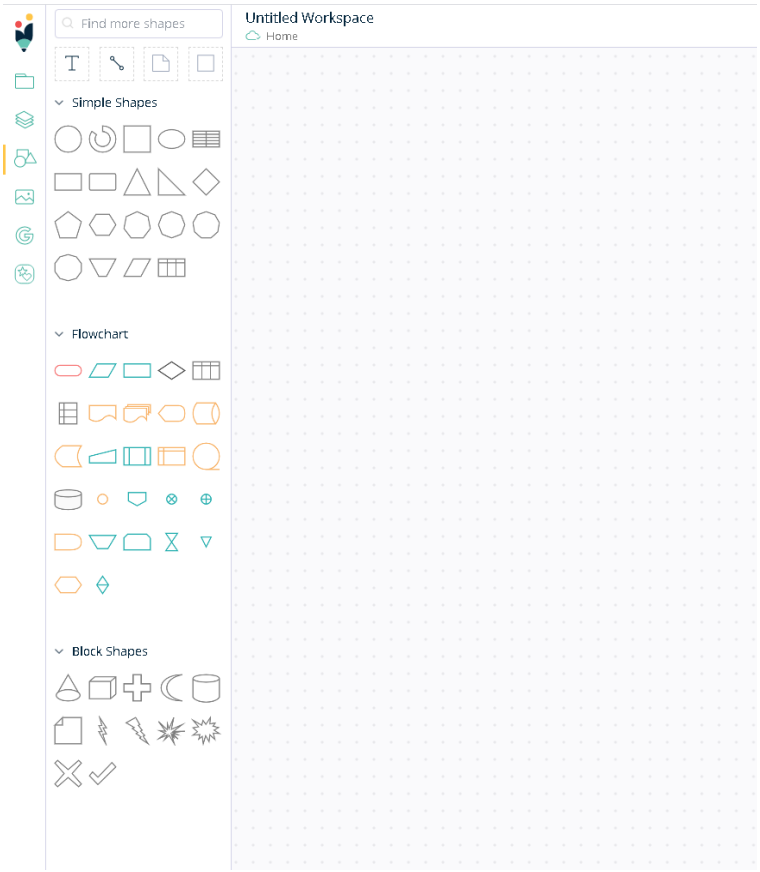

Рисунок 3.2 – Робоче середовище Creately

Також важливу роль відіграє палітра кольорів та стиль тексту.

У просторі наявні сайти, на яких зібрані різні шрифти, і можна обрати собі до вподоби. Як приклад сайт fonts-online[https://www.fonts-online], на ньому можна ввести потрібні вам слова, та одразу бачити як буде виглядати у різних стилях, багатий вибір тематика навіть наявна панель для зміни кольору з тегом для html коду (див. рис. 3.3).

Кольори та їх тегування можна знайти, просто вбивши у браузері в пошук слова кольори html, як приклад сайт ColorSheme, де наявні всі основні кольори та їх теги і rgb координати (див. рис. 3.4).

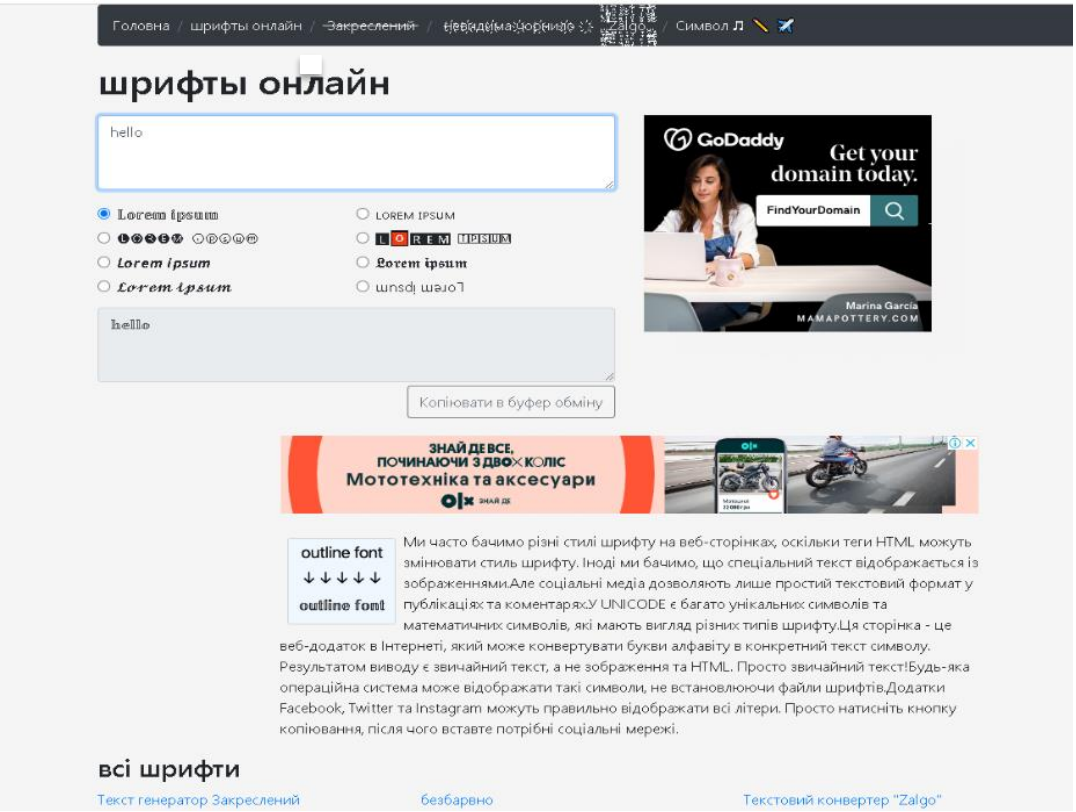

Рисунок 3.3 – Сайт для вибору шрифтів

Рисунок 3.4 – Таблиця html кольорів

3.2 Розробка сторінок сайту

Переходячи на сайт, користувача зустрічає Головна сторінка основною інформацією про провайдера та сайт (рис. 3.5).

