РІВНЕНСЬКИЙ ДЕРЖАВНИЙ ГУМАНІТАРНИЙ УНІВЕРСИТЕТ

Факультет математики та інформатики

Кафедра інформаційно-комунікаційних технологій та
методики викладання інформатики

«До захисту допущено»
Завідувач кафедри
_____ проф. Войтович І.С.

«____» _____ 2021р.

протокол №_____

КВАЛІФІКАЦІЙНА РОБОТА

РОЗРОБКА ЕЛЕКТРОННОГО НАВЧАЛЬНОГО РЕСУРСУ З РОЗДІЛУ “СТРУКТУРИ ДАНИХ ТА АЛГОРИТМИ ЇХ ОБРОБКИ” ДИСЦИПЛІНИ «ТЕОРЕТИЧНІ ОСНОВИ ПРОГРАМУВАННЯ»

здобувача другого (магістерського) рівня вищої освіти
спеціальності 014.09 Середня освіта (Інформатика)

Вознюка Олега Юрійовича _____

Керівник:_____ Бабич С.М., доцент кафедри інформаційно-комунікаційних технологій та методики викладання інформатики, канд. тех. наук

Рецензент: _____ Максимцев Ю.Р., доцент кафедри фізики, астрономії та методики викладання Рівненського державного гуманітарного університету, канд. фіз.-мат. наук

Рецензент: _____ Сінчук А.М., доцент кафедри інформатики та прикладної математики Рівненського державного гуманітарного університету, канд. техн. наук

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.

Студент_____

Рівне – 2021 року

АНОТАЦІЯ

Вознюк О.Ю. Розробка електронного навчального ресурсу з розділу “Структури даних та алгоритми їх обробки” дисципліни «Теоретичні основи програмування». – На правах рукопису.

Магістерська робота на здобуття освітнього ступеня магістра за спеціальністю 014 Середня освіта (Інформатика). – Рівненський державний гуманітарний університет, Рівне, 2021.

Магістерська робота присвячена розробці інтерактивного навчального ресурсу з розділу «Структури даних та алгоритми їх обробки» дисципліни «Теоретичні основи програмування». У ході виконання роботи було здійснено пошук інформації відповідно до тематики проекту, проведено її аналіз та дослідження. Запропоновано тип і структуру навчального електронного ресурсу. Здійснено огляд сучасних програмних засобів для реалізації поставленого завдання. Для власної розробки обрано інтерпретовану об'єктно-орієнтовану мову програмування високого рівня зі строгою динамічною типізацією Python, інтегроване середовище розробки PyCharm, систему управління базами даних SQLite, безкоштовний веб-фреймворк Django. Створено проект електронного навчального ресурсу у вигляді електронного навчального контенту, наповнено його навчальними матеріалами. Здійснено тестування створеного програмного продукту.

Ключові слова: онлайн-навчання, інформаційна система, дистанційне навчання, інтерактивний навчальний ресурс, електронний навчальний контент.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

СКБД — Система керування базами даних

ПЗ — Програмне забезпечення

ОС — Операційна система

HTML — Гіпертекстова мова розмітки

CSS — Каскадні таблиці стилів

ВСТУП

У сучасному світі питання автоматизації монотонних, рутинних процесів діяльності людини постає все більш гостро. Коріння автоматизації проникає в усі сфери людського життя, починаючи з побутових необхідностей, таких як закупівля продуктів харчування або їх приготування, і закінчуючи сферою освіти.

У сфері освіти кожного дня спостерігаються все більш стрімкі тенденції на використання засобів автоматизації, таких як електронні журнали, що допомагають автоматизувати підрахунок оцінок, або системи подання знань.

Саме з цією тематикою і пов’язана тема даної роботи.

Виходячи з усього вищесказаного, можна зробити висновок про високу актуальність даної розробки.

Об’єктом дослідження є електронний навчальний ресурс як важливий інструмент навчально-виховного процесу.

Предметом дослідження є процес та програмні засоби розробки електронних навчальних ресурсів.

Мета роботи: створення електронного навчального ресурсу з розділу «Структури даних та алгоритми їх обробки» дисципліни «Теоретичні основи програмування» та його наповнення навчальним контентом.

Відповідно до предмету та мети дослідження визначено наступні **завдання**:

- здійснити пошук інформації відповідно до тематики магістерської роботи;
- проаналізувати теоретичні підходи та стан наукової розробленості питання розробки електронних навчальних ресурсів;
- розробити модель електронного навчального ресурсу;
- здійснити огляд сучасних програмних засобів для реалізації практичної частини роботи;

- з’ясувати особливості використання програмно-апаратних засобів електронних навчальних ресурсів;
- розробити проект електронного навчального ресурсу з розділу “Структури даних та алгоритми їх обробки” дисципліни «Теоретичні основи програмування»;
- наповнити створений ресурс навчальними матеріалами.

Методи дослідження: теоретичний аналіз наукової літератури, порівняння, систематизація, класифікація дали змогу дослідити і узагальнити матеріали з питань розробки електронних навчальних ресурсів, розробити хід дослідження;

Практичне значення полягає у розробці та впровадженні технології електронних навчальних ресурсів; рекомендаціях щодо застосування розробленої технології електронних навчальних ресурсів.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Інформаційна система

Інформаційна система (ІС) — це формальна, соціально-технічна та організаційна система, призначена для збору, обробки, зберігання та поширення інформації [1]. З точки зору соціальних технологій, інформаційна система складається з чотирьох компонентів: завдань, персоналу, структури (або ролей) і технології [2]. Інформаційну систему можна визначити як інтеграцію компонентів, які збирають, зберігають та обробляють дані. Вони використовуються для надання інформації, просування знань та цифрових продуктів [3].

Комп’ютерна інформаційна система – це система людей і комп’ютерів, які обробляють або інтерпретують інформацію [4-7]. Цей термін іноді використовується для простого позначення комп'ютерної системи з встановленим програмним забезпеченням.

Інформаційна система — це академічне вивчення системи, конкретної довідкової інформації та додаткової мережі апаратного та програмного забезпечення, що використовується людьми та організаціями для збору, фільтрації, обробки, створення та поширення даних. У центрі уваги — інформаційні системи з певними межами, користувачі, процесори, бібліотеки зберігання, входи, виходи та вищезгадані комунікаційні мережі [8].

У багатьох організаціях відділ або підрозділ, відповідальний за інформаційні системи та обробку даних, називають «інформаційними службами» [9-12].

Будь-яка конкретна інформаційна система призначена для підтримки операцій, управління та прийняття рішень [13-14]. Інформаційна система — це інформаційно-комунікаційні технології (ІКТ), які використовуються

організацією, і спосіб взаємодії людей з цією технологією для підтримки бізнес-процесів [15].

Деякі автори чітко розрізняють інформаційні системи, комп’ютерні системи та бізнес-процеси. Інформаційні системи зазвичай включають компонент ІКТ, але вони не лише пов’язані з ІКТ, а й зосереджені на кінцевому використанні інформаційних технологій. Інформаційні системи також відрізняються від бізнес-процесів, вони допомагають контролювати ефективність бізнес-процесів [16].

Альтер [17-18] продемонстрував переваги трактування інформаційних систем як особливого типу робочої системи. Робоча система — це система, в якій люди або машини використовують ресурси для виконання процесів і операцій для виробництва конкретних продуктів або послуг для клієнтів. Інформаційна система — це робоча система, діяльність якої призначена для збору, передачі, зберігання, пошуку, обробки та відображення інформації [19].

Тому інформаційні системи взаємопов’язані з системами даних, з одного боку, і бізнес-системами з іншого [20]. Інформаційна система — це форма комунікаційної системи, в якій дані подаються та обробляються як форма соціальної пам'яті. Інформаційні системи також можна розглядати як напівформальну мову, яка підтримує прийняття людських рішень і дій.

Інформаційна система є основним напрямом дослідження організаційної інформатики [21].

Надаються два погляди на інтелектуальну власність, включаючи програмне забезпечення, апаратне забезпечення, дані, персонал і процедури [22]. Чжен представив різні системні уявлення про інформаційну систему, а також додав процес і ключові елементи системи, такі як середовище, межі, ціль та взаємодія.

Комп’ютерна асоціація визначає «експерти інформаційних систем орієнтуються на інтеграцію інформаційно-технологічних рішень та бізнес-процесів для задоволення інформаційних потреб підприємств» [23].

Існують різні типи інформаційних систем, наприклад такі як: системи обробки транзакцій, системи підтримки прийняття рішень, системи управління знаннями, системи управління навчанням, системи управління базами даних та офісні інформаційні системи. Ключем до більшості інформаційних систем є інформаційні технології, які зазвичай призначені для того, щоб люди могли виконувати завдання, які не в змозі виконати люди, наприклад, обробляти великі обсяги інформації, виконувати складні обчислення та керувати багатьма одночасними процесами.

Інформаційні технології – це дуже важливий і гнучкий ресурс, яким можуть користуватися менеджери [24]. Багато компаній створили посаду головного інформаційного директора (CIO), який сидить у виконавчому комітеті (головний технологічний директор) разом із головним виконавчим директором (CEO), фінансовим директором (CFO), головним операційним директором (COO) та головний технолог). CTO також може виконувати роль CIO і навпаки. Головний директор з інформаційної безпеки (CISO) зосереджується на управлінні інформаційною безпекою.

Шість компонентів, які повинні об’єднатися для створення інформаційної системи:

- термін «технічні засоби» відноситься до машин та обладнання. Сучасна інформаційна система включатиме сам комп’ютер та всі його аксесуари. До допоміжного устаткування відноситься обладнання введення і виведення, обладнання зберігання та обладнання зв'язку. У комп’ютерній інформаційній системі апаратне забезпечення може включати книги та чорнило;
- програмне забезпечення: термін програмне забезпечення відноситься до комп’ютерних програм і посібників (якщо є), які їх підтримують. Комп’ютерне програмне забезпечення – Комп’ютерно-зчитувана інструкція, яка використовується для керування апаратними схемами в межах функцій системи з метою отримання корисної інформації з даних.

Зазвичай програма зберігається на певному носії введення-виводу, як правило, на диску або стрічці. «Програмне забезпечення» комп'ютерної інформаційної системи включає, як підготувати обладнання до використання (наприклад, заголовки стовпців у книзі книги) та інструкції щодо їх використання (довідник для каталогу карток);

- дані: дані – це факт, який система використовує для отримання корисної інформації. У сучасних інформаційних системах дані зазвичай зберігаються на диску або стрічці у формі, що читається комп'ютером, доки вони не знадобляться комп'ютеру. У колишніх комп'ютерних інформаційних системах дані зазвичай зберігалися в доступній для читання формі;
- процедура: Процедура – це політика функціонування інформаційної системи управління. «Програми для людей, програмне забезпечення для апаратних засобів» — це поширена метафора, яка використовується для ілюстрації ролі програм у системі;
- люди: кожній системі потрібні люди, щоб функціонувати. Зазвичай елементом системи, який найбільше не помічають, є особа, яка може бути компонентом, який має найбільший вплив на успіх або невдачу інформаційної системи. До них належать «не тільки користувачі, а й ті, хто керує та обслуговує комп'ютери, ті, хто підтримує дані, і ті, хто підтримує комп'ютерні мережі» [25];
- зворотній зв'язок: це інший компонент ІС, який визначає, що ІС може надавати зворотній зв'язок (Хоча цей компонент не потрібний для функціонування);

1.2 Онлайн-навчання

Освітня технологія (зазвичай скорочено EduTech або EdTech) — це поєднання комп'ютерного обладнання, програмного забезпечення та освітньої теорії та практики для сприяння навчанню. Якщо згадати його абревіатуру EdTech, то зазвичай це стосується галузі, в якій знаходиться компанія, що створює освітні технології [23].

Окрім реального освітнього досвіду, освітні технології базуються також на теоретичних знаннях з різних дисциплін, таких як: комунікація, педагогіка, психологія, соціологія, штучний інтелект, інформатика [4]. Він охоплює декілька областей, включаючи теорію навчання з використанням мобільних технологій, комп'ютерне навчання, онлайн-навчання та мобільне навчання.

Асоціація освітніх комунікацій і технологій (AECT) визначає освітні технології як «дослідницькі та етичні практики, які сприяють навчанню та покращують ефективність шляхом створення, використання та управління пов'язаними технічними процесами та ресурсами» [5]. Він описав технологію навчання як «теорію та практику проектування, розробки, використання, управління та оцінювання навчальних процесів і ресурсів» [6-8]. Отже, освітня технологія відноситься до всіх ефективних і надійних освітніх прикладних наук, таких як обладнання, а також процеси і процедури, отримані в результаті досліджень. У цьому випадку можуть бути задіяні теорії, алгоритми або евристичні процеси: Це не обов'язково означає фізичні технології. Освітня технологія – це процес активної інтеграції технологій в освіту, що сприяє формуванню більш різноманітного навчального середовища та методів та навчає студентів використовувати технології та досягати спільних цілей.

Тому опис інтелектуального та технологічного розвитку освітніх технологій має кілька різних аспектів:

- освітні технології як теорія та практика освітніх підходів до навчання;
- освітні технології як технічні засоби та засоби масової інформації, такі як масштабні онлайн-курси, які сприяють передачі знань, розвитку та

обміну. Зазвичай це те, що люди використовують, коли використовують термін «EdTech»;

- освітні технології для систем управління навчанням (LMS), такі як інструменти керування студентами та курсами та інформаційні системи управління освітою (EMIS);
- освітні технології, такі як управління бек-офісами, такі як системи управління навчанням для логістики та управління бюджетом, а також сховище записів навчання (LRS) для зберігання й аналізу навчальних даних;
- освітні технології – це сама по собі тема, такі курси можна назвати «комп'ютерні дослідження» або «інформаційно-комунікаційні технології (ІКТ)» [9].

Способи допомоги людям і дітям навчатися простіше, швидше, точніше можна простежити ще до появи дуже ранніх інструментів, таких як малюнки на стінах печер [26-27]. Письмові дошки та різні види рахівниць використовували щонайменше тисячу років [28]. З моменту запуску книги та брошури відіграють важливу роль у освіті. З початку 20 століття фотокопіювальні апарати (наприклад, мімеографи та шаблони Gestener) використовувалися для виготовлення невеликих партій копій (зазвичай 10-50 копій) для використання в класі або вдома. Використання медіа в освітніх цілях загалом можна простежити до першого десятиліття 20-го століття [29], коли були представлені навчальні фільми (1900-ті роки) та механічні тренажери Sidney Press (1920-ті). Першою широкомасштабною оцінкою з кількома варіантами була Army Alpha, яка використовувалася для оцінки інтелекту, а точніше, для оцінки тенденцій солдатів Першої світової війни. Широко використовувана технологія потім використовувалася для навчання солдатів з використанням фільмів та інших медіаматеріалів (наприклад, проекторів) під час і після Другої світової війни. Концепцію гіпертексту можна простежити до мемексу Ваннью Буша в 1945 році.

У 1950-х роках в навчальних закладах широко використовувалися діапроектори. Ковальське вудилище було розроблено в 1920-х роках і широко використовується з кінця 1950-х років.

У середині 1960-х професори психології Стенфордського університету Патрік Саппс і Річард К. Аткінсон намагалися використовувати комп'ютери для навчання арифметиці та правопису учнів початкової школи в Об'єднаному шкільному окрузі Пало-Альто, Каліфорнія, за допомогою телетайпів [30-31]. Освітня програма Стенфордського університету для обдарованої молоді виникла з тих ранніх експериментів.

Онлайн-освіта виникла в Університеті Іллінойсу в 1960 році. Хоча Інтернет зародився ще через дев'ять років, учні могли отримати доступ до інформації класу через підключені комп'ютерні термінали. Перший онлайн-курс був наданий Університетською електронною мережею в 1986 році для комп'ютерів DOS і Commodore 64. Комп'ютерне навчання нарешті забезпечило перший онлайн-курс із взаємодією з реальним світом. У 2002 році MIT запустив безкоштовний онлайн-курс. Станом на 2009 рік приблизно 5,5 мільйонів студентів пройшли принаймні один онлайн-курс. Наразі третина студентів коледжу вивчає принаймні один онлайн-курс під час навчання в коледжі. В університеті DeVree 80% студентів отримують дві третини онлайн-попиту. Також у 2014 році з 5,8 мільйона студентів, які пройшли онлайн-курси, 2,85 мільйона студентів брали участь у всіх онлайн-курсах.

З цієї інформації можна зробити висновок, що кількість студентів, які відвідують заняття в Інтернеті, постійно зростає [32-33].

У 1971 році Іван Ілліч (Ivan Ilyich) опублікував надзвичайно впливову книгу «Асоціація знищення освіти», в якій уявляв «освітню мережу» як зразок для людей, щоб поділитися необхідним навчанням. У 1970-х і 1980-х роках Мюррей Турофф і Старр Роксана Хілц з Технологічного інституту Нью-Джерсі [34] та досягнення Університету Гвельфа в Канаді [35] зробили значний внесок у комп'ютерне навчання. У Великобританії Рада з освітніх технологій підтримує використання освітніх технологій, особливо через

управління державною Національною програмою розвитку навчання з використанням комп'ютера [36] (1973-77) та Програмою освіти з мікроелектроніки (1980-86).

До середини 1980-х років багато університетських бібліотек мали доступ до змісту курсу. У комп'ютерному навчанні (CBT) або комп'ютерному навчанні (CBL) навчальна взаємодія відбувається між учнем і комп'ютерним навчанням або моделюванням мікросвіту.

Цифрові комунікації та мережі в освіті зародилися в середині 1980-х років. Школи почали використовувати нові інструменти для проведення дистанційних курсів та використовувати комп'ютерні мережі для отримання інформації. Ранні системи електронного навчання, засновані на комп'ютерному навчанні, часто копіювали авторитарні стилі навчання, що вказує на те, що роль систем електронного навчання полягала в передачі знань, а не більш пізніх систем, заснованих на спільному комп'ютерному навчанні (CSCL), що заохочує розвиток знань.

Відеоконференції є важливим піонером освітніх технологій, відомих сьогодні. Особливою популярністю ця робота користується в музейній освіті. Навіть за останні роки популярність відеоконференцій зросла до понад 20 000 студентів у Сполучених Штатах і Канаді в період з 2008 по 2009 рік.

Недоліки цієї форми освітніх технологій очевидні: якість зображення та звуку часто зерниста або піксельна, для відеоконференцій потрібна міні-телевізійна студія в музеї для трансляції, а простір стає проблемою. І для провайдерів, і для учасників потрібне спеціальне обладнання [37].

Відкритий університет Сполученого Королівства [35] та Університет Британської Колумбії (де Web CT спочатку була розроблена, а тепер є частиною Blackboard Inc.) зробили революцію у використанні Інтернет-навчання [38], широко використовуючи веб-навчання, дистанційне навчання, та міжстудентська онлайн-дискусія [39]. Харасим (1995) [40] та інші практики надають великого значення використанню освітніх мереж.

