Міністерство освіти і науки України
Державний ВНЗ «Національний гірничий університет»

Факультет інформаційних технологій
(факультет)

Кафедра програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
дипломної роботи

магістра
(назва освітньо-кваліфікаційного рівня)

галузь знань *12 Інформаційні технології*
(шифр і назва галузі знань)

спеціальність *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

спеціалізація *Програмне забезпечення систем*
(код і назва спеціалізації)

освітній рівень *магістр*
(назва освітнього рівня)

кваліфікація *2132.2 Інженер програміст*
(назва кваліфікації)

на тему: *Дослідження ефективності використання мов програмування орієнтованих на web в залежності від вирішуваної задачі*

Виконавець:
студент *6* **курсу , групи** *121М-16-1*

(підпис) Барвіненко Д.М.
(прізвище та ініціали)

Керівники	Посада, прізвище, ініціали	Оцінка	Підпис
проекту	*проф. Куваєв В.М.*		
розділів:			
Економічний	*доц. Касьяненко Л.В.*		

Рецензент			

Нормоконтроль	*доц. Коротенко Л.М.*		

Дніпро
2018

Міністерство освіти і науки України
Державний вищий навчальний заклад
«Національний гірничий університет»

ЗАТВЕРДЖЕНО:
завідувач кафедри

програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 20 ____ року

ЗАВДАННЯ
на виконання кваліфікаційної роботи магістра

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

студенту *121М-16-1* *Барвіненко Д.М.*
(група) (прізвище та ініціали)

Тема дипломної роботи Дослідження ефективності використання мов програмування, орієнтованих на web в залежності від вирішуваної задачі

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора Державного ВНЗ «НГУ» від *26.12.2017 р.* № *2127-л*

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – мови програмування орієнтовані на веб.

Предмет досліджень – ефективність використання мов програмування, які мають веб орієнтоване направлення.

Мета НДР – проаналізувати мови програмування, які орієнтовані на веб для підготовлення самих необхідних характеристик, які можуть бути важливими для розробників при вибої мови на якій буде реалізовано ту чи іншу задачу.

Вихідні дані для проведення роботи – ананалізуючі скрипти на різних мовах програмування.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна результатів, що очікуються, полягає у: дослідженні основних характеристик мов програмування орієнтованих на веб в залежності від задачі, яку вони вирішують та обґрунтування рекомендацій що до вибору мови програмування в

залежності від типу задачі, яка програмується.

Практична цінність результатів полягає у: скороченні часу на аналіз мови програмування, при виборі технологій для вирішення завдань на програмування.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок-кінець**
Розділ 1	2018.01.01 - 2018.01.10
Розділ 2	2018.01.11 - 2018.01.15
Розділ 3	2018.01.16 - 2018.01.20
Економіка	2018.01.21 - 2018.01.23

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект від реалізації результатів роботи не очікується, так як скрипти не будуть розповсюджуватись на грошовій основі.

Соціальний ефект від реалізації результатів роботи очікується позитивним, завдяки можливості наочно порівняти мови програмування та вирішити раз і назавжди, яка краще.

7 ДОДАТКОВІ ВИМОГИ

Завдання видав _____ *Куваєв В.М.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Барвіненко Д.М.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: 26.12.201_

Термін подання дипломного проекту до ДЕК 24.01.2018_

Реферат

Пояснювальна записка: 96 с., 12 рис., 3 дод., 47 джерела.

Об'єкт дослідження: мови програмування орієнтовані на розробку для веб середовища.

Мета магістерської роботи: дослідити ефективність використання мов програмування в залежності від поставленої задачі.

Методи дослідження. При вирішенні поставленого завдання використовувалися математичні алгоритми та наукові досягнення в областях розробки інформаційних систем і програмного забезпечення.

Наукова новизна. Новизна отриманих результатів полягає в проведенні аналізу та виявленні недоліків і позитивних якостей мов програмування.

Практичне значення. Полягає в приведені результатів аналізу основних характеристик мов програмуванн, які можуть знадобитися для вирішення застосовуввати мову програмування для поставленої задачі.

Галузь застосування. В якості галузі застосування предстає веб середовище.

Значення роботи і висновки. Значення роботи відображуеться в порівняльних звітах проведеного аналізу мов програмування орієнтованих на веб середовище.

Прогнози щодо розвитку досліджень. Розробити універсальні програмні модулі, які можуть бути використані для порівняння мов програмування. Розробити комплекс програмних засобів і максимально розширюваний інструмент тестування всіх можливих аспектів ефективності мов програмування.

Економіка. У розділі «Економіка» проведено розрахунки трудомісткості розробки програмного забезпечення, витрат на створення ПО і тривалості його розробки, а також проведені маркетингові дослідження ринку збуту створеного програмного продукту.