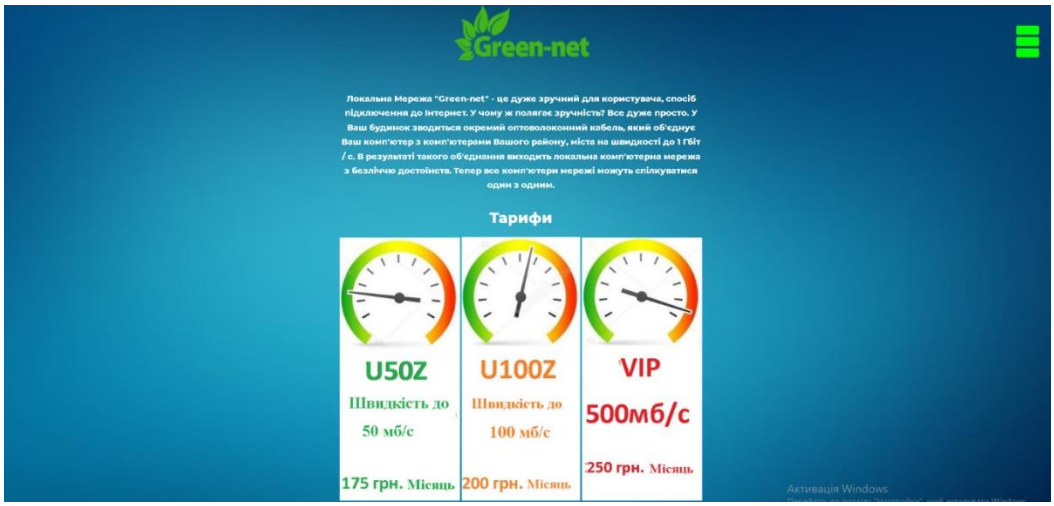

Рисунок 3.5 – Основна сторінка

Меню на Головній сторінці розроблено як бутербродне, аби не перекривати основний контент, шапку сайту займав логотип, та сторінка було простішою. При натисканні користувачу вилазить меню, яке також при натисканні можна згорнути, де він може побачити сторінки які наявні на сайті та перейти на них, при наведенні на напис, він змінює колір на зелений (рис. 3.6).

Рисунок 3.6 – Меню Головної сторінки

Таке меню реалізовано за допомогою CSS коду, та підключено до html документу (рис. 3.7).

```
/*TEXT*/
a {
  text-decoration: none;
  color: #4a4646;
  transition: color 0.3s ease;
}

a:hover {
  color: #00FF00;
}
/*Block menu*/
#asekandutd-ginevitable {
  display: block;
  position: absolute;
  top: 48px;
  right: 48px;
  z-index: 1;
  -webkit-user-select: none;
  user-select: none;
}
/*Block menu when activeted*/
#asekandutd-ginevitable input {
  display: block;
  width: 53px;
  height: 45px;
  position: absolute;
  top: -5px;
  left: -4px;
  cursor: pointer;
  opacity: 0;
  z-index: 2;
  -webkit-touch-callout: none;
}
/*Block menu when b*/
#asekandutd-ginevitable span {
  display: block;
  width: 50px;
  height: 20px;
  margin-bottom: 5px;
  position: relative;
  background: #00FF00;
  border-radius: 3px;
  z-index: 1;
  transform-origin: 5px 0px;
  transition: transform 0.7s cubic-bezier(0.77,0.2,0.05,1.0),
background 0.5s cubic-bezier(0.77,0.2,0.05,1.0),opacity 0.55s ease;
}

#asekandutd-ginevitable span:first-child {
  transform-origin: 0% 0%;
}

#asekandutd-ginevitable span:nth-last-child(2) {
  transform-origin: 0% 100%;
}

#asekandutd-ginevitable input:checked ~ span {
  opacity: 1;
  transform: rotate(45deg) translate(-2px, -1px);
  background: #1b1919;
```

Рисунок 3.7 – Реалізація меню головної сторінки

При переході користувачем на сторінку Про нас, змінюється меню, воно знаходиться у шапці, де також у лівій стороні наявний логотипу правій годинник. При наведенні на кнопку меню, напис змінює колір, а кнопка підводиться білою лінією (рис. 3.8).

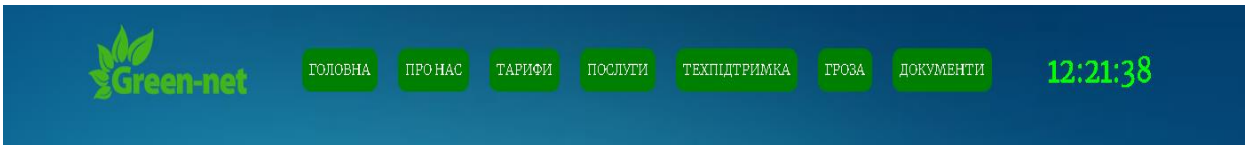

Рисунок 3.8 – Шапка сторінки Про нас

Меню створено за допомогою Css, задано стиль, розміри та функції (рис. 3.9).

```
/*Menu*/
.nav__link{
    display:inline-block;
    vertical-align:top;
    margin: 0 10px;
    position:relative;
color:#fff;
text-decoration:none;
transition:color 0.2s liner;

    background: #008000; /*  цвет фона */
    padding: 0.5rem 0.5rem; /* Поля вокруг текста */
    border-radius: 10px; /* Скругляем уголки */
}

.nav__link:after{ /*Полоска подчеркивания меню */
    content:"";
    display:block;
    width:100%;
    height:3px;
    display:none;
    background-color:white;
    position:absolute;
    top:100%;
    left:0;
    z-index:1;
}

.nav__link:hover{
color:#00FF00;
}

.nav__link:hover:after{
    display:block;
}
```