З появою у 1990-х роках всесвітньої павутини вчителі почали використовувати нові технології для використання багатооб'єктно-орієнтованих сайтів, а саме текстових мережевих систем віртуальної реальності, для створення веб-сайтів курсів разом із простими наборами інструкцій для їх студентів.

У 1994 році була створена перша онлайн-школа. У 1997 році Gradiadei описав критерії оцінки продуктів і розробки курсів, заснованих на технології, які включали портативність, відтворюваність, масштабованість, зручність використання та високу можливість довгострокової економічної ефективності [41].

Покращення функцій Інтернету дозволило створити нові рішення для спілкування за допомогою мультимедійних або веб-камер. За оцінками Національного центру статистики освіти, у період з 2002 по 2005 рік кількість студентів K-12, які навчалися на дистанційних курсах в Інтернеті, зросла на 65%, завдяки більшій гнучкості, простоті спілкування вчителя та студента та швидкому зворотному зв'язку.

Згідно з дослідженням 2008 року, проведеним Міністерством освіти США, протягом 2006-2007 навчального року приблизно 66% державних і приватних шкіл брали участь у програмі фінансової допомоги студентам. Пропонуються деякі курси дистанційного навчання; записи показують, що 77% студентів пройшли кредитні курси для онлайн-курсів. У 2008 році Європейська комісія прийняла заяву, яка підтвердила потенціал електронного навчання для сприяння рівності та покращення освіти в ЄС [42].

Комп'ютерно-опосередковане спілкування (КМК) — це опосередковані комп'ютером стосунки між учнями та викладачами. На відміну від цього, CBT/CBL зазвичай означає персоналізоване (самоосвітнє) навчання, тоді як CMC передбачає допомогу викладачів і вимагає гнучких сценаріїв навчальної діяльності. Крім того, сучасні ІКТ надають освітні інструменти для підтримки навчальних спільнот та пов'язаних завдань управління знаннями.

Студенти, які виросли в цю цифрову епоху, широко знайомляться з різними медіа [43-44]. Великі високотехнологічні компанії фінансують школи, щоб вони могли навчати студентів технологіям [45].

2015 був першим роком, коли приватні некомерційні організації набирали більше онлайн-студентів, ніж некомерційні організації, хоча державні університети все ще набирали більшість онлайн-студентів. Восени 2015 року понад 6 мільйонів студентів були зареєстрованими хоча б на один онлайн-курс [46].

У 2020 році через пандемію COVID-19 багато шкіл у всьому світі були змушені закритися на карантин, що призвело до того, що все більше учнів початкових класів брали участь у дистанційному навчанні, а студенти коледжів та вузів брали участь в онлайн-курсах для дистанційного навчання [47]. Такі організації, як ЮНЕСКО, запропонували освітні технологічні рішення, щоб допомогти школам просувати дистанційну освіту [48]. Розширення карантину через пандемію та зосередження на дистанційному навчанні залучили рекордний капітал для електронної промисловості. У 2020 році тільки в Сполучених Штатах стартапи ed-tech залучили 1,78 мільярда доларів США, охопивши 265 угод, проти 1,32 мільярда доларів у 2019 році [49].

1.3 Огляд існуючих рішень

Найпопулярнішими рішеннями в обраній області на сучасний день можна назвати різного роду ресурси для онлайн-навчання.

Основним прикладом реалізації подібного концепту можна вважати GeekBrains. Дана платформа позиціонує себе як «освітній портал» та надає онлайн-курси на різні теми.

Головна сторінка ресурсу зображена на рисунку 1.1.

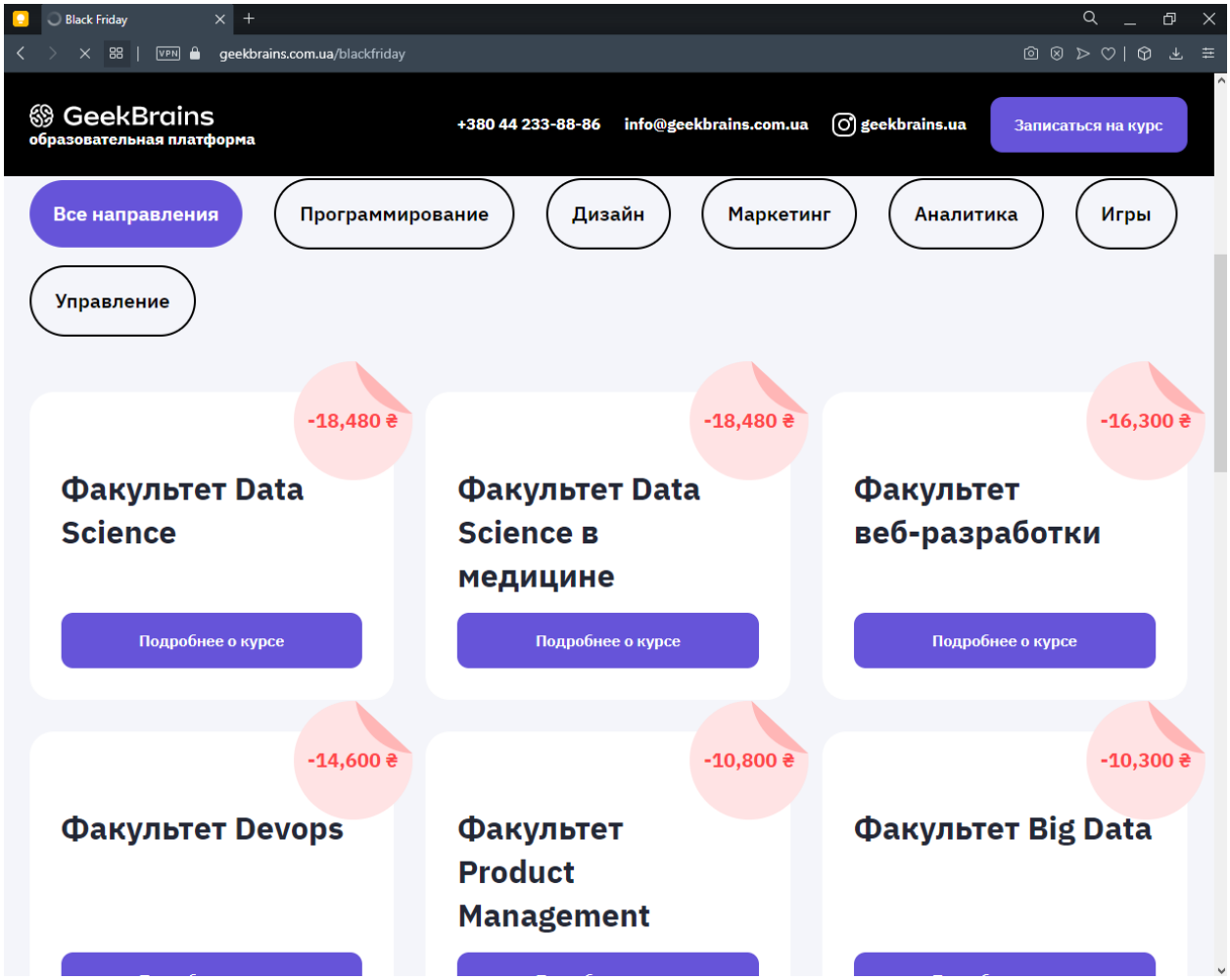


Рисунок 1.1 — Головна сторінка GeekBrains

Окрім даного порталу існують і інші, наприклад udemy (рис. 1.2) або skillfactory (рис. 1.3).

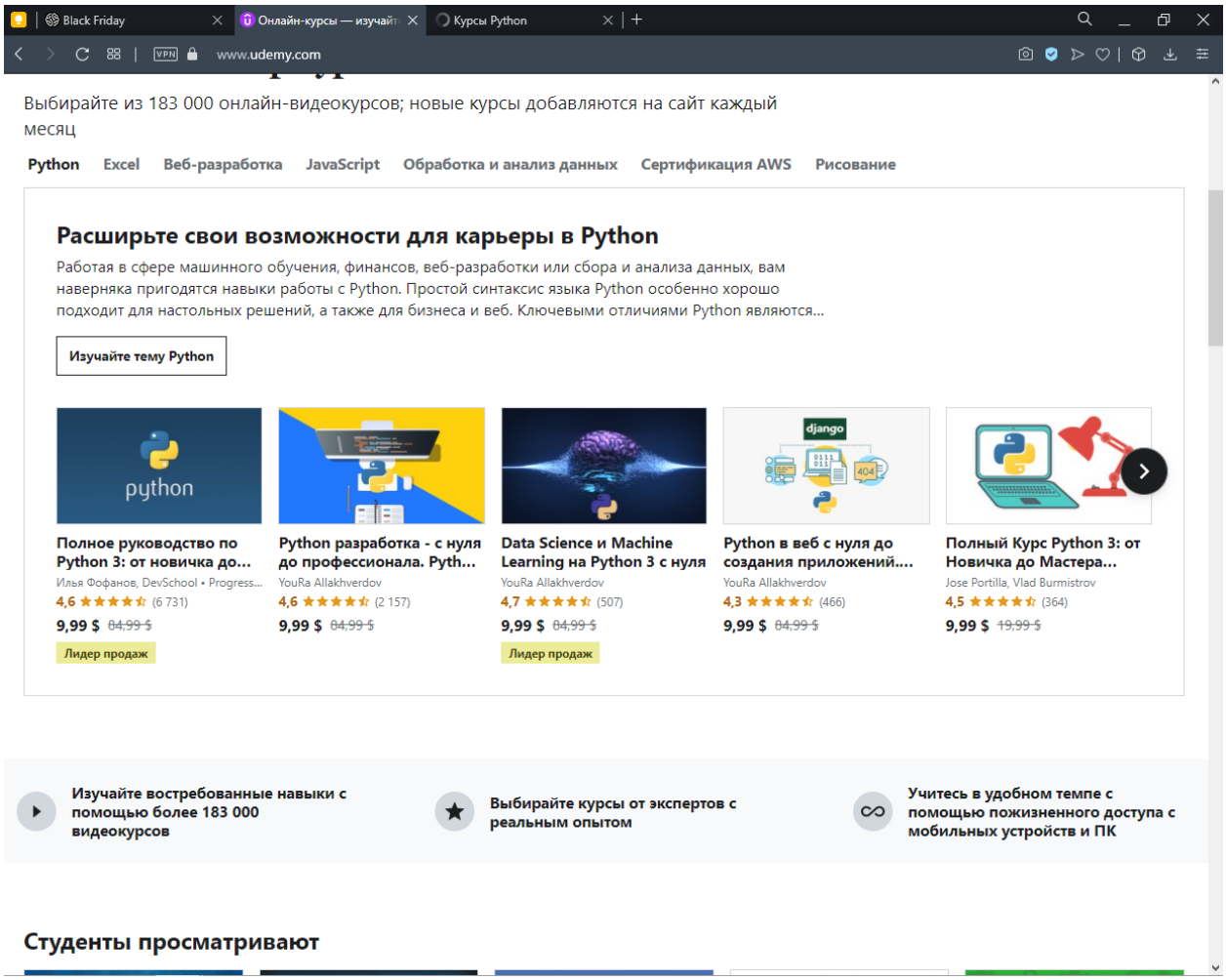


Рисунок 1.2 — Головна сторінка Udemy

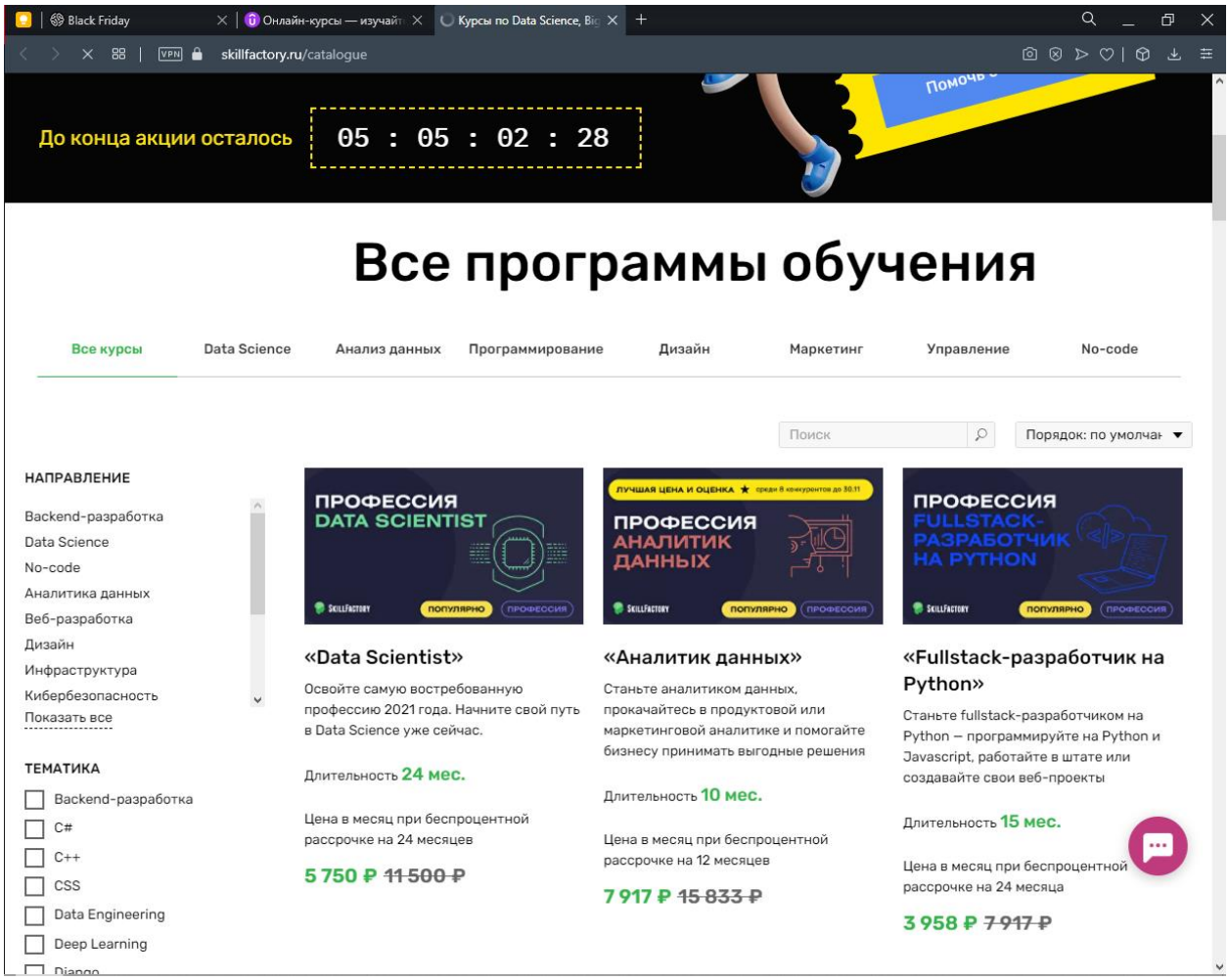


Рисунок 1.3 — Головна сторінка SkillFactory

Усі портали мають велику кількість переваг, таких як велика кількість доступних курсів, проте усі вони мають один спільний недолік: велика розробленість і відсутність структурованості курсів, увесь матеріал розбито на менші підкурси, а більшість матеріалу не має практичної цінності.

1.4 Постановка задачі

Основна задача роботи полягає в розробленні електронного навчального ресурсу з розділу «Структури даних та алгоритми їх обробки» дисципліни «Теоретичні основи програмування».

Система буде представляти собою типовий набір онлайн курсів, за виключенням виправлення основного недоліку усіх курсів — неструктурованості матеріалу.

Система буде представляти собою набір лекцій, до кожної з якої є спеціальне тестування отриманих знань.

При необхідності тестування можна пройти повторно, після виправлення помилок і повторення матеріалу лекцій.

Користувачі мають можливість зареєструватися і авторизуватися в системі за своїм логіном і паролем.

Доступ для незареєстрованих користувачів заборонено.

РОЗДІЛ 2
ОГЛЯД ІНСТРУМЕНТАРІЮ РОЗРОБКИ

2.1 Огляд мови програмування

Python — це інтерпретована мова програмування високого рівня. Філософія дизайну Python підкреслює читабельність коду за допомогою значного відступу. Його мовна структура та об’єктно-орієнтований підхід розроблені, щоб допомогти програмістам писати чіткі логічні коди для малих і великих проектів.

Python динамічно збирає сміття. Він підтримує декілька парадигм програмування, включаючи структурне, об’єктно-орієнтоване та функціональне програмування. Через повну стандартну бібліотеку Python часто описують як мову, що включає батареї.

Як наступник мови програмування ABC, Гвідо ван Россум почав вивчати Python наприкінці 1980-х і вперше був випущений в 1991 році під назвою Python 0.9.0 [32]. Python 2.0 був випущений у 2000 році і представив нові функції, такі як розуміння списку та систему збору сміття з використанням підрахунку посилань. Його буде припинено у версії 2.7.18 у 2020 році [33]. Python 3.0 був випущений у 2008 році. Це основна версія мови, яка не повністю сумісна з протилежною, і більшість коду Python 2 не може залишатися незмінною на Python 3.

Python завжди був однією з найпопулярніших мов програмування.

Python був задуманий наприкінці 1980-х років Гвідо ван Россумом з Centrum Wiskunde & Informatica (CWI) в Нідерландах, як наступник мови програмування ABC, натхненної SETC, здатної обробляти винятки та взаємодіяти з операційною системою Amoeba. Його реалізація почалася в грудні 1989 року. Ван Россум був повністю відповідальним за проект як провідний розробник до 12 липня 2018 року, коли він оголосив про свою «постійну відпустку» і отримав титул «Диктатора, дружнього до життя»

Python. Він відображав свою довгу історію через спільноту Python. Довгострокове зобов’язання працювати в якості головного менеджера проекту. Тепер він поділяє своє лідерство як член наглядової ради з п’яти осіб. У січні 2019 року активні розробники ядра Python обрали Бретта Кеннона, Ніка Коглана, Баррі Варшава, Керол Віллінг і Ван Россума до ради директорів із п’яти осіб. Відтоді Гвідо ван Россум зняв свою кандидатуру в раду директорів 2020 року.

Python 2.0 був випущений 16 жовтня 2000 року з багатьма важливими новими функціями, включаючи збірник сміття для виявлення циклу та підтримкою Unicode.

Python 3.0 був випущений 3 грудня 2008 року. Це серйозна редакція мови і не повністю сумісна з іншою стороною. Багато з його основних функцій було перенесено до Python версій 2.6.x і 2.7.x. Реліз Python 3 містить утиліту 2 to 3, яка автоматично (принаймні частково) перетворює код Python 2 на Python 3.