Список ключових слів: Python, JS, JavaScript, Ruby, WebWorkers, API, Tests, Unit, ActiveRecord, HTTP, HTTPS, TCP, IP, MySQL, PostgresSQL.

Abstract

Explanatory note: 96 pp., 12 pp., 3 pp., 47 pp.

Object of research: programming languages are development-oriented for the web environment.

The purpose of the master's work: to study the efficiency of using programming languages depending on the task.

Research methods. In solving the problem, mathematical algorithms and scientific achievements in the areas of information systems and software development were used.

Scientific novelty. The novelty of the results obtained is to analyze and identify the disadvantages and positive qualities of programming languages.

Practical meaning. It is in the results of the analysis of the main characteristics of the programming languages that may be required for the decision to use the programming language for the task.

Field of application. As a field of application, the web environment appears.

Value of work and conclusions. The value of the work is reflected in the comparative reports of the analysis of programming languages for the web-based environment.

Projections for research development. Develop universal program modules that can be used to compare programming languages. Develop a suite of software tools and the most extensible tool for testing all possible aspects of the effectiveness of programming languages.

Economy. In the section "Economics" the calculations of the complexity of software development, the cost of software creation and the length of its development, as well as marketing researches of the market of the created software product were carried out.

Keyword list: Python, JS, JavaScript, Ruby, WebWorkers, API, Tests, Unit, ActiveRecord, HTTP, HTTPS, TCP, IP, MySQL, PostgresSQL.

Зміст

ВСТУП

Останнім часом все частіше можна почути про збільшення масштабів розробки програмного забезпечення орієнтованого на веб. Веб індустрія займає велику долю ринку порівняно з іншими напрямками. Це багатомільярдні компанії, що займаються розробкою розважальних продуктів та програмних забезпечень, що складаються з мільйонів рядків програмного коду і файлів мультимедіа.

Цьому сприяло масштабне зростання числа користувачів інтернету. Веб-додатки стали складніші і масштабніші. Сучасні веб-додатки - великі і складні програмні комплекси. Витрати тільки на програмну розробку часто вимірюються сотнями людино-місяців. За обсягом задіяних технологій, залучених коштів і рівня професійної підготовки розробників, веб індустрія давно зайняла аж ніяк не останнє місце в світі IT.

Спочатку інтернету вся інформація була тільки в текстовому вигляді. Разом з збільшенням швидкості інтернет з’єднання виростають і можливості користувачів. Зараз в найпопулярніших сайтах української частини мереж зареєстрована велика кількість людей це наприклад - є rozetka (10 млн користувачів) і olx (20 млн). Світовий лідер - Facebook (більше 500 млн користувачів по всьому світу). Facebook дуже активно розвивається на нашому ринку, і зараз в мережі зареєстровано 3,7 млн співвітчизників. Інший популярний в світі сервіс Twitter (мікроблоги) в Україні зібрав вже понад 600 тис. користувачів.

Саме через таку велику кількість людей і виникає потреба в ще більшій кількості професійних програмістів, які змогли би впоратися з вимогами. А для таких професіоналів гостро стоїть питання вибору технологій, мов програмування, алгоритмів та засобів вирушення цих завдань.

З збільшення кількості людей ростуть і їх потреби, такі гіганти веб напрямку, як Facebook, Instagram, Twitter вводять штучний інтелект для своїх стрічок новин. Починають використовуватись такі шаблони програмування, як

мікросервіси, маркетплейс платформи застосовують в своїй роботі блокчейн. В такому різноманітності потреб та вимог стає дуже важливим створення об’ємного та обґрунтованого технічно порівняння мов та інструментів, які б максимально швидко вирішили поставлені перед ними завдання.

Тому існує досить велика кількість мов програмування орієнтованих на розробку web-додатків і, очевидно, що за процесом активізація їх використання ховається все більш складний етап порівняння та виявлення більш підходящого до вирішуваної задачі. Головна причина цього явища - відсутність єдиного механізму порівняння. Технічним керівникам команд доводиться витрачати значну кількість годин для виявлення, порівняння, та тестування, мов програмування які підходять для поставлених завдань. А згодом, в разі переходу до іншої задачі, доводиться, по суті, все повторювати заново, знову витрачати час та нерви. На великих підприємствах нерідко виникає необхідність впровадження і одночасного використання декількох мов програмування, а поверхневе порівняння яке може провести технічний лідер не дасть достатньо об’ємного результату. В цьому випадку необхідні витрати на висококваліфікованих програмістів можуть легко вийти за рамки допустимих, оскільки фахівці, які володіють одночасно декількома мовами програмування, зустрічаються досить рідко, а їх «ціна» росте аж ніяк не пропорційно кількості виконаних завдань. Перерахування подібних ситуацій можна продовжити, але вже зі сказаного, очевидно, що необхідно якимось чином вирішувати проблеми систематизації, профілювання та порівняння, щоб уніфікувати всі наявні нині мови розробки програмного забезпечення.