Рисунок 3.9 – Реалізація меню сторінки Про нас

Годинник створено за допомогою JavaScript, створюється змінна і підв’язується до системного годинник (рис. 3.10).

```
/*clock*/
function clock(){
  var date = new Date(),
        hours = (date.getHours() < 10) ? '0' + date.getHours() : date.getHours(),
        minutes = (date.getMinutes() < 10) ? '0' + date.getMinutes() : date.getMinutes(),
        seconds = (date.getSeconds() < 10) ? '0' + date.getSeconds() : date.getSeconds();
  document.getElementById('clock').innerHTML = hours + ':' + minutes + ':' + seconds;
}
setInterval(clock, 1000);
clock();
```

Рисунок 3.10 – Реалізація годинника

Кожен заголовок сторінки анімовано, він з’являється поступово зліва на право по одній букві з затримкою у 3.5 секунди, реалізовано за допомогою CSS (рис. 3.11).

```
/*animation Заголовки*/
.satodsan-uvemopag {
  color: #008000;
  font-family: monospace;
  overflow: hidden;

  white-space: nowrap;
  margin: 0 auto;
  letter-spacing: .17em;
  animation:
  typing 3.5s steps(30, end),
  blink-caret .7s step-end infinite;
}

@keyframes typing {
  from { width: 0 }
  to { width: 100% }
}
```

Рисунок 3.11 – Реалізація анімації заголовків

Також біля кожного заголовку по тематиці наявні стікери, реалізовані копіюванням стікерів у інтернеті та введені їх як звичайний текст, браузер читає їх самостійно, не потребується кодування (рис. 3.12).

Рисунок 3.12 – Заголовки зі смайлами

На сторінці наявний слайдер, з промо провайдера (рис. 3.13).

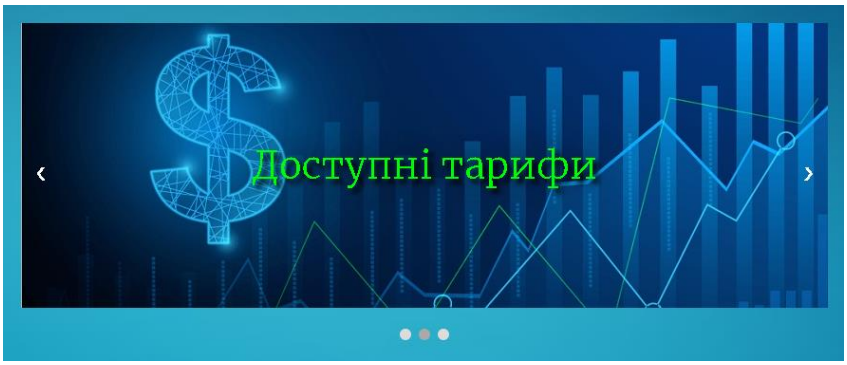

Рисунок 3.13 – Слайдер

Слайдер реалізовано за допомогою JavaScript та стилізовано у CSS, код створює область слайдера, дозволяє перемикати вперед і назад, як за допомогою кнопок внизу так і стрілок з обох сторін (рис. 3.14).

```
/* Основная функция слайдера */
function showSlides(n) {
    var i;
    var slides = document.getElementsByClassName("item");
    var dots = document.getElementsByClassName("slider-dots_item");
    if (n > slides.length) {
      slideIndex = 1
    }
    if (n < 1) {
        slideIndex = slides.length
    }
    for (i = 0; i < slides.length; i++) {
        slides[i].style.display = "none";
    }
    for (i = 0; i < dots.length; i++) {
        dots[i].className = dots[i].className.replace(" active", "");
    }
    slides[slideIndex - 1].style.display = "block";
    dots[slideIndex - 1].className += " active";
}
```

Рисунок 3.14 – Реалізація слайдера у JavaScript

Розміри та місця розміщень кнопок, стилі заголовків та колір написів (рис. 3.15).

```
130
131     /* Собственно сам слайдер */
132     .slider{
133         max-width: 90%;
134         position: relative;
135         margin: auto;
136         height: 300px;
137         margin-bottom: 15px;
138     }
139     /* Картинка мастабируется по отношению к родительскому элементу */
140     .slider .item img {
141         object-fit: cover;
142         width: 100%;
143         height: 300px;
144         border: none !important;
145         box-shadow: none !important;
146     }
147     /* Кнопки вперед и назад */
148     .slider .prev, .slider .next {
149         cursor: pointer;
150         position: absolute;
151         top: 0;
152         top: 50%;
153         width: auto;
154         margin-top: -22px;
155         padding: 16px;
156         color: white;
157         font-weight: bold;
158         font-size: 18px;
159         transition: 0.6s ease;
160         border-radius: 0 3px 3px 0;
161     }
162     .slider .next {
163         right: 0;
164         border-radius: 3px 0 0 3px;
```

Рисунок 3.15 – Стилізація у css слайдера

Стиль тексту на кожній сторінці підключено за допомогою скачування та підключення через посилання у корінь знаходження шрифту, шрифт обрано Inter bold, текст білого кольору (рис. 3.16).

Рисунок 3.16 – Стиль тексту

Переходячи на сторінку Тарифні плани, користувач може побачити таблиці із назвами, швидкістю, та ціною тарифу (рис. 3.17).

Рисунок 3.17 – Таблиці із тарифами

На сторінці Техпідтримка, користувач може побачити номера телефонів для зв’язку з операторами, та кнопку переходу до особового кабінет інтернет провайдера, також реалізовано кнопку переходу до телеграм боту для зворотнього зв’язку з абонентами (рис. 3.18).

Рисунок 3.18 – Сторінка Техпідтримки

Особовий кабінет інтернет провайдера (рис. 3.19).