Термін дії Python 2.7 спочатку був встановлений у 2015 році, а потім перенесений на 2020 рік через побоювання, що велика кількість існуючого коду може бути нелегкою в перенесені на Python 3. Він не випускатиме більше виправлень безпеки та інших покращень. Наприкінці життєвого циклу підтримується лише Python 3.6. та Python новіших версій.

Python 3.9.2 і 3.8.8 прискорені, оскільки всі версії Python мають проблеми з безпекою, що призводить до можливого віддаленого виконання коду та отруєння веб-кешу.

Python — це мова програмування з кількома парадигмами. Він повністю підтримує об’єктно-орієнтоване програмування та структурне програмування, а багато його функцій підтримують функціональне програмування та аспектно-орієнтоване програмування (включаючи метапрограмування та метаоб’єктний (магічний метод)). Розширення підтримує багато інших парадигм, включаючи дизайн контрактів і логічне програмування.

Python використовує комбінацію динамічного введення тексту та підрахунку посилань і збирач сміття, який визначає цикли для керування пам’яттю. Він також має динамічне розділення імен (пізнє прив’язування) для зв’язування імен методів і змінних під час виконання програми.

Python розроблено для забезпечення певної підтримки функціонального програмування в традиції Lisp. Він має функції фільтрації, відображення та скорочення; перераховує розуміння, словник, набір і вираз генератора. Стандартна бібліотека має два модулі (itertools і functools), які реалізують функціональні інструменти, запозичені з Haskell і Standard ML.

Дзен Python (PEP 20) підсумовує основні концепції мови, включаючи такі сентенції:

- красиве - краще, ніж потворне;
- явне краще, ніж неявне;
- просте - краще, ніж складне;
- складний - це краще, ніж складний;
- підрахунок читабельності.

Python не має всіх своїх функцій, вбудованих у його ядро, але розроблено, щоб бути дуже розширюваним (з модулями). Ця компактна модульність робить його особливо популярним як спосіб додавання програмованих інтерфейсів до існуючих програм. Ідея Ван Россума про невелику хост-мову, велику стандартну бібліотеку та легко розширюваний перекладач походить від його незадоволення ABC, який підтримує протилежний підхід. Python прагне до простішого та стислого синтаксису та граматики, надаючи розробникам можливість вибору методів кодування. На відміну від девізу Perl «Є більше одного способу зробити це», Python висвітлює філософію дизайну: «Повинен бути один – бажано лише один – очевидний спосіб зробити це» [69]. Алекс Мартеллі, член Python Software Foundation і автор книги про Python, написав: «У культурі Python опис чогось як «розумного» не вважається компліментом».

Розробники Python прагнуть уникнути передчасної оптимізації та відмовляються виправляти некритичні частини референсної реалізації CPython, які забезпечують невелике збільшення швидкості за ціною наочності. Коли швидкість важлива, програмісти Python можуть перемістити важливі за часом функції в розширення, написані на таких мовах, як C, або використовувати PyPy, компілятор «точно вчасно». Також доступний Cython, який перетворює скрипти Python на C і здійснює прямі виклики API рівня C до інтерпретатора Python.

Важлива мета для розробників Python — зробити його веселим. Це знайшло відображення в назві мови — данині британській комедійній групі «Монті Пайтон» — і іноді грайливому підході до підручників та довідкових матеріалів, таких спаму та яєць (із відомого етюду Монті Пайтона) стандартних foo and bar.

Поширеним новим словом у спільноті Python є python, яке може мати широкий спектр значень, пов’язаних зі стилем програмування. Сказати, що код є pythonic, означає, що він добре використовує ідіоми Python, що є природним, або що він демонструє вільну мову, яка відповідає мінімалістичній філософії Python і підкреслює читабельність. На відміну від цього, код, який важко зрозуміти або прочитати як приблизну копію іншої мови програмування, називається не Python.

Користувачів і шанувальників Python, особливо тих, кого вважають обізнаними або досвідченими, часто називають Pythonists. Python має бути легкою для читання мовою. Його формат візуально не захаращений, часто використовуються ключові слова англійською мовою, тоді як інші мови використовують розділові знаки. На відміну від багатьох інших мов, він не використовує дужки для розділення блоків і допускає крапку з комою після операторів, але рідко використовується, якщо взагалі використовується. У порівнянні з C або Pascal, він має менше граматичних винятків і особливих випадків.

Python використовує пробіли замість фігурних дужок або ключових слів для розділення блоків. Збільшення відступу відбувається після певних операторів; зменшення відступу означає кінець поточного блоку. Тому візуальна структура програми точно відображає семантичну структуру програми. Цю функцію іноді називають «зовнішніми» правилами, і в деяких інших мовах вона є, але в більшості мов відступ не має семантичного значення. Рекомендований розмір відступу – чотири пробіли.

Оператори Python включають (серед іншого):

- використовуйте знак рівності = оператор присвоєння;
- оператор if, умовно виконати блок коду, а також else та elif (скорочення від else-if);
- оператор for для перегляду ітераційного об’єкта шляхом захоплення кожного елемента в локальну змінну для використання вкладеним блоком;
- поки умова істинна, виконується оператор while блоку коду;
- оператор try, який дозволяє витягам, що містяться у вкладених блоках коду, захоплювати й обробляти вміст, відмінний від речень; він також гарантує, що незалежно від того, як виходить блок, чистий код у блоці в кінцевому підсумку буде виконано;
- оператор raise використовується для отримання вказаного винятку або повторного виклику перехопленого винятку;
- оператор класу, який виконує блок коду та приєднує свій локальний простір імен до класу для використання в об’єктно-орієнтованому програмуванні;
- оператор def, який визначає функцію або метод;
- оператор Python 2.5, випущений у вересні 2006 р. [82], охоплює блок коду в диспетчері контексту (наприклад, отримання блокування перед запуском блоку коду і зняття блокування після цього або відкриття файлу, а потім його закриття), що дозволяє

Поведінка, подібна до пошуку ресурсів, — це ініціалізація (RAII), яка замінює поширену ідіому try / finally;

- оператор break виходить з циклу;
- оператор continue пропускає цю ітерацію і переходить до наступного елемента;
- оператор del видаляє змінну, а це означає, що посилання з імені на значення видаляється, а спроби використати змінну призведуть до помилки. Видалену змінну можна повторно призначити;
- оператор проходу, виконує функцію NOP. Це синтаксично необхідно для створення порожнього блоку коду;
- оператори тверджень, які використовуються для перевірки умов, що застосовуються під час налагодження;
- оператор yield, який повертає значення з функції генератора. У Python 2.5 yield також є оператором. Ця форма використовується для реалізації спільного плану;
- оператор повернення, який використовується для повернення значення з функції;
- оператор імпорту, що використовується для імпорту модуля, чия функція або змінна може використовуватися в поточній програмі. Існує три способи імпорту: імпорт <ім’я модуля> [як <псевдонім>] або за допомогою <ім’я модуля> імпорт * або за допомогою <ім’я модуля> імпорт <визначення 1> [як <псевдонім 1>], <визначення 2> [як <псевдонім 2>];
- оператор присвоєння (=) діє шляхом прив'язки імені як посилання на окремий динамічно розподілений об'єкт. Потім змінні можуть бути відскочені в будь-який час до будь-якого об’єкта. У Python ім'я змінної є загальним власником посилання і не має фіксованого типу даних, пов’язаного з ним. Однак у даний момент часу змінна буде посилатися на якийсь об'єкт, який матиме тип. Це називається

динамічним набором тексту і протиставляється статично набраним мовам програмування, де кожна змінна може містити лише значення певного типу.

Python не підтримує оптимізацію хвостових викликів або першокласні розширення, і, за словами Гвідо ван Россума, ніколи не буде. Однак, розширивши генератор Python, у версії 2.5 надається краща підтримка функцій, подібних до корутина. До 2,5 генераторів є ледачими ітераторами, інформація передається від генераторів в одному напрямку. З Python 2.5 ви можете передавати інформацію назад до функції генератора, а з Python 3.3 ви можете передавати інформацію через кілька рівнів стека.

Деякі вирази Python подібні до виразів у таких мовах, як C і Java, а деякі ні:

- додавання, віднімання та множення однакові, але поведінка ділення різна. У Python є два типи розділів. Це поділ на поверхню (або ціле поділ) // і з плаваючою комою/діленням. Python також використовує оператори ** для просування;
- новий оператор @ infix був представлений у Python 3.5. Він призначений для використання такими бібліотеками, як NumPy для множення матриць;
- у Python 3.8 введено синтаксис: =, який називається «оператором моржа». Він призначає значення змінним як частину більшого виразу [91];
- у Python == порівнює з Java за значенням, Java порівнює значення за значенням і порівнює об'єкти за посиланням. (У Java ви можете використовувати метод equals для порівняння значень об’єктів ().) Оператор Python is можна використовувати для порівняння ідентифікаторів об’єктів (порівняння посилань). У Python порівняння можна ланцюжком, наприклад a <= b <= c;

- Python використовує слова та, або, не для своїх булевих операторів, а не для символічного &&, ||,! використовується в Java та C;
- Python має тип виразу, що називається розумінням списку, а також більш загальний вираз, який називається генераторським виразом;
- анонімні функції реалізовані за допомогою лямбда-виразів; однак вони обмежені тим, що тіло може бути лише одним виразом;
- умовні вирази в Python пишуться як x, якщо c else y (відрізняється за порядком операндів від оператора c? X: y, загального для багатьох інших мов);
- Python розрізняє списки та кортежи. Список записується як [1, 2, 3], змінюється і не може використовуватися як ключі словника (ключі словника повинні бути незмінними в Python). Кортеж записується як (1, 2, 3) і є незмінним, тому, поки всі елементи кортежу незмінні, його можна використовувати як ключ словника. Оператор + можна використовувати для об'єднання двох кортежів, що безпосередньо не змінює їх вміст, а створює новий кортеж, що містить два надані елементи кортежу. Тому, враховуючи, що змінна t спочатку дорівнює (1, 2, 3), коли виконується t = t + (4, 5), спочатку обчислюється t + (4, 5), щоб отримати (1, 2, 3 , 4, 5), а потім призначити його назад до t, тим самим фактично «змінюючи вміст t», відповідаючи інваріантній природі об'єкта кортежу. Дайте зрозуміти, що кортежі в контексті не потребують дужках;
- Python має послідовність розпакування, в якій кілька виразів, кожен з яких обчислює все, що можна призначити (змінні, атрибути запису тощо), пов'язуються в тій самій формі, що й літерал кортежу, і в цілому поміщають твердження твердження зліва Сторона середнього розміру. Оператор очікує повторюваний

об’єкт праворуч від знака рівності, який виводить таку ж кількість значень, як і вираз, наданий для запису під час сортування, і фільтрує його, і призначає кожне створене значення відповідному виразу зліва;

- у Python є оператор % «формат рядка». Це схоже на рядок printf в C, наприклад "спам =% s egg =% d"% ("blah", 2) оцінюється як "спам = blah-eggs = 2". У Python 3 і 2.6+ це доповнюється методом format() класу str, наприклад "спам = {0}яйця = {1}". Формат («Зачекайте», 2). Python 3.6 додав "f-рядки": blah = "blah"; egg = 2; f'spam = {blah} egg = {eggs}';
- рядки в Python можна об’єднати шляхом «додавання» (той самий оператор, що й додавання значень цілого чи плаваючої коми). Наприклад, "спам" + "яйце" повертає "спам". Навіть якщо ваш рядок містить числа, вони все одно додаються як рядки, а не цілі числа. Наприклад, "2" + "2" повертає "22";
- Python має різні типи рядкових літералів;
- рядки розділені одинарними або подвійними лапками. На відміну від оболонки Unix, мови Perl і Perl, які впливають на Perl, одинарні та подвійні лапки працюють однаково. Обидва типи рядків використовують зворотну косу риску (\) як вихідні символи. Інтерполяція рядка вже доступна як "відформатований рядковий літерал" у Python 3.6;
    - рядки з подвійними лапками, які починаються та закінчуються серією з трьох одинарних або подвійних лапок. Вони можуть охоплювати кілька рядків і функціонувати, як тут документи в оболонках, Perl і Ruby;
    - сирі різновиди рядків, що позначаються префіксом рядкового літералу перед r. Послідовності втечі не інтерпретуються; отже, необроблені рядки корисні там, де

буквенні зворотні слеші є загальними, такі як регулярні вирази та шляхи у стилі Windows. Порівняйте "@ -citation" у C #;

- Python має індекси масивів та вирази нарізування масивів у списках, що позначаються як [ключ], [старт: зупинка] або [старт: зупинка: крок]. Індекси базуються на нулі, а негативні індекси відносно кінця. Зрізи беруть елементи від початкового індексу до, але не включаючи, індексу зупинки. Третій параметр зрізу, який називається кроком або кроком, дозволяє пропускати та обертати елементи. Індекси зрізів можуть бути опущені, наприклад, [:] повертає копію всього списку. Кожен елемент зрізу є неглибокою копією.

Python чітко розрізняє вирази та вирази, що відрізняється від мов, таких як Common Lisp, Scheme або Ruby. Це може призвести до дублювання певних функцій.

Метод для об’єкта — це функція, приєднана до класу об’єкта; синтаксис instance.method (аргумент) — це синтаксичний цукор Class.method (екземпляр, аргумент) для поширених методів і функцій. Методи Python мають явний параметр self для доступу до даних екземпляра, що є протилежністю неявного self (або цього) в деяких інших об’єктно-орієнтованих мовах програмування (наприклад, C++, Java, Objective-C або Ruby).

Python використовує набір качок і має типізовані об’єкти, але нетиповані імена змінних. Обмеження типу не перевіряються під час компіляції; навпаки, операції над об’єктом можуть бути невдалими, а це означає, що об’єкт не має відповідного типу. Хоча Python динамічно типізується, він забороняє невизначені операції (наприклад, додавання чисел до рядка) і не намагається зрозуміти їх тихо.

Python дозволяє програмістам визначати власні типи, використовуючи класи, які найчастіше використовуються в об’єктно-орієнтованому програмуванні. Новий екземпляр класу створюється шляхом виклику класу

(наприклад, SpamClass() або EggsClass()).Клас — це екземпляр типу метакласу (сам екземпляр), який дозволяє виконувати метапрограмування та відображати.

До версії 3.0 Python мав два типи класів: старі та нові. Синтаксис двох стилів однаковий, різниця полягає в тому, чи успадковується об’єкт класу безпосередньо чи опосередковано (усі класи нового стилю успадковуються від об’єкта і є екземплярами типу). У Python 2, починаючи з Python 2.2, ви можете використовувати ці два типи класів. Класи старого стилю були ліквідовані в Python 3.0.

Довгостроковий план передбачає підтримку інкрементного введення, а в Python 3.5 синтаксис мови дозволяє вказувати статичні типи, але вони не перевіряються в реалізації CPython за замовчуванням. Експериментальна додаткова статична перевірка типу mypy підтримує перевірку типу під час компіляції.

CPython є еталонною реалізацією Python. Він написаний на C, відповідає стандарту C89 і має кілька вибраних функцій C99 (в більш пізніх версіях C він вважається застарілим; CPython включає власні розширення C, але сторонні розширення не обмежуються старими C версія, наприклад, може бути реалізована на C11 або C++). Він компілює програми Python у проміжні байт-коди, які потім виконуються його віртуальною машиною. CPython постачається з великою стандартною бібліотекою, яка написана сумішшю C і рідного Python. Він працює на багатьох платформах, включаючи Windows (починаючи з Python 3.9, інсталятор Python навмисно не зміг встановити в Windows 7 і 8; Windows XP підтримувалася до Python 3.5) і більшість сучасних Unix-подібних систем, включаючи macOS (і Apple M1). Mac), починаючи з Python 3.9.1, з експериментальним інсталятором) та неофіційною підтримкою, наприклад VMS. Переносність платформи є одним з її головних пріоритетів, і навіть OS/2 і Solaris підтримувалися під час Python 1 і 2; з тих пір рівень підтримки багатьох платформ знизився.

Python в основному розробляється за допомогою Програми покращення Python (PEP), яка є основним механізмом, який надає ключові нові можливості, покращує думку спільноти щодо проблем і фіксує рішення щодо дизайну Python. Стиль кодування Python наведено в PEP 8. Відмінні PEP розглядаються та коментуються спільнотою Python та радою директорів.

Покращення мови відповідають розробці довідкової реалізації CPython. Список розсилки python-dev є основним форумом для розробки мови. Конкретні питання обговорюються в інструменті відстеження помилок Roundup на bugs.python.org. Спочатку розробка велася у власному сховищі вихідного коду Mercurial, поки Python не перейшов на GitHub у січні 2017 року.

Відповідно до того, наскільки збільшилася кількість версій, існує три типи загальнодоступних випусків CPython:

Зворотна мовна версія несумісна. У цьому випадку код буде зламаний і потрібне ручне перенесення. Перша частина номера версії збільшена. Ці версії рідкісні-версія 3.0 вийшла через 8 років після виходу версії 2.0.

Основна або «функціональна» версія виходить приблизно кожні 18 місяців, але з прийняттям щорічної каденції випуску, починаючи з Python 3.9, очікується, що випускатиме її щорічно. Вони в основному сумісні, але вводять нові функції.

Друга частина номера версії збільшена. Патч підтримує всі основні версії протягом кількох років після його випуску.

Версії, які не містять нових функцій, випускаються приблизно кожні 3 місяці і випускаються, коли було виправлено достатню кількість помилок з моменту останнього випуску. Уразливості безпеки також були усунені в цих версіях. Третя і остання частина номера версії збільшено.

Багато альфа-, бета- та реліз-кандидатів також попередньо переглядаються та тестуються перед остаточним релізом. Хоча кожна проблема має приблизний графік, якщо код не готовий, вони зазвичай

затримуються. Команда розробників Python відстежує стан коду, виконуючи велику кількість модульних тестів під час процесу розробки.

Головна академічна конференція для Python — PyCon. Також існують спеціальні навчальні програми Python, наприклад Pyladies.

Pythons 3.10 припиняє wstr (буде видалено в Python 3.12; це означає, що розширення Python потрібно буде змінити до того часу), а також планує додати відповідність шаблону до мови.

Використання

З 2003 року Python є однією з десяти найпопулярніших мов програмування в індексі програмного забезпечення TIOBE, а станом на лютий 2021 року це третя за популярністю мова (після Java та C). Вона була обрана мовою програмування року («Річний найвищий ріст рейтингу») у 2007, 2010, 2018 та 2020 роках (єдина мова, яка робила це чотири рази).

Емпіричні дослідження показали, що для проблем програмування, включаючи маніпулювання рядками та пошук у словнику, мови сценаріїв, такі як Python, є більш продуктивними, ніж звичайні мови, такі як C і Java, і виявилося, що споживання пам'яті часто краще, ніж у Java, і немає краще, ніж C або Наскільки поганий C++".