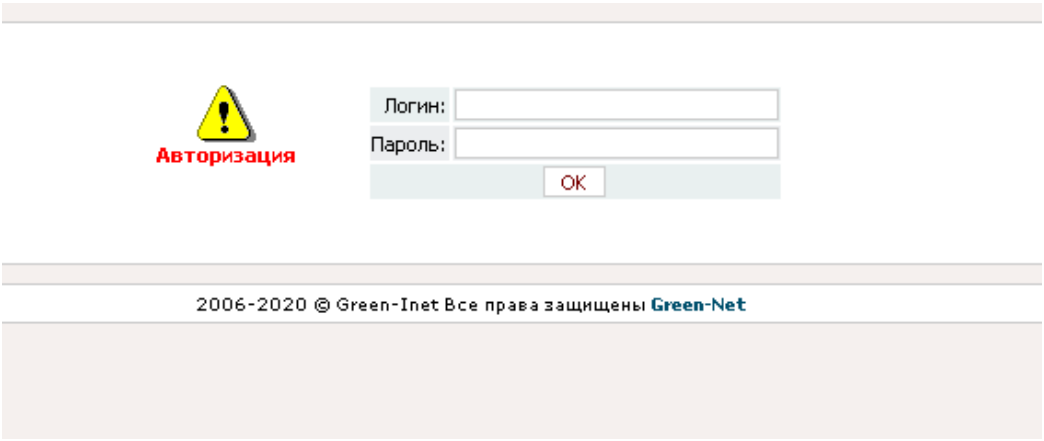

Рисунок 3.19 – Особовий кабінет

На вкладці Гроза, напис кожні 2 секунди мигає білим кольором, реалізовано за допомогою CSS, створюється область, яка оминає об'єкти, і навколо них створює біле блимання(рис. 3.20).

```
.area {
  text-align: center;
  font-size: 70px;
  color: #008000;
  letter-spacing: 7px;
  font-weight: 750;
  text-transform: uppercase;
  animation: blur .95s ease-out infinite;
  text-shadow: 0px 0px 5px #fff, 0px 0px 7px #fff;
}

@keyframes blur {
  from {
    text-shadow:0px 0px 10px #fff,
      0px 0px 10px #fff,
      0px 0px 25px #fff,
      0px 0px 25px #fff,
      0px 0px 25px #fff,
      0px 0px 25px #fff,
      0px 0px 25px #fff,
      0px 0px 25px #fff,
      0px 0px 25px #fff,
      0px 0px 50px #fff,
      0px 0px 50px #fff,
      0px 0px 50px #7B96B8,
      0px 0px 150px #7B96B8,
      0px 10px 100px #7B96B8,
      0px 10px 100px #7B96B8,
      0px 10px 100px #7B96B8,
      0px 10px 100px #7B96B8,
      0px -10px 100px #7B96B8,
      0px -10px 100px #7B96B8;
  }
}
```

Рисунок 3.20 – Реалізація блимання заголовку Гроза

Також на сторінці Гроза наявний прогноз погоди (рис. 3.21).

Рисунок 3.21 – Прогноз погоди

Реалізовано за допомогою посилання на сайт з прогнозом погоди, та створено рамки для нього (рис. 3.22).

```
<!--Погодник-->
<center>
<iframe width="200" height="240" scrolling="no" src="https://pogodnik.com/informer/daily"> Посмотреть прогноз погоды на <a href="https://pogodnik.com"> <img src="https://pogodnik.com/images/prognoz-pogody-200x240.png"> </a> </
</center>
```

Рисунок 3.22 – Реалізація прогнозу погоди

Також на сторінці наявний невеликий список правил безпеки користування мережею під час несприятливих погодних умов (рис. 3.23).

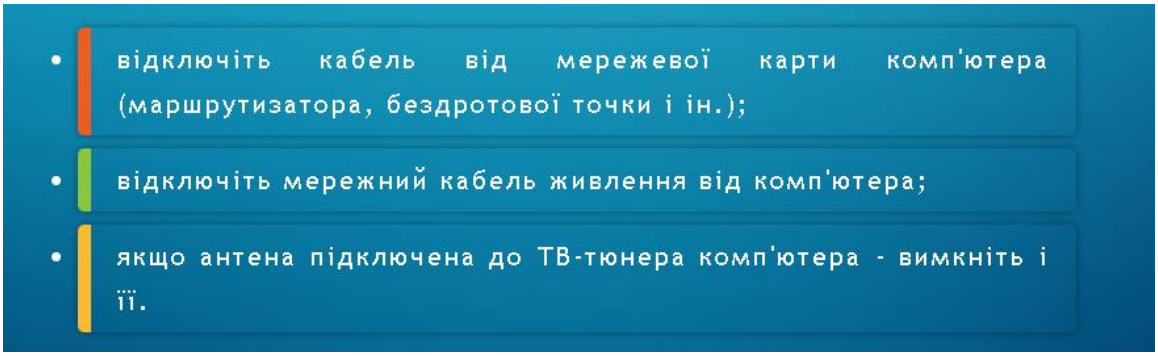

Рисунок 3.23 – Список

Реалізовано список за допомогою html (рис. 3.24).

```
   <!--Style spisok-->
   .border {
list-style: none;
padding: 0;
}
.border li {
font-family: "Trebuchet MS", "Lucida Sans";
padding: 7px 20px;
margin-bottom: 10px;
border-radius: 5px;
border-left: 10px solid #f05d22;
box-shadow: 2px -2px 5px 0 rgba(0,0,0,.1),
     -2px -2px 5px 0 rgba(0,0,0,.1),
    2px 2px 5px 0 rgba(0,0,0,.1),
    -2px 2px 5px 0 rgba(0,0,0,.1);
font-size: 20px;
letter-spacing: 2px;
transition: 0.3s all linear;
}
.border li:nth-child(2){border-color: #8bc63e;}
.border li:nth-child(3){border-color: #fcba30;}
.border li:nth-child(4){border-color: #1ccfc9;}
.border li:nth-child(5){border-color: #493224;}
.border li:hover {border-left: 10px solid transparent;}
.border li:nth-child(1):hover {border-right: 10px solid #f05d22;}
.border li:nth-child(2):hover {border-right: 10px solid #8bc63e;}
.border li:nth-child(3):hover {border-right: 10px solid #fcba30;}
.border li:nth-child(4):hover {border-right: 10px solid #1ccfc9;}
.border li:nth-child(5):hover {border-right: 10px solid #493224;}
```

Рисунок 3.24 – Реалізація списку

На сторінці Документи, користувач може побачити посилання на звіт провайдера про якість послуг за останні два роки та договір про надання послуг (рис. 3.25).

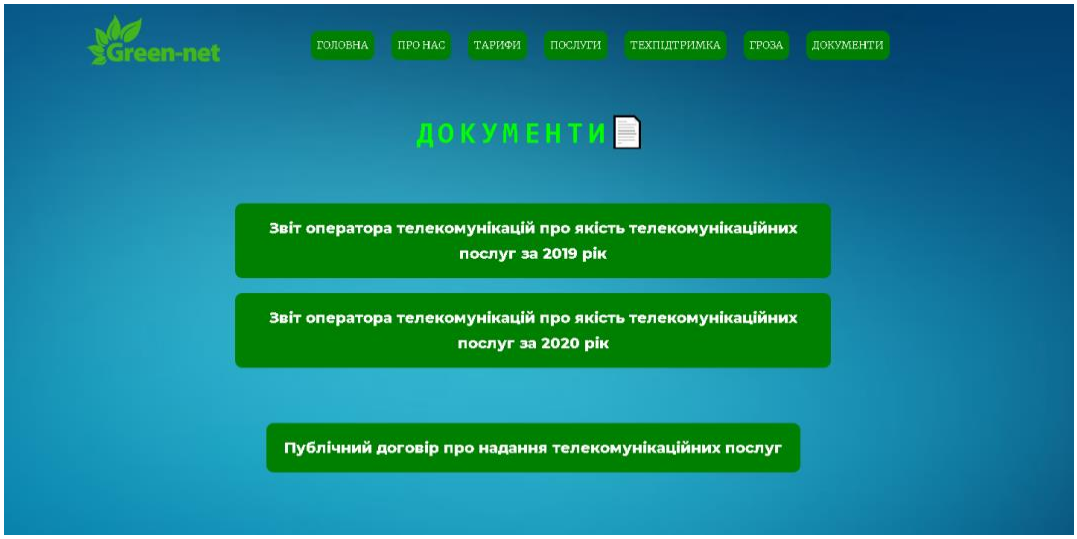

Рисунок 3.25 – Сторінка Документи

При натисканні на посилання відкривається у браузері pdf файл з інформацією (рис. 3.26).

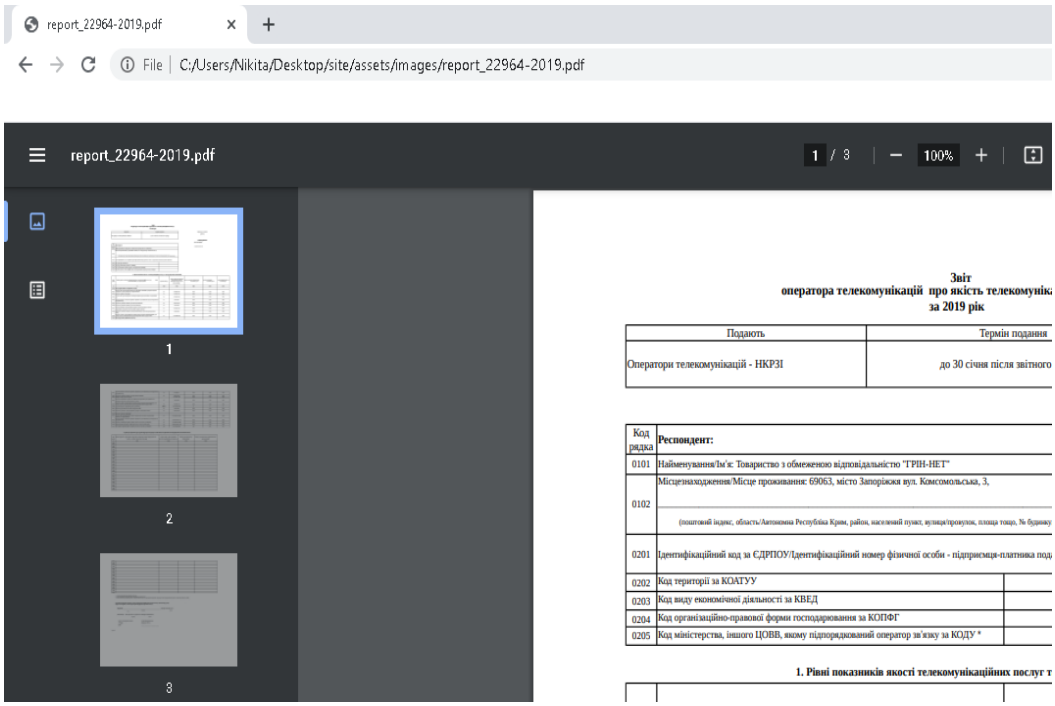

Рисунок 3.26 – Pdf файли

3.3 Розробка телеграм боту

Натиснувши на кнопку TELEGRAM BOT, на сторінці сайту Техпідтримка, перехід здійснюється до телеграм боту за посиланням (рис. 3.27).

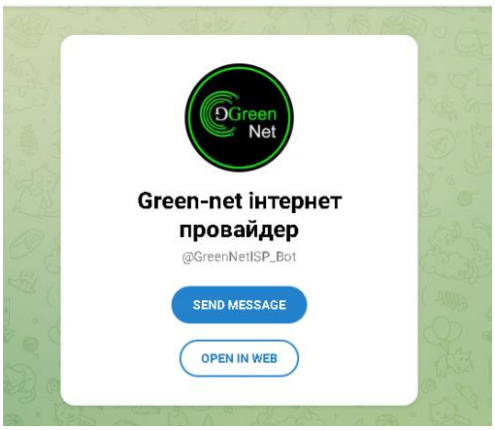

Рисунок 3.27 – Перехід до телеграм боту

Натиснувши на одну з кнопок вас перекидає до телеграм боту (рис. 3.28).