Великі організації, які використовують Python, включають Wikipedia, Google, Yahoo!, CERN, NASA, Facebook, Amazon, Instagram, Spotify, а також деякі менші організації, такі як ILM та ITA. Сайт соціальних новин Reddit в основному написаний на Python.

Python може бути мовою сценаріїв для веб-програм, наприклад mod_wsgi для веб-сервера Apache. Стандартні API були розроблені для полегшення роботи цих програм через інтерфейс шлюзу веб-сервера. Веб-фреймворки, такі як Django, Pylons, Pyramid, TurboGears, web2py, Tornado, Flask, Bottle і Zope, підтримують розробників для розробки та підтримки складних програм. Pyjs і IronPython можна використовувати для розробки клієнтської сторони програм на основі Ajax. SQLAlchemy можна використовувати як перетворювач даних у реляційних базах даних. Twisted —

це фреймворк для програмного зв’язку між комп’ютерами, який використовується (наприклад) Dropbox.

Такі бібліотеки, як NumPy, SciPy і Matplotlib, дозволяють ефективно використовувати Python в наукових обчисленнях, тоді як спеціалізовані бібліотеки, такі як Biopython і Astropy, забезпечують функції, залежні від домену. Програмне забезпечення SageMath-Math з інтерфейсом ноутбука, яке можна запрограмувати на Python: його бібліотека охоплює багато аспектів математики, включаючи алгебру, комбінаторику, числову математику, теорію чисел і числення. OpenCV має палітурні прив'язки з багатим набором функцій для комп'ютерного зору та обробки зображень.

Python часто використовується в проектах штучного інтелекту та машинного навчання, включаючи такі бібліотеки, як TensorFlow, Keras, Pytorch і Scikit-learn. Python, як мова сценаріїв з модульною архітектурою, простим синтаксисом і багатими інструментами обробки тексту, часто використовується для обробки природних мов. Python був успішно інтегрований у багато програмних продуктів як мова сценаріїв, включаючи програмне забезпечення кінцевих елементів, таке як Abaqus, засоби моделювання 3D-параметрів, такі як FreeCAD, пакети 3D-анімації, такі як 3ds Max, Blender, Cinema 4D, Lightwave, Houdini, Maya, modo, MotionBuilder, Softimage, композитор візуальних ефектів Nuke, програми для 2D-зображень (такі як GIMP, Inkscape, Scribus і Paint Shop Pro) і програми для нот (наприклад, автор і група). Налагоджувач GNU використовує Python як хороший принтер для відображення складних структур, таких як контейнери C++. Esri рекламує Python як найкращий вибір для написання сценаріїв у ArcGIS. Він також використовувався в багатьох відеоіграх і був прийнятий як перша з трьох доступних мов програмування в Google App Engine, двома іншими є Java і Go.

Багато операційних систем використовують Python як стандартний компонент. Він постачається з більшістю дистрибутивів Linux, AmigaOS 4 (з використанням Python 2.7), FreeBSD (як пакет), NetBSD, OpenBSD (як пакет) і macOS, і може використовуватися з командного рядка (термінал). У багатьох

дистрибутивах Linux використовуються інсталятори, написані на Python: Ubuntu використовує інсталятор Ubiquity, а Red Hat Linux і Fedora — інсталятор Anaconda. Gentoo Linux використовує Python у своїй системі керування пакетами Portage.

Python широко використовується для інформаційної безпеки, включаючи розробку експлойтів.

Більшість додатків Sugar Laptop XO, які зараз розробляє Sugar Labs, написані на Python.

Проект одноплатної комп’ютерної програми Raspberry Pi прийняв Python як основну мову програмування користувачів.

LibreOffice включає Python і має намір замінити Java на Python. Його постачальник сценаріїв Python є основною особливістю версії 4.0 від 7 лютого 2013 року.

Виходячи зі сфери застосування та величезних можливостей мови Python, описаної вище, можна зробити висновок, що вона універсальна. Ось чому ця мова програмування була обрана для виконання цієї роботи.

2.2 Огляд середовища розробки

PyCharm — це інтегроване середовище розробки (IDE) для комп’ютерного програмування, особливо для Python. Його розробила чеська компанія JetBrains. Він забезпечує аналіз коду, графічний налагоджувач, інтегрований модуль тестування та інтеграцію системи контролю версій (VCS), а також підтримує використання Django для веб-розробки та Anaconda для аналізу даних.

PyCharm є кросплатформним і має версії для Windows, macOS та Linux. Спільнота доступна за ліцензією Apache, а також є професійна версія з додатковими функціями, випущена за власною ліцензією.

Бета-версія була випущена в липні 2010 року, а версія 1.0 з'явилася через 3 місяці. Версія 2.0 була випущена 13 грудня 2011 року, версія 3.0 – 24 вересня 2013 року, версія 4.0 – 19 листопада 2014 року.

PyCharm Community Edition — версія PyCharm з відкритим кодом, яка була запущена 22 жовтня 2013 року.

2.3 Огляд СКБД

Для структурованого зберігання використовуються бази даних SQLite – популярний формат баз даних, який з'являється в багатьох мобільних системах і традиційних операційних системах. SQLite — це технологічна бібліотека, яка реалізує незалежну конфігурацію бази даних SQL без серверів із нульовою конфігурацією. Код SQLite є загальнодоступним, тому його можна використовувати безкоштовно для будь-яких цілей, комерційних чи приватних.

SQLite — це вбудований механізм баз даних SQL. На відміну від більшості інших баз даних SQL, SQLite не має окремого серверного процесу. SQLite безпосередньо читає та записує звичайні дискові файли. Повна база даних SQL з кількома таблицями, індексами, тригерами та представленнями міститься в одному дисковому файлі. Як правило, чим швидше працює SQLite, тим більше пам'яті ви надаєте йому. Однак навіть у середовищі з низьким рівнем пам'яті продуктивність зазвичай досить хороша. Відповідно до способу використання, SQLite може бути швидшим, ніж прямий ввід / вивід файлової системи.

Більшість вихідного коду SQLite призначено лише для тестування та перевірки. Автоматизований набір тестів запускає мільйони тестових випадків, включає сотні мільйонів окремих операторів SQL і забезпечує 100% тестування гілок. SQLite реагує на помилки виділення пам'яті та дискового вводу-виводу. Все це перевіряється автоматизованим тестуванням за допомогою спеціальних інструментів тестування, які імітують збої системи. Звичайно, навіть з усіма цими тестами все одно є помилки. Але на відміну від деяких подібних проектів (особливо від комерційних конкурентів), SQLite відкритий і чесний щодо всіх помилок, а також надає список помилок і похвилинну історію змін коду.

Тому, щоб чітко зрозуміти переваги, ми вибрали одного з конкурентів SQLite, а саме SQL Server, а потім коротко порівняли їх:

- модель ціноутворення:

- SQLite безкоштовний;

- Серверні операційні системи:
  - SQLite не потребує сервера;
  - SQL Server працює на Linux та Windows;
- API та інші методи доступу:
    - SQLite підтримує такі драйвери:
      - ADO.NET;
      - JBDC;
      - ODBC;
    - SQL Server підтримує:
      - OLE DB;
      - ADO.NET;
      - JBDC;
      - ODBC;
      - Табличний потік даних (TDS);
- Підтримувані мови програмування:
  - SQL Server підтримує такі мови програмування (C#, C++, Delphi, Go, Java, JavaScript (Node.js), PHP, Python, R, Ruby, Visual Basic);
  - SQLite підтримує майже будь-які мови програмування, про які ви можете подумати (Actionscript, Ada, Basic, C, C#, C++, D, Delphi, Forth, Fortran, Haskell, Java, JavaScript, Lisp, Lua, MatLab, Objective-C, OCaml, Perl, PHP, PL/SQL, Python, R, Ruby, Scala, Scheme, Smalltalk, Tcl);
- Підтримка збереженої процедури:
  - SQLite не підтримує збережену процедуру;
  - SQL Server має його з мовами Transact SQL та .NET;
- Методи розділення:
  - SQLite не підтримує;

  - У SQL Server таблиці можуть розподілятися між кількома файлами (горизонтальне розділення);
- Методи реплікації:
  - SQLite не підтримує;
  - SQL Server підтримує;
- Контроль доступу користувачів:
  - SQLite не має концепції контролю доступу користувачів;

SQL Server має чіткі права доступу відповідно до стандарту SQL.

Виходячи з усіх описаних вище переваг SQLite та факту первинної наявності бази даних SQLite в проектах Django, основною СКБД для розроблення даної інформаційної системи обрано саме SQLite.

2.4 Огляд додаткового інструментарію

Django — це безкоштовний веб-фреймворк на основі Python, який відповідає архітектурі моделі-шаблон-перегляд (MTV). Його підтримує Django Software Foundation (DSF), американська незалежна організація, заснована як неприбуткова організація 501 (c) (3).

Основна мета Django — полегшити створення складних веб-сайтів, керованих базою даних. Структура підкреслює можливість повторного використання та «з'єднання» компонентів з меншою кількістю коду, низькою зв'язністю, швидкістю розробки та принципом «неповторності». Python є скрізь, навіть для налаштувань, файлів і моделей даних. Django також надає додатковий інтерфейс керування для створення, читання, оновлення та видалення.Він динамічно генерується за допомогою інтроспекції та налаштовується за допомогою моделі адміністратора.

Деякі відомі веб-сайти, які використовують Django, включають PBS, Instagram, Mozilla, The Washington Times, Disqus, Bitbucket і Nextdoor.

HTML – це мова розмітки, що використовується веб-браузерами для інтерпретації та об’єднання тексту, зображень та інших матеріалів у візуальні або аудіо веб-сторінки. Характеристики за замовчуванням для кожного елемента розмітки HTML визначаються в браузері, і ці характеристики можуть бути змінені або покращені веб-дизайнером за допомогою додаткового CSS. Технічний звіт ISO TR TR 9537 містить багато текстових елементів. Технологія, що використовує SGML, також охоплює особливості ранніх мов форматування тексту, наприклад команду RUNOFF, яка була розроблена на початку 1960-х для системи CTSS (Shared Time Sharing System): ці команди форматування отримані з друкарських машинок для ручного форматування команд документа. . Однак узагальнена нотація SGML заснована на елементах (вкладена область анотації з атрибутами), заснована не тільки на ефекті друку, а й на розділенні структури та розмітки; HTML поступово рухався в цьому напрямку за допомогою CSS.

Бернерс-Лі вважає HTML програмою SGML. У середині 1993 року Інженерна група з розробки Інтернету (IETF) випустила першу пропозицію щодо специфікації HTML, Інтернет-проект Бернерса-Лі та Дена Конноллі «Мова розмітки гіпертексту (HTML)», який включав визначення типів документів SGML, які використовуються для визначення граматики. Термін дії проекту закінчився через шість місяців, але він був визнаний спеціальною етикеткою NCSA Mosaic, яка використовується для вбудовування вбудованих зображень, що відображає стандартну концепцію IETF, засновану на успішних прототипах. Так само з кінця 1993 року конкуруючий Інтернет-проект Дейва Реггетта HTML + (Hypertext Markup Format) запропонував стандартизувати вже реалізовані функції, такі як електронні форми та повні форми.

Після закінчення терміну дії HTML і HTML+ на початку 1994 року IETF заснувала Робочу групу HTML, яка завершила створення «HTML 2.0» у 1995 році, що стало першою специфікацією HTML, яка вважається стандартом, на якому повинні базуватися майбутні реалізації.

Подальшому розвитку під егідою IETF заважають конкуруючі інтереси. З 1996 року консорціум World Wide Web Consortium (W3C) і комерційні постачальники програмного забезпечення підтримують специфікацію HTML. Однак у 2000 році HTML також став міжнародним стандартом (ISO/IEC 15445:2000). HTML 4.01 був випущений наприкінці 1999 року, і до 2001 року не було випущено жодних додаткових помилок. У 2004 році HTML5 був розроблений Робочою групою з технологій застосування веб-технологій (WHATWG), що стало спільним результатом W3C у 2008 році, і був завершений і стандартизований 28 жовтня 2014 року.

Семантичний HTML – це спосіб написання HTML, який підкреслює важливість закодованої інформації в її представленні (вигляді). HTML містить семантичну розмітку з самого початку, але він також містить розмітку презентації, таку як теги <font>, <i> і . Існують також семантично нейтральні теги span і div. Починаючи з кінця 1990-х років, коли каскадні таблиці стилів почали працювати в більшості браузерів, веб-авторам було

рекомендовано уникати використання розмітки презентації HTML для розділення презентації та вмісту.

Під час дискусії про семантичну павутину в 2001 році Тім Бернерс-Лі та інші навели приклади способів, за допомогою яких «агенти» інтелектуального програмного забезпечення могли автоматично сканувати Інтернет і знаходити, фільтрувати та співвідносити раніше не пов'язані опубліковані факти на благо людства. Такі проксі навіть зараз не поширені, але деякі ідеї веб 2.0, змішувача та веб-сайтів порівняння цін можуть бути близькими. Основна відмінність між цими гібридами веб-додатків і семантичним проксі Бернерса-Лі полягає в тому, що узагальнення та гібридизація поточної інформації зазвичай розробляється і порівнюється веб-розробниками, які вже знають веб-розташування та семантику API конкретних даних, до яких вони хочуть підключитися. І комбінований.

Важливим типом веб-проксі є веб-сканер або пошукова система, яка автоматично сканує та читає веб-сторінки, але не знає, що може знайти. Ці програмні агенти покладаються на семантичну чіткість веб-сторінок, які вони знаходять, оскільки вони використовують різні технології та алгоритми для читання та індексації мільйонів веб-сторінок щодня, а також надають інструменти пошуку для користувачів Інтернету, що значно зменшить корисність Світу. Широка мережа .

Для того, щоб павуки пошукових систем оцінили цінність фрагментів тексту, які вони знаходять у HTML-документах, а також тих, хто створює змішувачі та інші гібриди, і оскільки вони розробляють більш автоматизовані агенти, існує широка потреба в семантичній структурі в HTML. Рівномірно використовується для визначення значення опублікованих текстів.

Розмітка презентації в поточних рекомендаціях щодо HTML і XHTML застаріла. Більшість функцій презентації попередніх версій HTML більше не дозволяється, оскільки вони призводять до зниження доступності, підвищення витрат на обслуговування сайту та збільшення розміру документів.

Хороший семантичний HTML також покращує доступність веб-документів. Наприклад, коли програма зчитування з екрана або аудіобраузер можуть правильно створити документ, вони не витрачатимуть час людей із вадами зору на читання повторюваної або застарілої інформації з правильною розміткою. Всесвітня павутина в основному складається з документів HTML, які передаються з веб-сервера до веб-браузера за допомогою протоколу передачі гіпертексту (HTTP). Проте HTTP використовується для надання зображень, аудіо та вмісту, відмінного від HTML. Для того, щоб веб-браузер знав, як обробляти кожен отриманий документ, разом з документом передається інша інформація. Ці метадані зазвичай включають тип MIME (наприклад, text/html або application/xhtml + xml) і кодування символів.

У сучасних браузерах тип MIME, надісланий разом із HTML-документом, впливає на початкову інтерпретацію документа. Документи, надіслані за допомогою типу XHTML MIME, мають бути правильно сформованими XML; синтаксичні помилки призведуть до непрацездатності браузера. Той самий документ, надісланий за допомогою типу HTML MIME, може бути успішно відображений, оскільки деякі браузери більш толерантні до HTML.

У Рекомендації W3C зазначено, що документи XHTML 1.0, які відповідають специфікаціям у Додатку C до Рекомендації, можуть мати тип MIME. XHTML 1.1 також передбачає, що документи XHTML 1.1 мають бути позначені типами MIME.

Більшість графічних поштових клієнтів дозволяють використовувати підмножину HTML (часто неправильно визначену) для забезпечення форматування та семантичної розмітки, яку не може забезпечити звичайний текст. Це може включати друковану інформацію, таку як кольорові заголовки, підкреслений і цитований текст, вбудовані зображення та графіки. Багато з цих клієнтів включають редактор графічного інтерфейсу для створення електронних листів HTML і механізм візуалізації для їх відображення. Деякі люди критикують проблеми сумісності використання HTML в електронній

пошті, оскільки це може допомогти замаскувати фішингові атаки, проблеми з доступністю для сліпих або слабозорих, оскільки це може заплутати фільтри спаму, а також оскільки розмір повідомлення більше, ніж чистий текст. . Додаток HTML (HTA; розширення файлу «.hta») — це програма Microsoft Windows, яка використовує HTML і динамічний HTML у браузері для надання графічного інтерфейсу для програми. Звичайні файли HTML обмежуються моделлю безпеки веб-браузера, передаються лише на веб-сервер і обробляють лише об'єкти веб-сторінки та файли cookie. HTA працює як повністю надійна програма, тому має більше дозволів, таких як створення/редагування/видалення файлів та записів у реєстрі Windows. Оскільки вони працюють за межами моделі безпеки браузера, HTA не може виконуватися через HTTP, але має бути завантажений з локальної файлової системи (наприклад, файли EXE) і виконаний.

Каскадні таблиці стилів (CSS) — це мова таблиць стилів, яка використовується для опису подання документів, написаних мовою розмітки, як-от HTML. CSS, як і HTML і JavaScript, є наріжним каменем Всесвітньої мережі.

CSS має на меті розділити презентацію та вміст, включаючи макет, кольори та шрифти. Це поділ може покращити зручність використання вмісту, забезпечити більшу гнучкість та контроль у специфікаціях презентації, а також дозволить форматувати декілька веб-сторінок разом, вказавши відповідний CSS в окремих файлах .css, тим самим зменшуючи складність структурованого вмісту. Також дозволяє файли .css кешувати, щоб підвищити швидкість завантаження сторінки між обміном файлами та відформатованими сторінками.

Розділення формату та вмісту також дає змогу представити одну й ту саму сторінку розмітки в різних стилях для різних методів візуалізації, таких як екран, друк, голос (через мовний браузер або програму зчитування з екрана) та тактильні гаджети на основі Брайля. При доступі до вмісту на мобільних пристроях CSS також має альтернативні правила форматування. Каскадна

назва походить від заданої схеми пріоритету, щоб визначити, яке правило стилю застосовується, коли декілька правил відповідають певному елементу. Ця каскадна схема пріоритетів є передбачуваною.

Консорціум World Wide Web Consortium (W3C) підтримує специфікацію CSS. Тип Інтернет-медіа/текст css (тип MIME) зареєстровано для CSS RFC 2318 (березень 1998 р.). W3C надає безкоштовні послуги перевірки CSS для документів CSS.

Крім HTML, інші мови розмітки також підтримують використання CSS, включаючи XHTML, звичайний XML, SVG і XUL.

CSS має простий синтаксис і використовує кілька англійських ключових слів для представлення назв різних атрибутів стилю.