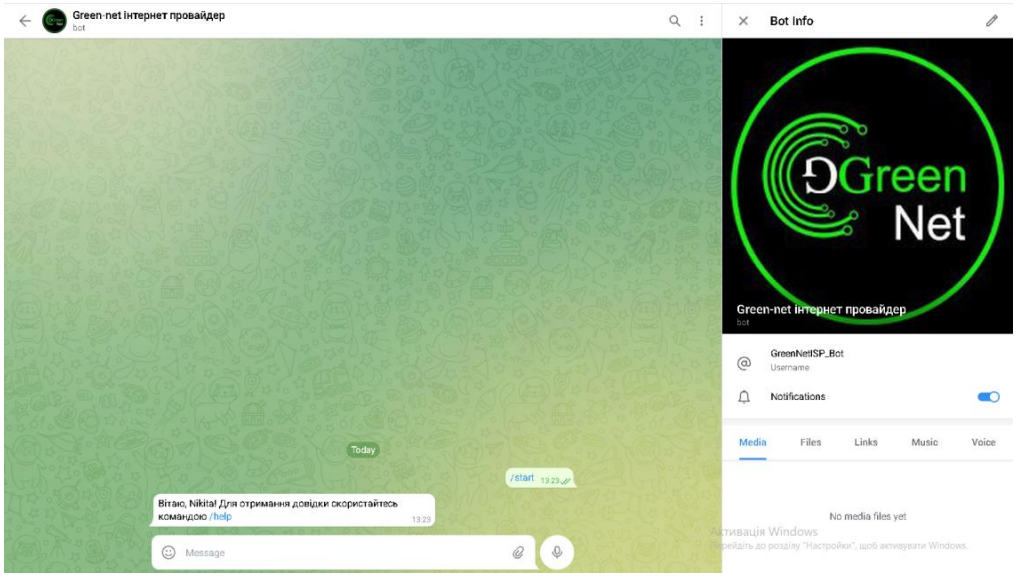

Рисунок 3.28 – Телеграм бот

Далі за допомогою команди help, телеграм бот запитує, чим може допомогти, та видає меню з навігацією (рис. 3.29).

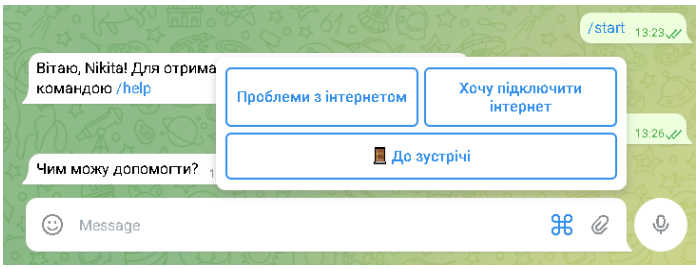

Рисунок 3.29 – Навігація телеграм боту

Далі телеграм бот пропонує сценарії, якщо обрати сценарій з проблемами з інтернетом, телеграм бот видає інформацію що потрібно перезавантажити роутер і далі запитає чи вирішило це проблему, якщо користувач відповідає так, телеграм бот повертає користувача на головну.

Якщо користувач обрав варіант проблему не вирішено, то телеграм бот просе внести номер телефону для зв’язку з оператором, після вводу номеру

користувачем, повертає на головну. Якщо абонент ввів неправильний формат номеру то телеграм бот видає текстове повідомлення про неправильний формат номеру.

При виборі користувачем сценарію хочу підключити інтернет,бот інформує користувача про доступні наразі тарифні плани (рис. 3.30).

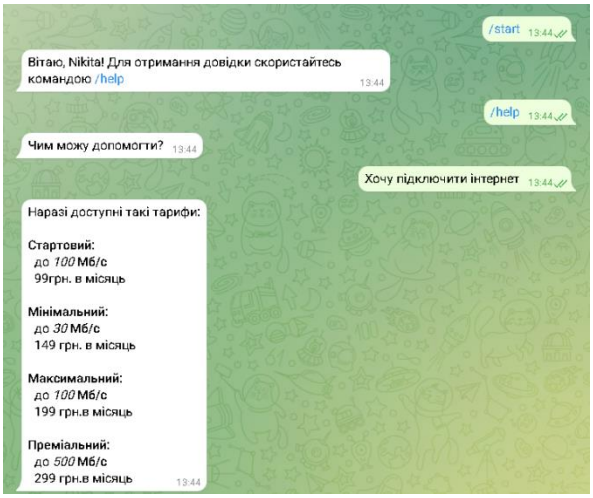

Рисунок 3.30 – Бот інформує про тарифні плани

Далі за сценарієм можна обрати або повернення на головну сторінку або обрати клавішу готовий підключитись. Якщо ви вже є абонентом цього інтернету або подавали заявку раніше,але обрали варіант готовий підключитись,бот інформує вас про це. Якщо ж ви новий клієнт,та вашого номеру телефону немає в базі,аналогічно запитує номер як і зі сценарієм про проблеми з інтернетом які не вдалось вирішити.

Телеграм бот був розроблений за допомогою мови програмування python, з використанням бібліотеки aiogram – синхронний фреймворк для створення телеграм ботів, та бази даних SQlite.

Розпочнемо з розгляду бази даних (див. рис. 3.31, 3.32).

```
    # Імпорт Необхідних Бібліотек
    import sqlite3
    class Database:
        def __init__(self, db_file):
            """Ініціалізація єкземплярів класа після їх створення"""
            self.conn = sqlite3.connect(db_file)
            self.cursor = self.conn.cursor()
        # def create_table_recipe(self):
        #     sql = '''
        #     CREATE TABLE Customers (
        #     id integer primary key autoincrement ,
        #     telegram_id integer,
        #     fix_status integer,
        #     cell_number varchar(15),
        #     need_connection integer
            def customer_exists(self, customer_id) -> bool:
            """Перевірка користувача в базі"""
            result = self.cursor.execute("Select telegram_id from Customers where telegram_id = ?",
(customer_id, ))
            return bool(len(result.fetchall()))
        def show_customers_to_fix(self, status):
            """Вивід списка користувачів, котрим необхідно лагодити інтернет"""
            result = self.cursor.execute("Select `cell_number` from Customers where fix_status = ?", (status,
))
            return result.fetchall()
        def show_customers_to_connect(self, status):
            """Вивід списка користувачів, яким необхідно підключити інтернет"""
            result = self.cursor.execute("Select cell_number from Customers where need_connection = ?",
(status,))
            return result.fetchall()
        def add_customer(self, customer_id, cell_number):
            """Додати користувача в базу"""
            self.cursor.execute(
                "Insert into Customers (telegram_id, fix_status, cell_number, need_connection) values (?, ?, ?,
?)",  (customer_id, 0, cell_number, 1, ))
            return self.conn.commit()
        def update_fix_status_to_one(self, cell_number, customer_id):
            """Оновлення статусу ремонту інтернету з 1 на 0"""
            self.cursor.execute("Update Customers set fix_status = 1, cell_number = ? where telegram_id =
?",
                                (cell_number, customer_id, ))
            return self.conn.commit()
        def update_fix_status_to_zero(self, customers_id):
            """Оновлення статусу ремонту інтернета з 0 на 1"""
            self.cursor.execute("Update Customers set fix_status = 0 where telegram_id = ?", (customers_id,))
            return self.conn.commit()
        def update_connection_status_by_command(self, cell_number):
            """Оновлення статусу підключення інтернету по команді через бота"""
            self.cursor.execute("Update Customers set need_connection = 0 where cell_number = ?",
(cell_number, ))
            return self.conn.commit()
        def close(self):
            """Розрив з'єднання з базой"""
            self.conn.close()
```

Рисунок 3.31 – Код

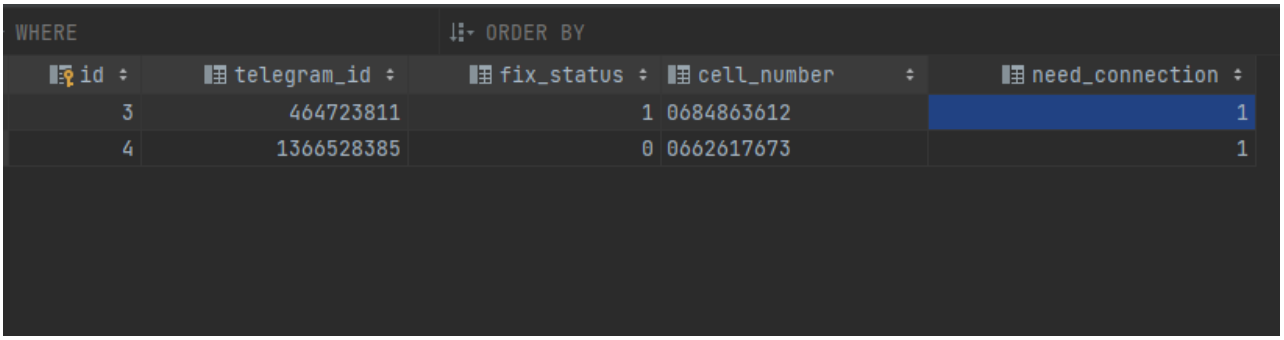

Рисунок 3.32 – База даних телеграм боту

Ми можемо бачити що база виводить телеграм айди за допомогою бібліотеки iogram, далі fix_status – показує 0 якщо абонент не вказував що має проблеми з інтернетом, 1 – якщо має. Номер телефону в комірці cell_number – номер користувача який він вказав та чи хоче абонент підключитись – 1, якщо абонент не хоче або вже підключений – 0.

Розглянемо реалізацію функціоналу бота якщо абонент хоче підключитись (див. рис. 3.33, 3.34).

Бот перевіряє чи наявний користувач в базі даних за його ID телеграм, якщо ні то запрощує до введення номеру телефону, для внесення в базу даних та подальшого контактування з ним через оператора і виведе меню з поверненням на головну. Якщо ж користувач вже наявний в базі то бот виведе повідомлення з кнопкою повернення на голову (рис. 3.33).

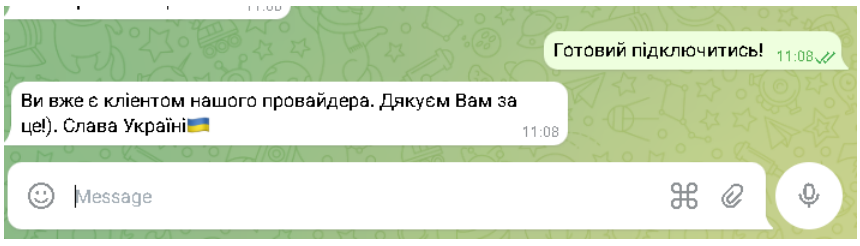

Рисунок 3.33 – Повідомлення абонента який вже є в базі бота

```
# Кнопки для меню роботи з ботом
ConfirmMenu = ReplyKeyboardMarkup(
    [       [KeyboardButton(text="На головну ")]
    ], resize_keyboard=True
)# Декоратор який реагує на вказаний текст
@dp.message_handler(text="Готовий підключитись!")
async def confirm_new_customer(message: types.Message):
```

```
        """Перевірка користувача в базі. Якщо такий користувач відсутній, то ініціалізується
        слухач повідомлень на наяву валідного номера телефону"""
        # Перевірка користувача в базі, якщо він відсутній, то бот присилає повідомлення-запрощення для
вводу телефона
        if not db.customer_exists(message.from_user.id):
            await message.answer(
                "Залиште свій номер телефону, з Вами зв`яжєтся наш оператор для узгодження дати та часу
підключення",
                reply_markup=ConfirmMenu)
            # Прослуховування подальшого повідомлення на наявність номеру
            await ConfirmNew.cell_number.set()
        else:
            # В випадку, якщо користувач є в базі, то бот інформує його і не надає можливості для
            # повторної реїстрації
            await message.answer("Ви вже є кліентом нашого провайдера. Дякуєм Вам за це!). Слава УкраїніUA",
                                 reply_markup=ConfirmMenu)
    # Декоратор який працює з вказаним стейтом
    @dp.message_handler(state=ConfirmNew.cell_number)
    async def confirm_new_customer(message: types.Message, state: FSMContext):
        """Отримання валідного номера телефону, перевірка його за допомогою регулярного вираження та
подальший запис до бази
         за умови, що телефон проходить перевірку. Якщо ні, то повертається повідомлення із попередженням,
         що номер введений не коректно"""
        # Зміна стану на повідомлені з номером телефона
        async with state.proxy() as data:
            data['cell_number'] = message.text
        # Зберігаємо бот для подальшої роботи з ним
        await state.finish()
        # Додаткова перевірка на відсутність телефону в БД
        if not db.customer_exists(message.from_user.id):
            # Виклик функції для валідації номера телефона за допомогою регулярного виразу
            # Якщо телефон проходе валідацію, номер успішно додається в БД
            if validate_number(data["cell_number"]):
                db.add_customer(message.from_user.id, data.get("cell_number"))
                await message.answer("Ваш номер записано. Гарного дня!")
            # У випадку невдалої перевірки номер не буде внесено до бази
            else:
                await message.answer("Телефон введено неправильно")
```