Таблиця стилів складається зі списку правил. Кожне правило або набір правил складається з одного або кількох селекторів і рекламного блоку.

У CSS селектор оголошує, до якої частини розмітки застосовано стиль, відповідаючи тегам і атрибутам у самій розмітці.

Селектори можуть застосовувати такі документи:

- всі елементи певного типу, напр. заголовки другого рівня h2
- елементи, визначені атрибутом, зокрема:
    - id: унікальний ідентифікатор у документі, ідентифікований префіксом хешу, наприклад #id;
    - клас: ідентифікатор, який може анотувати кілька елементів у документі, ідентифікований префіксом точки, наприклад .classname;
- елементи залежно від того, як вони розміщені щодо інших у дереві документів.

Класи та ідентифікатори чутливі до регістру, починаються з літери та можуть містити буквено-цифрові символи, дефіси та підкреслення. Цей клас можна застосувати до будь-якої кількості екземплярів будь-якого елемента. Ідентифікатор можна застосувати лише до одного елемента.

Псевдокласи використовуються в селекторах CSS, щоб дозволити форматування на основі інформації, яка не міститься в дереві документів. Прикладом широко використовуваного псевдокласу є hover, який визначає вміст лише тоді, коли користувач «вказує» на видимий елемент, зазвичай наводячи на нього курсор миші. Він додається до селектора, наприклад: hover або #elementid:hover. Псевдокласи класифікують елементи документа, наприклад: посилання або: відвідано, при цьому вибираються псевдоелементи, які можуть складатися з деяких елементів, наприклад ::first-line або ::first-letter.

Селектори можна комбінувати різними способами для досягнення більшої специфічності та гнучкості. Кілька селекторів можна об'єднати в список інтервалів, щоб вказати елементи за розташуванням, типом елемента, ідентифікатором, класом або будь-якою комбінацією цих елементів. Важливий порядок селекторів. Наприклад, div .myClass {color: red;} застосовується до всіх елементів myClass в елементі div, а .myClass div {color: red;} застосовується до всіх елементів div в елементі myClass. Це не слід плутати з ідентифікаторами пулу, такими як div.myClass {color: red;}, який застосовується до елементів div класу myClass.

Рекламний блок складається зі списку оголошень у дужках. Кожне оголошення містить атрибут, двокрапку (:) і значення. Якщо в блоці є кілька оголошень, ви повинні вставити крапку з комою (;), щоб розділити кожне оголошення. Необов'язкова крапка з комою може використовуватися після останнього (або окремого) оголошення. Властивості вказані в стандарті CSS. Кожен атрибут має набір можливих значень. Деякі атрибути можуть впливати на будь-який тип елемента, тоді як інші атрибути застосовуються лише до певних груп елементів.

Значенням може бути ключове слово, наприклад "center" або "inherit", або числовий батьківський елемент, наприклад 200px (200 пікселів), 50vw (50% ширини області перегляду) або 80% (80% ширини області перегляду) . Ви можете використовувати ключові слова (наприклад, «червоний»), шістнадцяткові значення (наприклад, #FF0000, також скорочено як #F00),

значення RGB від 0 до 255 (наприклад, rgb (255, 0, 0)) , значення RGBA Представляє колір і альфа-прозорість (наприклад, rgba (255, 0, 0, 0,8)), або значення HSL або HSLA (наприклад, hsl (000, 100%, 50%), hsla (000, 100%, 50%) 80%)).

Ненульове значення, що представляє лінійне вимірювання, має включати одиницю довжини, яка може бути буквеним кодом або абревіатурою, наприклад 200px або 50vw, або символ відсотка, наприклад 80%. Деякі одиниці - см (сантиметри); дюйми (дюйми); мм (міліметри); точки (пік); і pt (точка) - абсолютні, що вказують на те, що розмір дисплея не залежить від структури сторінки; інші - em (em) ; ex (ex) і px (пікселі) є відносними, що означає, що такі фактори, як розмір шрифту батьківського елемента, впливатимуть на вимірювання відтворення. Ці вісім блоків є особливістю CSS 1 і будуть збережені в усіх наступних версіях. Запропонований модуль значення та одиниці CSS рівня 3, якщо він буде прийнятий як рекомендація W3C, надасть ще сім одиниць довжини: ch; ask; rem; vh; vmax; мінімум; та public.

До появи CSS майже всі атрибути презентації документів HTML містилися в тегах HTML. Усі кольори шрифту, стилі фону, вирівнювання елементів, межі та розміри повинні бути чітко описані в HTML, зазвичай повторювані описи. CSS дозволяє авторам переносити більшу частину інформації в інший файл, таблицю стилів, що полегшує HTML.

Наприклад, заголовок (елемент h1), підзаголовок (h2), підзаголовок (h3) тощо визначено структурно за допомогою HTML. Вибір шрифту, розміру, кольору та акценту для цих елементів є наочним на друкованому виробі та на екрані. До CSS автори документів, які хотіли призначити такі типографічні функції всім заголовкам h2, повинні були повторювати розмітку представлення HTML для кожного входження такого типу заголовка. Це робить документ складнішим, більшим, більш схильним до помилок і важчим для підтримки. CSS дозволяє відокремити презентацію від структури. CSS може визначати кольори, шрифти, вирівнювання тексту, розмір, межі,

інтервали, макет та багато інших типографічних функцій і може виконувати ці операції незалежно для перегляду екрана та друку.

CSS також визначає невізуальні стилі, такі як швидкість читання та акценти для звукових читачів тексту. Зараз W3C не підтримує використання всієї презентаційної розмітки HTML.

Наприклад, під HTML до CSS елемент заголовка, визначений червоним текстом, буде записаний як:

```
<h1> <font color = "red"> Глава 1. </font> </h1>
```

Використовуючи CSS, один і той же елемент можна закодувати, використовуючи властивості стилю замість презентаційних атрибутів HTML:

```
<h1 style = "color: red;"> Глава 1. </h1>
```

Переваги цього можуть бути не одразу ясними, але сила CSS стає більш очевидною, коли властивості стилю розміщуються у внутрішньому елементі стилю або, ще краще, у зовнішньому файлі CSS. Наприклад, припустимо, документ містить елемент стилю:

```
< style >
   h1 {
      колір: червоний;
   }
</style>
```

Тоді всі елементи h1 у документі автоматично стануть червоними, не вимагаючи явного коду. Якщо згодом автор хотів зробити елементи h1 синіми, це можна зробити, змінивши елемент стилю на:

```
< style >
   h1 {
      колір: синій;
   }
</style>
```

а не шляхом копіткого перегляду документа та зміни кольору для кожного окремого елемента h1.

Стилі також можна помістити у зовнішній файл CSS, як описано нижче, і завантажити, використовуючи синтаксис, подібний до:

<link href = "path / to / file.css" rel = "stylesheet" type = "text / css">

Це додатково відокремлює стилі від документів HTML і дозволяє редагувати стилі кількох документів, просто редагуючи спільний зовнішній файл CSS.

Інформацію CSS можна надати з різних джерел. Такими джерелами можуть бути веб-браузери, користувачі та автори. Інформація від авторів може бути класифікована за вбудованими, типом носія, важливістю, специфікою селектора, порядком правил, успадкуванням і визначеннями атрибутів. Інформація про стиль CSS може бути в окремому документі або вбудованою в документ HTML. Ви можете імпортувати кілька таблиць стилів. Залежно від використовуваного вихідного пристрою можна використовувати різні стилі; наприклад, екранна версія може сильно відрізнятися від друкованої, тому автор може налаштувати презентацію відповідно до кожного засобу.

Таблиця стилів з найвищим пріоритетом керує відображенням вмісту. Претензії, які не встановлені як джерело з найвищим пріоритетом, будуть передані джерелам з нижчим пріоритетом, наприклад стилям агента користувача. Процес називається каскадним.

Однією з цілей CSS є надання користувачам більшого контролю над представити. Люди, яким важко читати заголовки, виділені червоним курсивом, можуть використовувати іншу таблицю стилів. Відповідно до різних браузерів і веб-сайтів, користувачі можуть вибрати одну з різних таблиць стилів, наданих дизайнером, або видалити всі додані стилі та використовувати стиль браузера за замовчуванням для перегляду сайту, або просто замінити стиль червоного курсиву без зміни атрибутів.

Спадкування є ключовою особливістю CSS; воно спирається на відносини між предками та нащадками. Спадкування — це механізм, за допомогою якого атрибути можуть застосовуватися не тільки до певного елемента, але й до його нащадків. Спадок покладається на дерево документів,

яке є ієрархією елементів XHTML на вкладених сторінках. Елементи-нащадки можуть успадковувати значення властивостей CSS від будь-якого елемента-предка, який їх містить. Нащадки зазвичай успадковують властивості, пов’язані з текстом, але не властивості, пов’язані з вікном. Властивості, які можуть бути успадковані, включають колір, шрифт, інтервал між літерами, висоту рядка, стиль списку, вирівнювання тексту, відступ тексту, перетворення тексту, видимість, пробіли та пробіли між словами. Властивості, які не можуть бути успадковані, — це фон, межа, відображення, плаваюча частина і очищення, висота і ширина, поля, мінімальна і максимальна висота і ширина, контур, переповнення, відступ, положення, формат тексту, вертикальне вирівнювання та z-індекс.

Спадкування можна використовувати, щоб уникнути повторного оголошення певних властивостей у таблиці стилів, дозволяючи коротший CSS.

Спадкування в CSS відрізняється від успадкування в мовах програмування на основі класів, де клас B можна визначити як «як клас A, але модифікований». За допомогою CSS ви можете використовувати «Клас A, але змінити» для стилізації елемента. Однак неможливо визначити клас CSS B, і тоді ви можете використовувати його для стилізації кількох елементів без повторних змін.

CSS вперше запропонував Хокон Віум Лі 10 жовтня 1994 року. У той час Лі працював з Тімом Бернерсом-Лі в ЦЕРН. Приблизно в той же час було запропоновано кілька інших мов таблиць стилів для Інтернету. Обговорення загальнодоступних списків розсилки та в рамках World Wide Web Consortium привели до першої рекомендації W3C CSS (CSS1) [24], опублікованої в 1996 році. Особливо вплинула пропозиція Берта Босса; він є співавтором CSS1 і вважається співавтором CSS.

З моменту створення стандартної загальної мови розмітки (SGML) у 1980-х роках таблиці стилів завжди існували в тій чи іншій формі, а CSS був розроблений для надання таблиць стилів для Інтернету. Мовні таблиці стилів Однією з мовних вимог є те, що таблиці стилів надходять з різних джерел в

Інтернеті. Тому існуючі мови таблиць стилів, такі як DSSSL і FOSI, не підходять. З іншого боку, CSS дозволяє стилям документа впливати на кілька таблиць стилів за допомогою «каскадних» стилів.

З розвитком HTML він містить все більше і більше різноманітних можливостей стилів для задоволення потреб веб-розробників. Завдяки більш складному HTML ця еволюція дозволяє дизайнерам краще контролювати зовнішній вигляд сайту. Зміни, внесені веб-браузерами, такими як ViolaWWW і WorldWideWeb, ускладнюють постійний перегляд веб-сайтів, і користувачі мають менше контролю над тим, як відображатиметься веб-вміст. Браузер/редактор, розроблений Тімом Бернерсом-Лі, має таблицю стилів, яка суворо закодована в програмі. Тому таблиці стилів не можуть бути пов’язані з документами в Інтернеті. Роберт Кайо, також із CERN, хоче відокремити структуру від презентації, щоб різні таблиці стилів могли описувати різні друковані презентації, екранні презентації та редактори.

Покращення можливостей веб-презентації завжди було темою, яка цікавила багатьох людей у веб-спільноті, і дев’ять різних мов таблиць стилів були запропоновані в списку розсилки в стилі www. Серед дев’яти пропозицій дві мають особливо сильний вплив на CSS: каскадні таблиці стилів HTML і пропозиції таблиць стилів на основі потоку (SSP). Два браузери були використані як початкові рекомендовані тести; Лі та Ів Лафон співпрацювали над впровадженням CSS у браузері Arena Дейва Реджетта. Берт Босс реалізував власний продукт SSP у браузері Argo. Пізніше Лі і Бос спільно розробили стандарт CSS ("H" було вилучено з назви, оскільки ці таблиці стилів також можна застосовувати до інших мов розмітки, крім HTML).

Пропозиція Лі була представлена на конференції Mosaic and Internet Conference (пізніше названа WWW2), що відбулася в Чикаго, штат Іллінойс, у 1994 році, і знову з Бертом Боссом у 1995 році. [23] Приблизно в цей час був створений W3C, який зацікавився розробкою CSS. З цією метою він організував семінар, який проводив Стівен Пембертон. Тому W3C додав роботу CSS до роботи Ради з огляду HTML (ERB). Лі та Босс є ключовими

технічними співробітниками в цьому аспекті проекту, включаючи інших учасників, включаючи Томаса Ріардона з Microsoft. У серпні 1996 року Netscape Communication Corporation представила альтернативну мову таблиць стилів під назвою JavaScript Sheets Style (JSSS). Специфікація ніколи не була завершена і застаріла. До кінця 1996 року CSS був готовий стати офіційною версією, а перший рівень рекомендаційних стандартів CSS був випущений у грудні.

HTML, CSS і DOM розробляються в груповій редакції HTML (ERB). На початку 1997 року ЄБРР був розділений на три робочі групи: робочу групу HTML під головуванням Дена Конноллі з W3C; робочу групу DOM під головуванням Лорен Вуд з SoftQuad; робочу групу CSS під головуванням Кріса Ліллі з W3C.

Робоча група CSS почала вирішувати невирішені проблеми в CSS рівня 1, що призвело до створення рівня 2 CSS рівня 4 листопада 1997 року. Він був опублікований як рекомендація W3C 12 травня 1998 року. CSS Level 3, який був запущений у 1998 році, все ще розробляється станом на 2014 рік.

У 2005 році робоча група CSS вирішила більш суворо імплементувати вимоги стандарту. Це означає, що опубліковані стандарти, такі як CSS 2.1, селектори CSS 3 і текст CSS 3, повертаються з пропозицій кандидатів на рівні робочого елемента.

Фреймворк CSS — це попередньо розроблена бібліотека, призначена для спрощення стилю веб-сторінок, що відповідають стандартам, за допомогою каскадної мови таблиць стилів. Структури CSS включають Blueprint, Bootstrap, Cascade Framework, Foundation і Materialize. Як і мови програмування та бібліотеки сценаріїв, фреймворки CSS зазвичай включають у вигляді зовнішніх таблиць .css, на які посилається HTML <head>. Вони надають безліч готових варіантів оформлення та розміщення веб-сторінок. Хоча багато з цих фреймворків було випущено, деякі автори в основному використовують їх для швидкого створення прототипів або досліджень, і вважають за краще «створювати» CSS, відповідний для кожного сайту випуску, без необхідності

розробляти, підтримувати та завантажувати багато невикористаних CSS для функцій стилю сайту.

Оскільки кількість ресурсів CSS, які використовуються в проекті, збільшується, командам розробників часто доводиться вибирати загальний метод проектування, щоб підтримувати їх організованість. Метою є простота розробки, простота співпраці під час розробки та розширена продуктивність таблиці стилів у браузері. Популярні методи включають OOCSS (об'єктно-орієнтований CSS), ACSS (атомний CSS), CSS (органічна каскадна таблиця стилів), SMACSS (масштабована і модульна архітектура для CSS) і BEM (блок, елемент, модифікатор).

2.5 Клієнт-серверна архітектура

Модель клієнт-сервер — це розподілена структура програми, яка розподіляє завдання або робочі навантаження між постачальниками ресурсів або послуг (так звані сервери) і запитувачами послуг (так звані клієнти). Клієнт і сервер зазвичай спілкуються в комп'ютерній мережі на різному обладнанні, але клієнт і сервер можуть бути в одній системі. Хост сервера запускає одну або кілька серверних програм, які спільно використовують ресурси з клієнтом. Клієнт зазвичай не ділиться жодними своїми ресурсами, але запитує вміст або послуги від сервера. Тому клієнт ініціює сеанс зв'язку з сервером, який очікує вхідних запитів. Прикладами комп'ютерних програм, які використовують модель клієнт-сервер, є електронна пошта, мережевий друк і всесвітня павутина.

Функція «клієнт-сервер» описує взаємозв'язок між інтерактивними програмами в програмі. Серверний компонент надає функцію або послугу одному або кільком клієнтам, які ініціюють такі запити на обслуговування. Сервери класифікуються відповідно до послуг, які вони надають. Наприклад, веб-сервер розміщує веб-сторінки, а файловий сервер — комп'ютерні сторінки. Загальними ресурсами може бути будь-яке програмне забезпечення для серверів і електронні компоненти, від програм і даних до процесорів і пристроїв зберігання даних. Спільний доступ до ресурсів сервера – це послуга.

Чи є комп'ютер клієнтом, сервером чи тим і іншим, визначається характером програми, яка потребує сервісних функцій. Наприклад, комп'ютер і програмне забезпечення файлового сервера можуть працювати на одному комп'ютері, щоб надавати різні дані клієнтам, які здійснюють різні типи запитів. Клієнтське програмне забезпечення також може взаємодіяти з програмним забезпеченням сервера на одному комп'ютері. Зв'язок між серверами, наприклад для синхронізації даних, іноді називають міжсерверним або міжсерверним.

Взагалі кажучи, служба — це абстракція комп'ютерних ресурсів, і клієнту не потрібно турбуватися про те, як працює сервер під час запиту та

відповіді. Клієнт повинен розуміти відповідь лише на основі відомого протоколу програми, тобто змісту та формату даних, запитуваних для послуги.

Клієнт і сервер обмінюються повідомленнями за допомогою шаблонів повідомлень запит-відповідь. Клієнт надсилає запит, а сервер повертає відповідь. Це повідомлення є прикладом міжпроцесорної комунікації. Для спілкування комп'ютери повинні мати спільну мову та правила, щоб і клієнт, і сервер знали, що станеться. Мова та правила спілкування визначаються протоколом спілкування. Усі клієнт-серверні протоколи працюють на рівні програми. Протокол прикладного рівня визначає основну схему діалогу. Для подальшої стандартизації обміну даними сервер може реалізувати інтерфейс прикладного програмування (API). API — це рівень абстракції для доступу до служб. Обмежуючи спілкування певним форматом вмісту, це допомагає аналізувати. Завдяки абстрактному доступу він полегшує міжплатформний обмін даними.

Сервер може отримувати запити від багатьох різних клієнтів за короткий проміжок часу. Комп'ютери можуть виконувати обмежену кількість завдань у будь-який час і покладатися на систему планування для визначення пріоритету вхідних запитів клієнтів для їх задоволення. Щоб запобігти зловживанням та максимізувати доступність, серверне програмне забезпечення може обмежувати доступність клієнтів. Атаки відмови в обслуговуванні призначені для використання зобов'язання сервера обробляти запити, перевантажуючи його надмірною частотою запитів. Якщо ви хочете передати конфіденційну інформацію між клієнтом і сервером, вам слід використовувати шифрування.