Рисунок 3.34 – Код реалізації

Перевірка правильності вводу номеру телефону реалізована за допомогою регулярного виразу (рис. 3.35).

```
    # Імпорт необхідних бібліотек
    import re
    # Функція, перевірки валідності номера за допомогою регулярного виразу
    def validate_number(cell_number) -> bool:
        match = re.fullmatch(r"^[\+]?3?[\s]?8?[\s]?\(?0\d{2}?\)?[\s]?\d{3}[\s|-]?\d{2}[\s|-]?\d{2}$", cell_number)
        return bool(match)
```

Рисунок 3.35 – Код для перевірки правильності вводу

Регулярний вираз перевіряє номер телефону, та якщо введено не правильно, телеграм бот видає повідомлення (рис. 3.36).

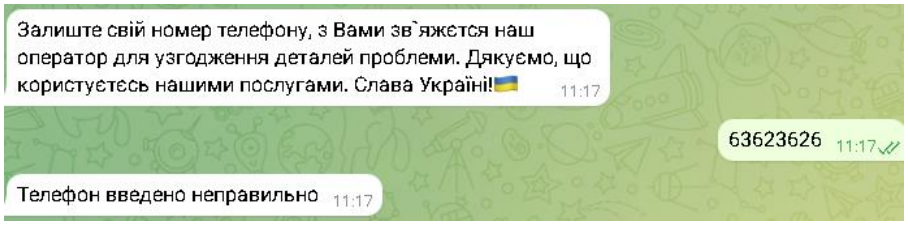

Рисунок 3.36 – Повідомлення бота про неправильність номеру

Третій розділ містить опис процесу створення сайту та телеграм боту, сторінок які зображені на скриншотах, та ділянки коду з реалізацією функцій телеграм боту. Порівняння текстових редакторів для редагування коду та описані і порівняні засоби розробок сайтів.

ВИСНОВКИ

Під час виконання дипломної роботи виконано наступні завдання:

– проаналізувано вебсайти інтернет провайдерів;
– проаналізовано сучасні технології у створенні вебсайтів та досліджено засоби розробки;
– розроблено проєкт сайту інтернет провайдера GreeNet;
– розроблено сайт інтернет провайдера GreeNet;
– розроблено телеграм-бот для зворотнього зв’язку з абонентами.

Розділи розробленого вебсайту: Головна, Про нас, Тарифи, Послуги, Гроза, Документи, Техпідтримка.

Створений програмний продукт задовольняє всім вимогам, що були зазначені на етапі постановки завдання. Отже, в ході виконання дипломної роботи було продемонстровано практичні навички створення та розробки вебсайтів та створено сайт з телеграм ботом.

ПЕРЕЛІК ПОСИЛАНЬ

1. Нильсен Я., Лоранжер Х. Web-дизайн: зручність використання Web-сайтів. 2015. Р. 322–329. URL: https://library.bsuir.by/m/12_101945_1_151704.pdf (дата звернення: 14.04.2023).

2. Компанєєтс М. О. Принципи проектування ефективних веб-сайтів. 2015. 109 с. URL: http://molodyvcheny.in.ua/files/journal/2015/9/66.pdf (дата звернення: 12.04.2023).

3. Фельке-Моррис Т. Велика книга веб-дизайна Террі Фельке-Моррис. Єксмо, 2012. 608 с.

4. Кравчук О. М. Розробка телеграм ботів на Python. URL: https://tinyurl.com/2p8v848y (дата звернення: 09.04.2023).

5. TELEGRAM. Telegram Bot API. URL: https://core.telegram.org/bots/api (дата звернення: 11.04.2023).

6. Guido van Rossum and the Python development team. Python Tutorial. January 12, 2020. URL: https://tinyurl.com/3hdud8nr (дата звернення: 19.04.2023).

7. Jubilee Enterprise. Introduction to HTML and CSS. 2016, Kelompok, Gramedia, Anggota IKAPI, Jakarta, 2016. 163 p. URL: https://tinyurl.com/ynhzet99 (дата звернення: 03.04.2023).

8. Nixon R. Learning PHP, MySQL & JavaScript with jQuery, CSS & HTML5. 2015. URL: https://tinyurl.com/3kj8s97h (дата звернення: 03.03.2023).

9. Уолтер А. Ємоційний веб-дизайн. Манн, Іванов і Фербер, 2012. 144 с.

10. Daker J. HTML and CSS: Design and Build Websites First Edition. Wrox, 2011. 490 p.

11. Janda M. Burn Your Portfolio: Stuff they don’t teach you in design school, but should. New Riders, 2016. 384 p.

12. Daber J. Html, Css, Website development and design. Exemo, 2022. 480 p.

13. Al Sweigart. The Recursive Book of Recursion: Ace the Coding Interview with Python and JavaScript. No Starch Press, 2022. 328 p.

14. Ernesti J., Kaiser P. Python: The comprehensive Guide. Rheinwerk Computing, 2022. 1078 p.

15. Abdel A.A. Build Your First Telegram Bot: A Step by Step Guide. 2018. URL:https://www.toptal.com/python/telegram-bot-tutorial-python (дата звернення: 10.03.2023).