Коли клієнт банку використовує веб-браузер (клієнт) для доступу до послуг онлайн-банкінгу, клієнт ініціює запит на веб-сервер банку. Облікові дані клієнта можуть зберігатися в базі даних, а веб-сервер служить клієнтом для доступу до сервера бази даних. Сервер додатків використовує бізнес-логіку банку для інтерпретації повернених даних і надання виводу веб-сервера. Нарешті, веб-сервер повертає відображені результати клієнтському веб-браузеру.

На кожному кроці цієї послідовності обміну повідомленнями клієнт-сервер комп’ютер обробляє запит і повертає дані. Це шаблон повідомлення запит-відповідь. Коли всі запити задоволені, послідовність завершується, і веб-браузер представляє дані клієнту.

Цей приклад ілюструє шаблон дизайну, який підходить для моделі клієнт-сервер: поділ проблеми.

Модель клієнт-сервер не передбачає, що хост-сервер повинен мати більше ресурсів, ніж клієнт-хост. Навпаки, це дозволяє будь-якому комп’ютеру загального призначення розширювати свої функції за рахунок спільного використання ресурсів інших хостів. Однак централізовані обчислення виділяють велику кількість ресурсів для кількох комп’ютерів. Чим більше обчислень завантажено з клієнтського хоста на центральний комп'ютер, тим простіше клієнтський хост. Він значною мірою залежить від мережевих ресурсів (серверів та інфраструктури) для обчислень і зберігання. Бездискові вузли навіть завантажують власну операційну систему з мережі, а комп’ютерний термінал взагалі не має операційної системи, це лише інтерфейс введення-виведення сервера. Навпаки, товсті клієнти, такі як персональні комп’ютери, багаті ресурсами і не покладаються на сервери для виконання основних функцій.

З 1980-х до кінця 1990-х років, коли ціна та потужність мікрокомп'ютерів впали, багато організацій перемістили свої обчислення з централізованих серверів, таких як мейнфрейми та міні-комп'ютери, на товсті клієнти. Це робить управління комп’ютерними ресурсами більш персоналізованим, але управління інформаційними технологіями ускладнюється. У 2000-х веб-додатки були достатньо зрілими, щоб конкурувати з прикладним програмним забезпеченням, розробленим для певних мікроархітектур. Поява цього зрілого, більш доступного масового сховища та сервісно-орієнтованої архітектури є одним із факторів, які привели до тенденції хмарних обчислень у 2010-х роках.

На додаток до моделі клієнт-сервер, розподілені обчислювальні програми зазвичай використовують однорангову (P2P) архітектуру.

У моделі клієнт-сервер сервер зазвичай розроблений для роботи як централізованої системи та надання послуг багатьом клієнтам. Вартість обчислювальної потужності, пам’яті та пам’яті сервера слід скоригувати відповідно до очікуваного навантаження. Системи балансування навантаження та збоїв часто використовуються для масштабування серверів за межі однієї фізичної машини.

Балансування навантаження визначається як упорядкований та ефективний розподіл мережевого або додаткового трафіку між кількома серверами в фермі серверів. Кожен балансувальник навантаження знаходиться між клієнтськими пристроями та серверами, отримує вхідні запити та розподіляє їх на будь-який доступний сервер, який може їх виконати. У одноранговій мережі два або більше комп’ютери (однорангова мережа) об’єднують свої ресурси та взаємодіють у децентралізованій системі. Одноранговий вузол — це рівний або еквівалентний вузол в неієрархічній мережі. На відміну від клієнтів у мережі клієнт-сервер або черзі клієнт-клієнт, одноранговий зв’язок взаємодіє безпосередньо. У одноранговій мережі алгоритми однорангового зв’язку балансують навантаження, навіть однорангові з помірними ресурсами можуть допомогти розподілити навантаження. Якщо сайт стане недоступним, він залишиться доступним, доки інші партнери нададуть його спільні ресурси. В ідеалі одноранговій мережі не потрібно досягати високої доступності, оскільки інші додаткові однорангові вузли можуть компенсувати будь-які простої ресурсів; у міру зміни доступності та перевантаження протокол перенаправляє запити.

І клієнт-сервер, і головний-підпорядкований вважаються підкатегоріями розподілених однорангових систем.

РОЗДІЛ 3
ПРОЕКТУВАННЯ І РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Діаграма варіантів використання

При проектуванні функціонального навантаження системи слід проаналізувати використані варіанти, що дасть повне розуміння призначення та можливої діяльності інформаційної системи в майбутньому.

Найпростіша діаграма використання — це спосіб мислення про взаємодію між користувачем і системою, який показує відносини між користувачем і різними видами використання, в яких бере участь користувач. Діаграми використання можуть ідентифікувати різні типи користувачів системи та різні моделі використання, і часто супроводжуються діаграмами інших типів. Ціль зображується колом або еліпсом.

Хоча кожну можливість можна детально розглянути за допомогою самих параметрів, використання діаграм може допомогти надати огляд системи вищого рівня. Я вже говорив: «План використання — це принцип вашої системи».

Завдяки своїй спрощеній природі плани використання можуть бути хорошим інструментом комунікації для зацікавлених сторін. Ці малюнки намагаються змоделювати реальний світ і дозволити зацікавленим сторонам зрозуміти, як буде розроблена система. Сіау і Лі провели дослідження, щоб визначити, чи існують реальні сценарії використання чи не потрібні. Було виявлено, що використання діаграм передає намір системи зацікавленим сторонам більш спрощеним чином, і вони «повніше пояснюються, ніж діаграми класів».

Метою використання діаграм є показати динамічні аспекти системи. Для забезпечення повного функціонального та технічного представлення системи

можна використовувати додаткові схеми та документи. Вони забезпечують спрощене та графічне представлення того, що насправді має робити система.

- Межа системи - прямокутник з ім'ям і еліпсом (прецедент). За відсутності корисної інформації її зазвичай можна опустити,
- Актор (англ. actor) — стилізована роль людини, яка являє собою набір ролей користувачів, які взаємодіють із сутностями (системами, підсистемами, класами) (широке розуміння: люди, зовнішні сутності, класи, інші системи). Актори не можуть бути пов'язані один з одним (за винятком відносин обробки/дослідження),
- Прецедент - еліпс з написом, що вказує на виконану системну операцію (можливо, включаючи можливі варіанти), що привела до результатів, які спостерігає учасник. Назва може бути назвою або описом (з точки зору актора) системи «що» (а не «як»). Під час сценарію актори обмінюються системною інформацією. Сценарій можна відобразити в прецедентній діаграмі відео коментарів UML. Кілька різних сценаріїв можуть бути пов’язані з прецедентом

На рисунку 3.1 зображено діаграму варіантів використання, яка описує можливі дії користувача в системі.

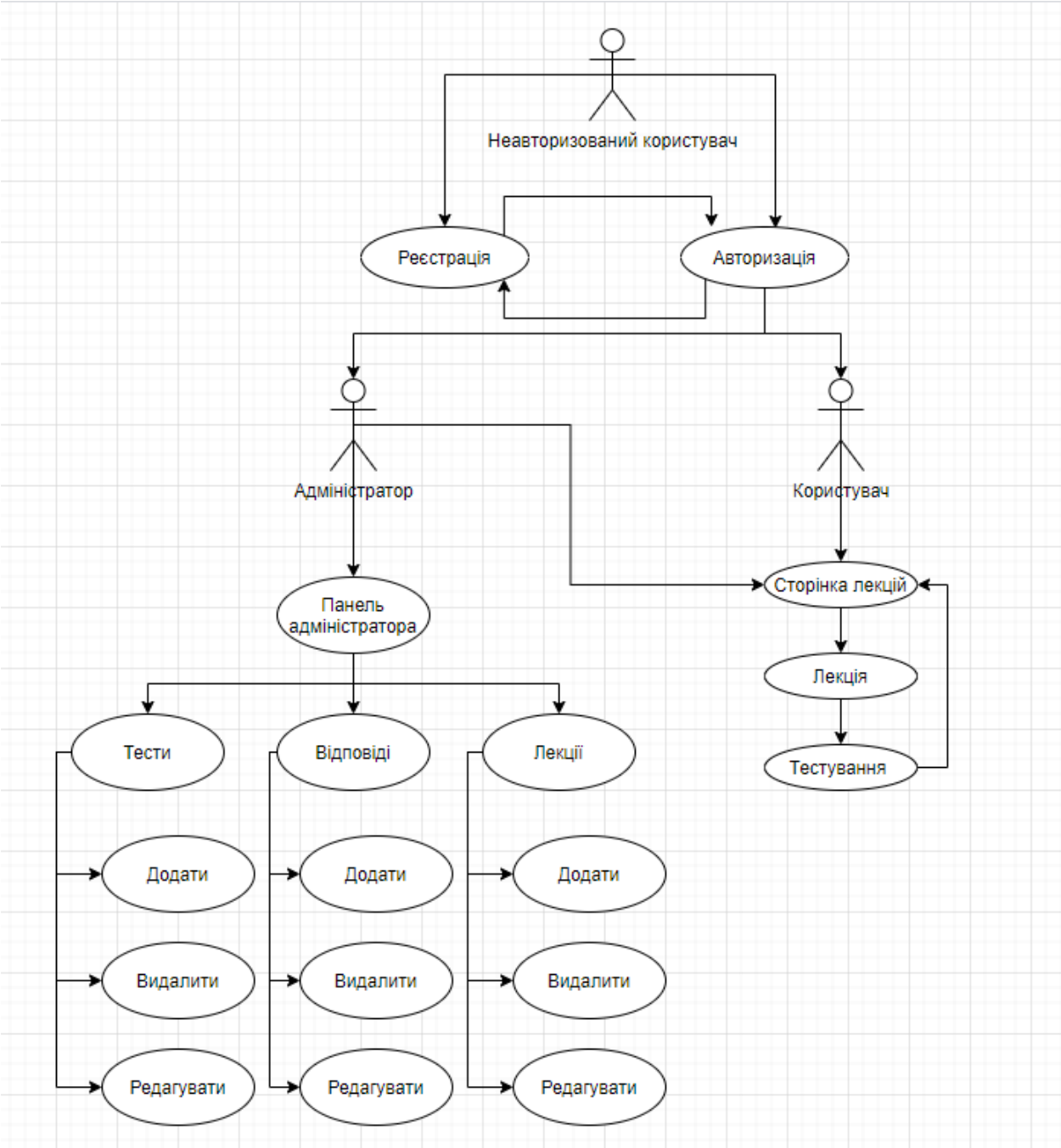


Рисунок 3.1 — Діаграма варіантів використання для користувача

3.2 Діаграма класів

При проектуванні програмного забезпечення внутрішня структура системи зазвичай відображається у вигляді діаграми класів, яка показує структуру класів і модулів проекту та взаємодію між ними.

У програмній інженерії діаграма класів уніфікованої мови моделювання (UML) — це статична структурна діаграма, яка описує структуру системи та показує системні класи, їх атрибути, операції (або методи) і зв’язки між об’єктами.

Діаграми класів є основними будівельними блоками об'єктно-орієнтованого моделювання. Використовується для загального концептуального моделювання структури програми та детального моделювання для перетворення моделі в програмний код. Діаграми класів також можна використовувати для моделювання даних. Класи на діаграмі класів представляють основні елементи, взаємодії в програмі та класи, що програмуються.

На малюнку клас представлений вікном, що містить три відділення:

- у верхньому відділенні міститься назва класу. Надруковано жирним шрифтом і відцентровано, а перша літера написана великими літерами;
- середній відсік містить атрибути класу. Вони вирівняні за лівим краєм, а перша буква мала;
- У нижньому відділенні містяться операції, які може виконувати клас. Вони також вирівняні по лівому краю, а перша літера є малою.

При проектуванні системи визначення багатьох класів і групування їх у схему класів допомагає визначити статичні відносини між ними. При детальному моделюванні категорія концептуального проектування зазвичай поділяється на кілька підкатегорій.

Залежність — це смислове відношення між залежними елементами та незалежними елементами моделі. Якщо зміна визначення одного елемента (сервера чи цілі) може призвести до зміни іншого (клієнта чи джерела), він

існує між двома елементами. Це об’єднання одностороннє. Релевантність відображається пунктирною лінією з відкритою стрілкою, що вказує від замовника до постачальника.

Для подальшого опису поведінки системи ці діаграми класів можуть бути доповнені діаграмами станів або автоматами станів UML.

Асоціація являє собою серію посилань. Бінарні асоціації (з двома кінцями) зазвичай виражаються у вигляді рядків. Асоціація може пов'язувати будь-яку кількість класів. Асоціація з трьома ланками називається потрійною асоціацією. Асоціацію можна назвати, а кінець асоціації можна змінити за допомогою таких атрибутів, як назва ролі, індикатор власності, множинність і видимість.

Існує чотири різні типи асоціацій: двостороння, одностороння, агрегаційна (включаючи комбіновану агрегацію) та рефлексивна. Найбільш поширеними є двосторонні та односторонні асоціації.

Наприклад, клас польоту асоціюється з класом літака з двостороннім рухом. Асоціація являє собою статичні відносини, спільні між об'єктами двох класів.

Агрегація є варіантом відношення "mas"; агрегація є більш специфічною, ніж асоціація. Це асоціація, яка представляє частину мети або частину відносин. Як показано на малюнку, у професора «має» клас для викладання. Як тип асоціації, агрегація може бути названа й мати ті самі модифікації, що й асоціація. Однак агрегація не може включати більше двох категорій; це має бути бінарна асоціація. Крім того, майже немає різниці між агрегацією та асоціацією в процесі впровадження, і графік може повністю опустити зв’язок агрегації. [7]

Коли клас є колекцією або контейнером інших класів, відбувається агрегація, але класи, що містяться, не мають сильної залежності від життєвого циклу контейнера. Коли контейнер знищено, вміст контейнера все ще існує.

В UML він представлений графічно у вигляді порожнистого ромба на класі хоста, з’єднавши його з однією лінією класу хоста. Агрегація - це

семантично розширений об'єкт, який розглядається як одиниця в багатьох операціях, хоча фізично складається з кількох менших об'єктів.

Приклад: бібліотека та студенти. Тут студенти можуть існувати без бібліотеки, а зв’язок між студентами та бібліотекою є сукупністю.

Це показує, що один із двох споріднених класів (підкласів) вважається спеціалізацією іншого (супертипу), а суперклас є узагальненням підкласів. На практиці це означає, що будь-який екземпляр підтипу також є екземпляром супертипу. Типове дерево узагальнень цієї форми можна знайти в біологічній класифікації: люди — це підклас вищих приматів, це підклас ссавців тощо. Цей зв’язок найлегше зрозуміти як фразу «А є Б» (люди — це ссавці, а ссавці — тварини).

Графічне представлення узагальнення UML — це форма порожнистого трикутника в кінці суперкласу рядків (або дерева рядків), що з’єднує його з одним або кількома підтипами.

Відношення узагальнення також називають спадковістю або відносинами «є».

Суперклас (базовий клас) у відносинах узагальнення також називають «батьківським класом», суперкласом, базовим класом або базовим типом.

Підтипи у відносинах спеціалізації також називаються "дочірніми" підкласами, похідними класами, похідними типами, успадкованими класами або успадкованими типами.

Зауважте, що ці відносини повністю відрізняються від біологічних відносин між батьком і дитиною: використання цих термінів дуже поширене, але може ввести в оману.

А є типом В

Наприклад, «дуб — це різновид дерева», «автомобіль — це вид транспорту»

Підсумок може відображатися лише в діаграмах класів і діаграмах використання.

У моделюванні UML відносини реалізації — це відносини між двома елементами моделі, де один елемент моделі (клієнт) реалізує (реалізує або виконує) поведінку, задану іншим елементом моделі (постачальником).

Графічне представлення реалізації UML являє собою порожнистий трикутник, з’єднаний з кінцем штрихового інтерфейсу (або дерева рядків) одного або кількох реалізаторів. Проста стрілка використовується в кінці пунктирного інтерфейсу, який з’єднує його з користувачем. На діаграмі компонентів використовується графічна умова «м'ячний розетка» (реалізатор кладе кульку або льодяник, а користувач показує кульку). Реалізація може бути відображена лише на діаграмі класів або компонентів. Реалізація – це зв’язок між класами, інтерфейсами, компонентами та пакетами, які з’єднують елементи замовника з елементами постачальника. Відношення реалізації між класом/компонентом та інтерфейсом показує, що клас/компонент реалізує операції, запропоновані інтерфейсом.

Залежність — це слабкіша форма спілкування, яка вказує на те, що один клас залежить від іншого класу, оскільки він використовує його в певний момент часу. Якщо незалежний клас залежить від змінної параметра або локальної змінної методу, то один клас залежить від іншого. Це відрізняється від асоціації, де атрибут залежного класу є екземпляром незалежного класу. Іноді відносини між цими двома класами слабкі. Вони взагалі не реалізуються зі змінними-членами. Натомість вони можуть бути реалізовані як параметри функцій-членів.

Представлення асоціації UML - це лінія, що з'єднує два пов'язані класи. На кожному кінці рядка є додаткові символи. Наприклад, ми можемо використовувати стрілку, щоб вказати, що кінчик можна побачити з хвоста стрілки. Ми можемо представити право власності, помістивши кульку, і роль, яку відіграє елемент на цьому кінці, полягає в вказуванні імені ролі та кількох екземплярів сутності (діапазон об’єктів, пов’язаних з іншого кінця).

Класи сутності моделюють довгоживучу інформацію, якою обробляє система, а іноді і поведінку, пов'язану з цією інформацією. Їх не слід ідентифікувати як таблиці баз даних чи інших сховищ даних.

Вони намальовані як кола з короткою лінією, прикріпленою до нижньої частини кола. Як варіант, їх можна намалювати як звичайні класи із позначенням стереотипу «сутність» над назвою класу.

На рисунку 3.2 зображено діаграму класів, яка відображає внутрішню будову проекту.

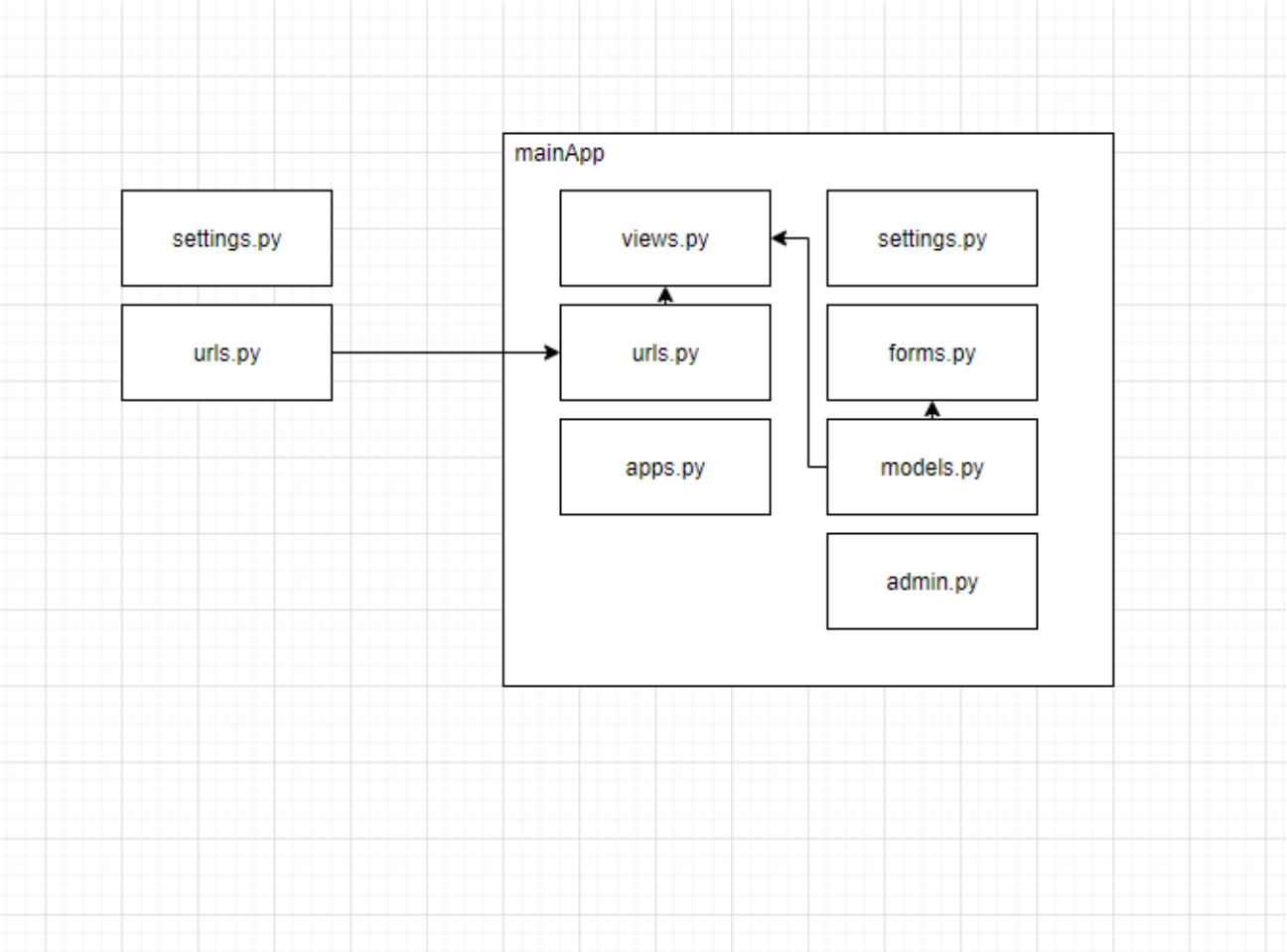


Рисунок 3.2 — Діаграма класів

3.3 Розробка графічного інтерфейсу

Графічний інтерфейс користувача складається з 4 основних сторінок, а саме:

- Сторінка авторизації (рис. 3.4)
- Сторінка реєстрації (рис. 3.5)
- Сторінка активних лекцій (рис. 3.6)
- Сторінка лекції (рис. 3.7)
- Сторінка опитування(рис. 3.8)

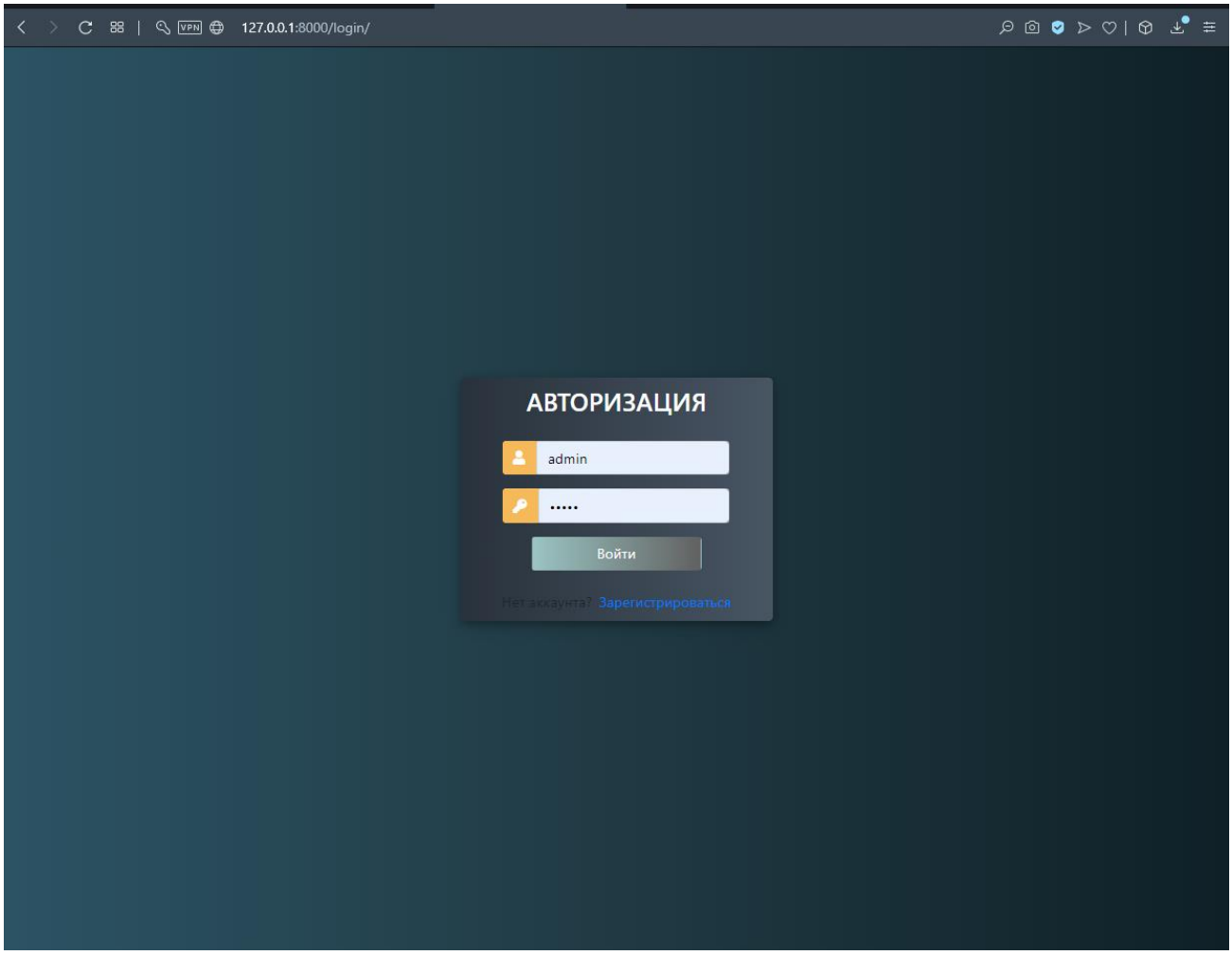


Рисунок 3.4 — Сторінка авторизації

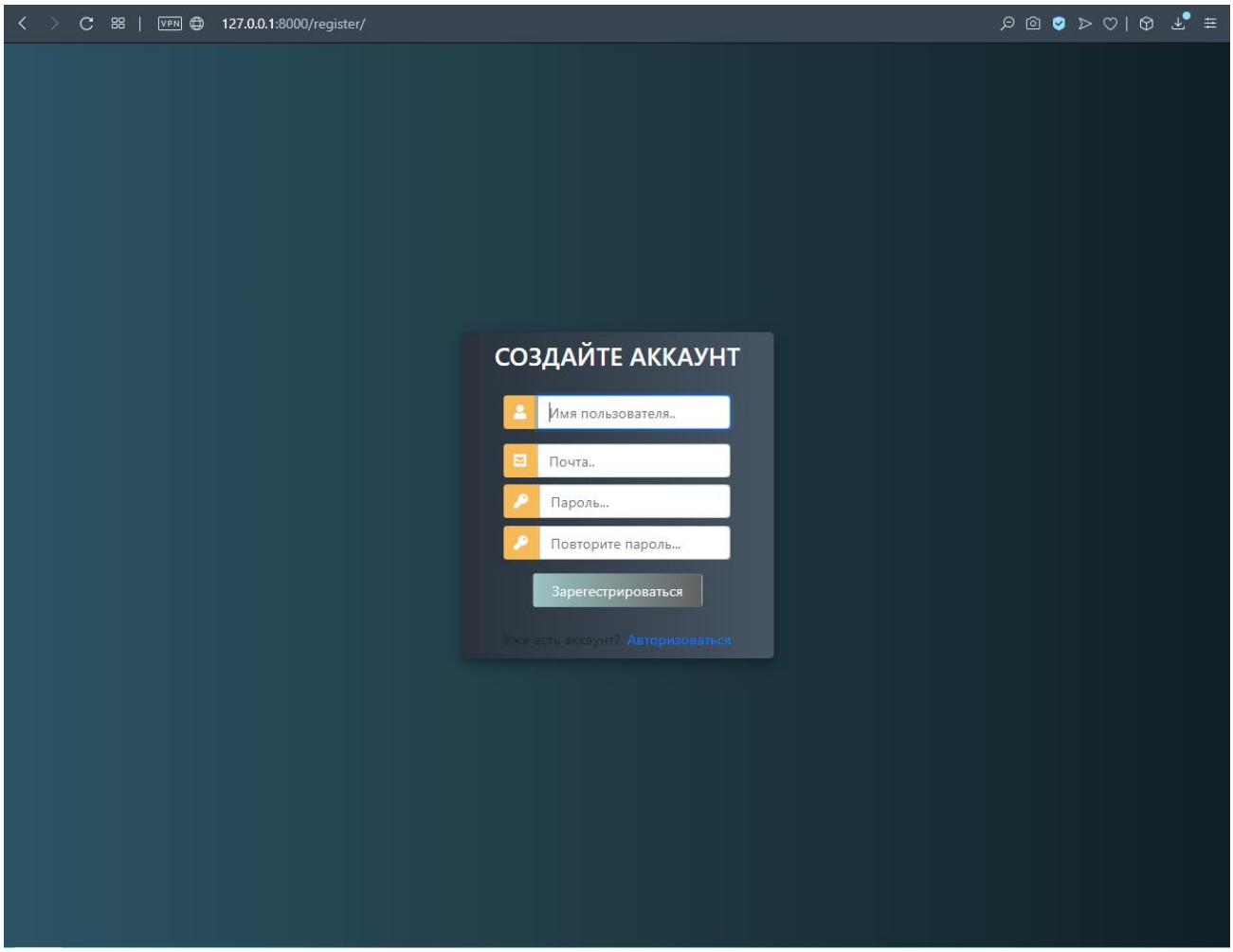


Рисунок 3.5 — Сторінка реєстрації

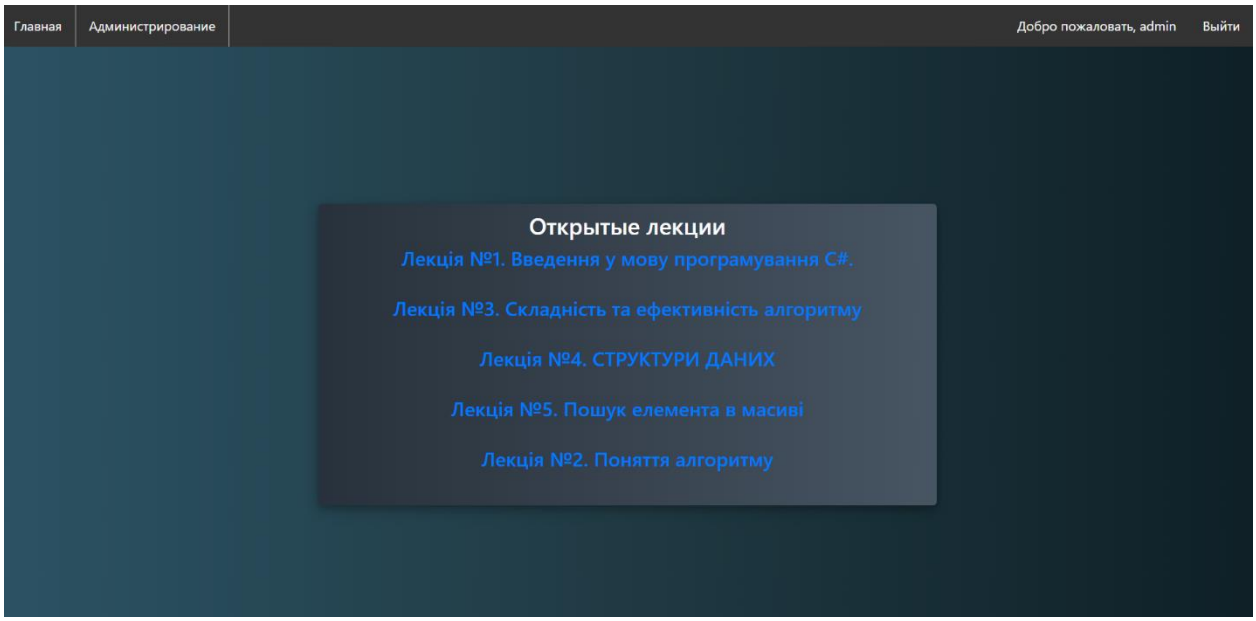


Рисунок 3.6 — Сторінка активних лекцій

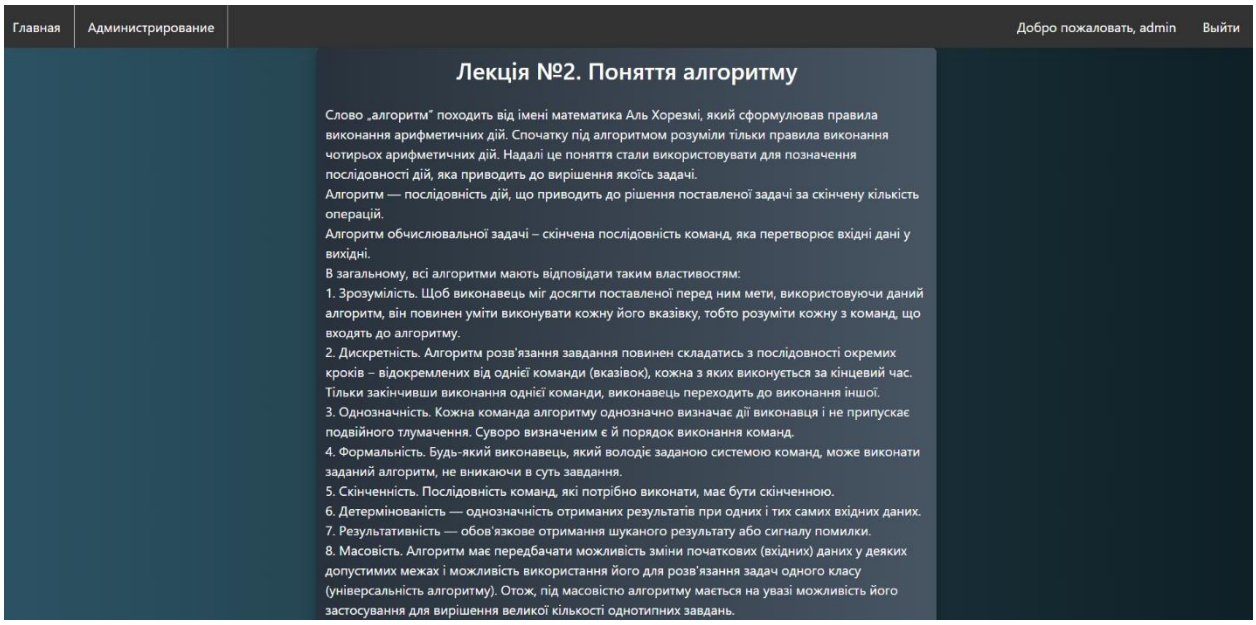


Рисунок 3.7 — Сторінка лекції

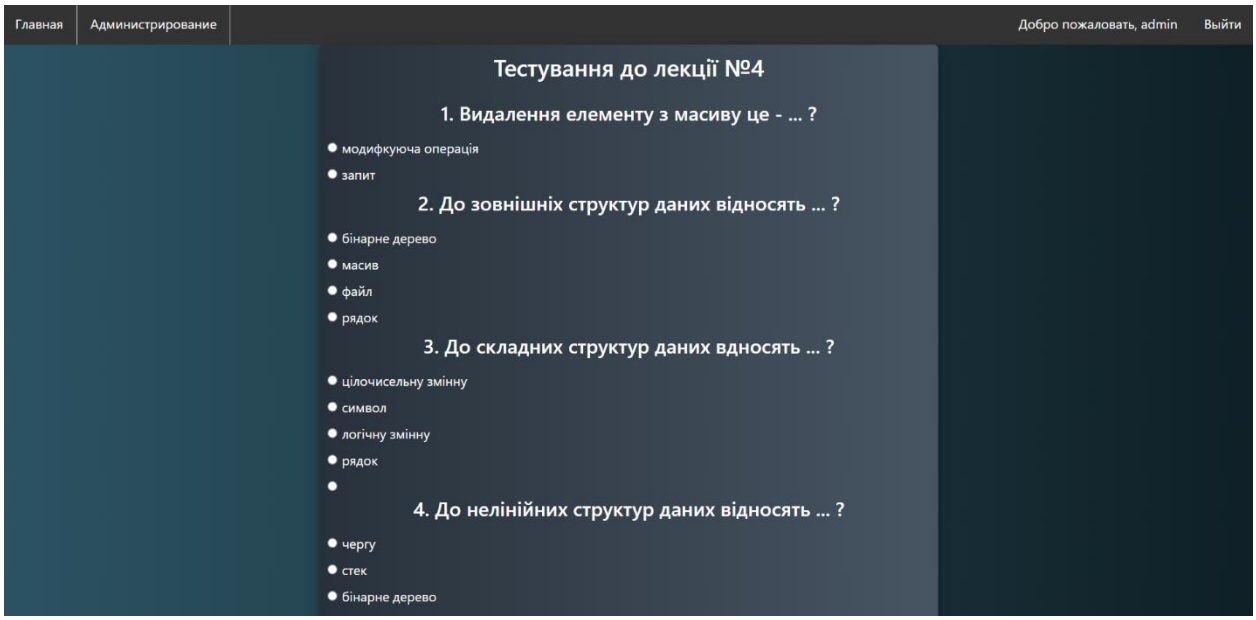


Рисунок 3.8 — Сторінка тестування

3.4 Тестування

Для тестування розробленого програмного продукту було обрано методику чорного ящика. Тестування чорного ящика — це метод тестування програмного забезпечення, який перевіряє функціональність програми, не заглядаючи в її внутрішні структури чи роботи. Цей метод тестування можна застосувати практично на кожному рівні тестування програмного забезпечення: одиничному, інтеграційному, системному та приймальному. Його іноді називають тестуванням на основі специфікації.

Спеціальні знання коду програми, внутрішньої структури та знання програмування загалом не потрібні. Тестувальник знає, що програмне забезпечення має робити, але не знає, як воно це робить. Наприклад, тестувальник знає, що конкретні вхідні дані повертають певний незмінний вихід, але не знає, як програмне забезпечення виробляє вихідні дані в першу чергу.

Таблиця 4.2 — Тестування додатку

№	Тест-кейс	Очікуваний результат	Отриманий результат
1	Невірні данні при авторизації	При введенні невірних даних система повідомляє користувача про те, що такого користувача не існує, або дані введені невірно	При введенні невірних даних система повідомляє користувача про те, що такого користувача не існує, або дані введені невірно
2	Пусті поля при авторизації	При спробі авторизації з пустими полями система повідомляє користувача про те,	При спробі авторизації з пустими полями система повідомляє користувача про те,

		що ці поля необхідно заповнити.	що ці поля необхідно заповнити.
3	Неспівпадаючі паролі при реєстрації	При введенні неспівпадаючих паролів система повідомляє користувача про те, що паролі не співпадають.	При введенні неспівпадаючих паролів система повідомляє користувача про те, що паролі не співпадають.
4	Пусті поля при реєстрації	При спробі реєстрації з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.	При спробі реєстрації з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.
5	Створення опитування з пустими полями	При спробі створення опитування з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.	При спробі створення опитування з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.
6	Створення пустого опитування	При спробі створення опитування з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.	При спробі створення опитування з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.

7	Спроба перейти в панель адміністратора без наявності прав адміністратора	При спробі користувача без прав адміністратора перейти в панель адміністрування система повідомляє користувача, шо він не є адміністратором і не може перейти на дану сторінку.	При спробі користувача без прав адміністратора перейти в панель адміністрування система повідомляє користувача, шо він не є адміністратором і не може перейти на дану сторінку.

Результати тестування показують, що система повністю функціональна і готова до використання в реальних умовах.

ВИСНОВКИ

Метою даної роботи було створення електронного навчального ресурсу з розділу «Структури даних та алгоритми їх обробки» дисципліни «Теоретичні основи програмування» та його наповнення навчальним контентом.

У ході роботи над даним проектом були виконані наступні завдання:

- здійснено пошук інформації відповідно до тематики магістерської роботи;
- проаналізовано теоретичні підходи та стан наукової розробленості питання розробки електронних навчальних ресурсів;
- розроблено модель електронного навчального ресурсу;
- здійснено огляд можливого інструментарію розробки;
- з’ясовано особливості використання програмно-апаратних засобів електронних навчальних ресурсів;
- розроблено проект електронного навчального ресурсу з розділу “Структури даних та алгоритми їх обробки” дисципліни «Теоретичні основи програмування»;
- наповнено створений ресурс навчальними матеріалами.

Завдяки чіткому виконанню завдань, поставлених на початку роботи, в результаті виконання роботи було отримано повноцінну систему, що здатна виконувати закладений в неї функціонал та готова до використання в реальних умовах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Robinson, Rhonda; Molenda, Michael; Rezabek, Landra. "Facilitating Learning" (PDF). Association for Educational Communications and Technology. Retrieved 18 March 2016.
2. "What's Better in the Classroom—Teacher or Machine?". Wall Street Journal.
3. "To win post-pandemic, edtech needs to start thinking big". TechCrunch.
4. Draft National Education Policy 2019 by Govt. of India (2018)
5. Richey, R.C. (2008). "Reflections on the 2008 AECT Definitions of the Field". TechTrends. 52 (1): 24–25. doi:10.1007/s11528-008-0108-2. S2CID 189912472.
6. D. Randy Garrison; Terry Anderson; Definitions and Terminology Committee (2003). E-Learning in the 21st Century: A Framework for Research and Practice. Routledge. ISBN 978-0-415-26346-7.
7. Al Januszewski A.; Molenda Michael. (2007) Educational Technology: A Definition with Commentary ISBN 978-0805858617
8. Lowenthal, P. R.; Wilson, B. G. (2010). "Labels do matter! A critique of AECT's redefinition of the field". TechTrends. 54 (1): 38–46. CiteSeerX 10.1.1.408.648. doi:10.1007/s11528-009-0362-y. S2CID 143977728.
9. Report by Tech.Ed.Gov (2017). NETP17.
10. "Technology in Education: An Overview - Education Week". www.edweek.org. Retrieved 2016-10-31.
11. Seels, B. B., & Richey, R. C. (1994). Instructional technology: The definition and domains of the field. Washington, DC: AECT.
12. Geng, F. (2014). Confusing terminologies: #e-learning, learning technologist, educational technologist,…discussed by @A_L_T members. Oxford, UK. https://blogs.it.ox.ac.uk/fawei/2014/07/29/confusing-terminologies-e-learning-learning-technologist-educational-technologistdiscussed-by-a_l_t-members/

13. Selwyn, N. (2011) Education and Technology: Key Issues and Debates. London: Continuum International Publishing Group.
14. Day, R; Payne, L (1987). "Computer-managed instruction: an alternative teaching strategy". J Nurs Educ. 26 (1): 30–6. PMID 3029349.
15. Moore, J. L.; Dickson-Deane, C.; Galyen, K. (2011). "E-Learning, online learning, and distance learning environments: Are they the same?". The Internet and Higher Education. 14 (2): 129–135. doi:10.1016/j.iheduc.2010.10.001.
16. "Universities Use Second Life to Teach Complex Concepts". Government Technology. Retrieved 2013-10-03.
17. "DoD gives PTSD help 'second life' in virtual reality | Article | The United States Army". Army.mil. Retrieved 2013-10-22.
18. Kurbel, Karl: Virtuality on the Students' and on the Teachers' sides: A Multimedia and Internet based International Master Program; ICEF Berlin GmbH (Eds.), Proceedings on the 7th International Conference on Technology Supported Learning and Training – Online Educa; Berlin, Germany; November 2001, pp. 133–136
19. J. Bransford; A. Brown; R. R. Cocking, eds. (2000). "Technology to support learning". How people learn: Brain, mind, experience. Washington, DC: National Academies Press. pp. 206–230.
20. Alsheail, Abdulrahman (2010). Teaching English as a Second/Foreign Language in a Ubiquitous Learning Environment: A Guide for ESL/EFL Instructors (PDF). (Master's Project). Archived from the original (PDF) on 2014-02-07. Retrieved 2016-04-02.
21. Hwang, G. J. (2014). Definition, framework and research issues of smart learning environments-a context-aware ubiquitous learning perspective. Smart Learning Environments, 1(1), 1-14.
22. Kinshuk; Chen, Nian-Shing; Cheng, I-Ling; Chew, Sie Wai (17 February 2016). "Evolution Is not enough: Revolutionizing Current Learning Environments to Smart Learning Environments". International Journal of

Artificial Intelligence in Education. 26 (2): 561–581. doi:10.1007/s40593-016-0108-x. S2CID 11084070.

23. Spector, Jonathan Michael (16 October 2014). "Conceptualizing the emerging field of smart learning environments". Smart Learning Environments. 1 (1). doi:10.1186/s40561-014-0002-7. S2CID 3745158.
24. Andone, Diana; Holotescu, Carmen; Grosseck, Gabriela (26 November 2014). 2014 International Conference on Web and Open Access to Learning (ICWOAL). pp. 1–4. doi:10.1109/ICWOAL.2014.7009244. ISBN 978-1-4799-5739-2. S2CID 15404201.
25. Lombardi, Patrizia; Giordano, Silvia; Farouh, Hend; Yousef, Wael (June 2012). "Modelling the smart city performance". Innovation: The European Journal of Social Science Research. 25 (2): 137–149. doi:10.1080/13511610.2012.660325. S2CID 155017799.
26. Molenda, M. (2008). "Historical foundations". In M. J. Spector, M. D. Merrill, J. Merrienboer, & M. P. Driscoll (Eds.), Handbook of Research on Educational Communications and Technology (Third., pp. 3–20). New York, NY: Lawrence Earlbaum Associates.
27. Nye, D. (2007). Technology Matters: Questions to Live With. Cambridge MA: MIT Press.
28. Biruni, Muhammad ibn Ahmad; Sachau, Eduard (1910). Alberuni's India. An account of the religion, philosophy, literature, geography, chronology, astronomy, customs, laws and astrology of India about A.D. 1030. London: K. Paul, Trench, Trübner & Co.
29. Saettler, P. (1990). The evolution of American educational technology. Englewood, CO: Libraries Unlimited.
30. Suppes, P.; Jerman, M.; Groen, G. (1966). "Arithmetic drills and review on a computer-based teletype" (PDF). The Arithmetic Teacher. 13 (4): 303–309. doi:10.5951/AT.13.4.0303. Archived from the original (PDF) on 2016-03-05. Retrieved 2015-09-04.

31. Suppes, P. (May 19, 1971). Computer Assisted Instruction at Stanford (PDF) (Report). Archived from the original (PDF) on July 17, 2010. Retrieved September 4, 2015.
32. "Promises and pitfalls of online education". 2017-06-09.
33. "Archived copy". Archived from the original on 2018-03-19. Retrieved 2018-03-19.
34. Hiltz, S. (1990). "Evaluating the Virtual Classroom". In Harasim, L. (ed.) Online Education: Perspectives on a New Environment. New York: Praeger, pp. 133–169.
35. Mason. R. and Kaye, A. (1989). Mindweave: Communication, Computers and Distance Education. Oxford, UK: Pergamon Press.

ДОДАТКИ

Додаток А Лістинг програмного коду

```
from django.urls import path

from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('register/', views.regist, name='register'),
    path('login/', views.loginPage, name='login'),
    path('logout/', views.logoutUser, name='logout'),
    path('questionnaire_page/<str:name>', views.questionnaire_page, name='questionnaire_page'),
    path('lection_page/<str:lec_id>', views.lection_page, name='lection_page'),
]
from django.db import models
from django.contrib.auth.models import User


class questionnaireData(models.Model):
    questionnaire_name = models.TextField('Название опросника')
    questions_text = models.TextField('Вопросы')
    questions_answers = models.TextField('Ответы')

    def __str__(self):
        return self.questionnaire_name
```

```
        class Meta:
            verbose_name = 'Опрос'
            verbose_name_plural = 'Опросы'


    class questionnaireAnswers(models.Model):
        questionnaire_name_fk = models.ForeignKey(questionnaireData, on_delete=models.CASCADE, verbose_name='Опрос')
        user = models.ForeignKey(User, on_delete=models.CASCADE, verbose_name='Пользователь')
        answers = models.TextField('Ответы')

        def __str__(self):
            return self.user.username + "/" + self.questionnaire_name_fk.questionnaire_name

        class Meta:
            verbose_name = 'Ответ'
            verbose_name_plural = 'Ответы'


    class lection(models.Model):
        lection_name = models.TextField(verbose_name='Название лекции')
        lection_text = models.TextField(verbose_name='Текст лекции')
        questionnaire_fk = models.ForeignKey(questionnaireData, on_delete=models.CASCADE, verbose_name='Опрос')

        def __str__(self):
            return self.lection_name
```

```
    class Meta:
        verbose_name = 'Лекция'
        verbose_name_plural = 'Лекции'
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login, logout
from django.contrib import messages
from .forms import CreateUserForm
from .models import questionnaireAnswers, questionnaireData, lection
from django.contrib.auth.models import User


def loginPage(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        user = authenticate(request, username=username, password=password)

        if user is not None:
            login(request, user)
            return redirect('index')
        else:
            messages.info(request, 'Имя пользователя или пароль введены неверно')
    context = {}
    return render(request, 'login.html', context)


def regist(request):
    form = CreateUserForm
    if request.method == "POST":
```

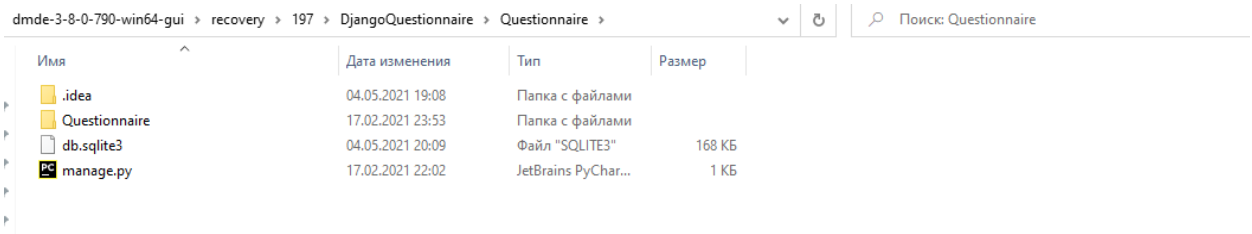
```
        form = CreateUserForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('login')
    context = {'form': form}
    return render(request, 'register.html', context)


def logoutUser(request):
    logout(request)
    return redirect('login')


def index(request):
    if request.user.is_authenticated:
        questionnaire_list = lection.objects.all()
        return render(request, 'index.html', {'questionnaire_list': questionnaire_list})
    else:
        return redirect('login')


def lection_page(request, lec_id):
    lection_l = lection.objects.filter(lection_name = lec_id)
    return render(request, 'lection_page.html', {'lection': lection_l[0]})


def questionnaire_page(request, name):
    questionnaireCurr = questionnaireData.objects.get(questionnaire_name=name)
```

```
    questName = questionnaireCurr.questionnaire_name
    questions = questionnaireCurr.questions_text.split('/')
    tempAns = questionnaireCurr.questions_answers.split('/')
    tempQA = []
    for i in range(len(questions)):
        tempQA.append(questions[i] + '*' + tempAns[i])

    quest_ans = []
    for i in range(len(tempQA)):
        quest_ans.append(tempQA[i].split('*'))

    answers = []
    if request.method == 'POST':
        for i in range(1, len(quest_ans) + 1):
            answers.append(request.POST.get('quest' + str(i)))
        addAnswersToDB(answers, questName, request)
        return redirect('index')
    return render(request, 'questionnaire_page.html', {'questName': questName, 'quest_ans': quest_ans})


def addAnswersToDB(answers, name, req):
    quest = questionnaireData.objects.get(questionnaire_name=name)
    strAnsw = ""
    for str in answers:
        strAnsw += str + "\n"
    answ = questionnaireAnswers(questionnaire_name_fk=quest, user=req.user, answers=strAnsw)
    answ.save()
```

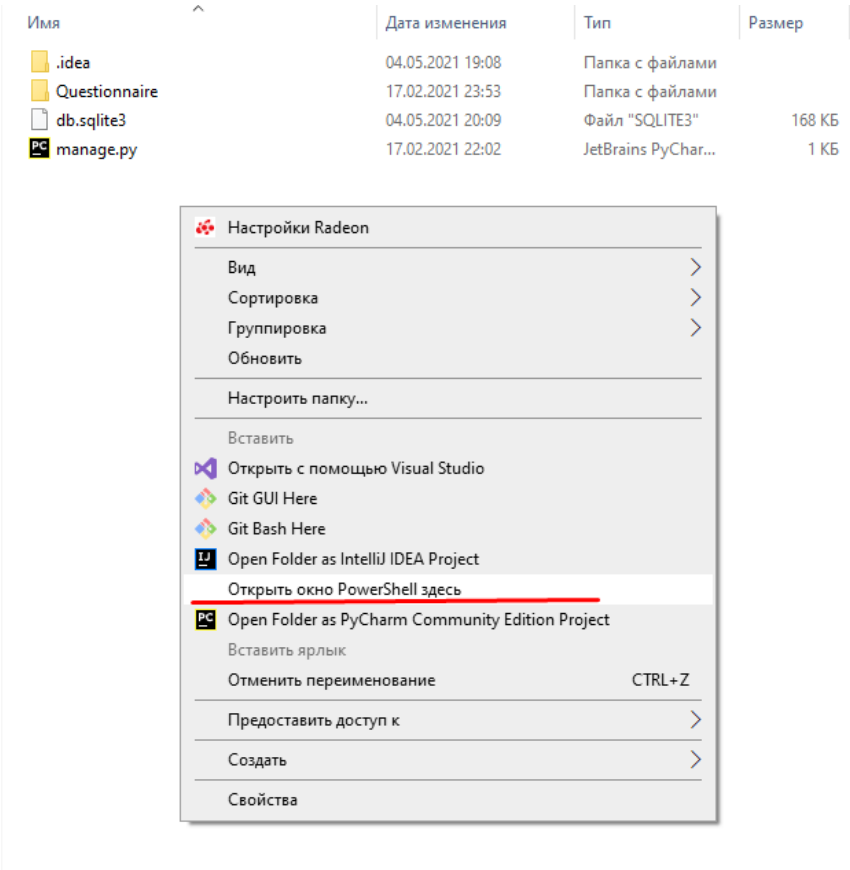
Додаток Б. Інструкція користувача

Для початку потрібно встановити Python. Для цього в Гугл завантажуємо з офіційного сайту останню версію програми і встановлюємо.

Далі відкриваємо папку, у якій лежить файл manage.py:



Затискаємо Shift та правою кнопкою клацаємо на порожнє місце в папці, відкриється меню, там обираємо *відкрити вікно Powershell тут*:



У консолі пишемо команду: pip install Django (це може не знадобитися, але про всяк випадок краще зробити). Коли джанго встановиться, пишемо в

тій же консолі наступну команду: python manage.py runserver. Після цього консоль виглядатиме так:

```
Windows PowerShell
PS C:\Users\Роман\Downloads\dmde-3-8-0-790-win64-gui\recovery\197\DjangoQuestionnaire\Questionnaire> python manage.py ru
nserver
Watching for file changes with StatReloader
Performing system checks...

System check identified some issues:

WARNINGS:
mainApp.children: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.d
b.models.AutoField'.
        HINT: Configure the DEFAULT_AUTO_FIELD setting or the MainappConfig.default_auto_field attribute to point to a s
ubclass of AutoField, e.g. 'django.db.models.BigAutoField'.
mainApp.marks: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.m
odels.AutoField'.
        HINT: Configure the DEFAULT_AUTO_FIELD setting or the MainappConfig.default_auto_field attribute to point to a s
ubclass of AutoField, e.g. 'django.db.models.BigAutoField'.

System check identified 2 issues (0 silenced).
May 04, 2021 - 20:20:33
Django version 3.2, using settings 'Questionnaire.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Консоль не закриваємо, вона відіграє роль сервера зараз. Просто згортаємо, у браузері переходимо за адресою, написаною в консолі: http://127.0.0.1:8000/.