На даний час існує багато порівняльних графіків, діаграм та текстів, але ні один з них не описує ефективність тої чи іншої мови в певній задачі. Часто мови програмування порівнюють тільки за їх популярністю за кількістю цитування цих мов в сучасних засобах Інтернет мовлення та кореспонденції. Більшість використовує це як індикатор для вибору тої чи іншої мови. Сама ця ідея є непослідовною і навіть дивною. Таким чином виходить, що старі мови

типу Сі, які просто через вельми поважний вік свого існування отримали більше всіх документацій та посилань, як приведено в даній дипломній роботі – це не є досить об’ємним фактором при виборі технології для реалізації сучасних запитів та задач користувачів розроблюваних програмних додатків.

Необхідність в виявленні параметрів за якими можна порівняти мови саме як мови, а не платформи стала дуже гостро. Популярність, інструменти розробки, позиція на ринку, максимальні заробітні плати - це важливо, але важливо саме подивитися на мови самі по собі.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАННЯ

1.1 Постановка задачі

Згідно з теми дипломного проекту, потрібно проаналізувати ефективність мов програмування орієнтованих на веб в залежності від задачі.

Потрібно проаналізувати технічні параметри роботи мов програмування такі, як – швидкодія, споживання оперативної пам’яті, навантаження на центральний процесор. Їх можна виміряти за допомогою математичних алгоритмів та за допомогою моделювання різних задач, які виникають при розробці під веб.

Наприклад:

- розробка сайтів візиток;
- розробка більш складних сайтів – магазинів, новинних, корпоративных і т.д.;
- розробка сайтів для налізу великих об’ємів даних;
- розробка сайтів на основі блокчейн підходу;
- розробка сайтів для математичних операцій.

Також слід брати до уваги швидкість розробки програм, кільки нових версій, які вийшли за останній час, кількість відомих складнощів з безпекою.

1.2 Історія Інтернету

Інтернет – всесвітня система сполучених комп'ютерних мереж, що базуються на комплекті Інтернет-протоколів. Інтернет також називають мережею мереж. Інтернет складається з мільйонів локальних і глобальних приватних, публічних, академічних, ділових і урядових мереж, пов'язаних між собою з використанням різноманітних дротових, оптичних і бездротових технологій. Інтернет становить фізичну основу для розміщення величезної кількості інформаційних ресурсів і послуг, таких як взаємопов'язані гіпертекстові документи та електронна пошта.

Інтернет не має централізованого управління, правил використання чи доступу. Кожна складова мережа встановлює свої власні стандарти. Централізовано визначаються правила використання адресного простору Інтернет-протоколу та системи доменних імен. Мережа побудована на використанні протоколу IP і маршрутизації пакетів даних.

Перший сайт з'явився в онлайні 6 серпня 1991 року. На цьому сайті була опублікована концепція технології WWW (World Wide Web), що використовує протокол передачі даних HTTP (HyperText Transfer Protocol) за допомогою системи адресації URI (Uniform Resource Identifier) за допомогою написання коду на мові гіпертекстової розмітки HTML (HyperText Markup Language).

Даний сайт мав ім'я info.cern.ch. Автором і власником сайту був Тім Бернерс-Лі. На даному сайті також була розміщена інформація по принципам установки, настройки і роботи серверів і браузерів. Крім того, сайт став першим в історії інтернет-каталогом, тому що автор розмістив на його сторінках перший список інших сайтів.

Природно, що матеріали та інструменти, необхідні для роботи першого сайту були підготовлені заздалегідь. До грудня 1990 року було розроблено гіпертекстовий браузер з функціями веб-редагування під назвою WorldWideWeb, сервер на базі NeXTcube і, власне, самі сторінки, які сервер видавав браузеру.

Тім Бернерс-Лі був упевнений, що гіпертекст може служити основою для мереж обміну даними. Своє перше дітище - гіпертекстове програмне забезпечення Enquire він створив за 10 років до створення свого першого сайту, в 1980 році.

У травні 1991 року в Європейському Центрі Ядерних Досліджень в Женеві (CERN) (в якому працював і презентував свій перший сайт Бернерс-Лі) був затверджений стандарт WWW.

До 1993 року були остаточно сформовані специфікації URI, HTTP і HTML. 30 квітня 1993 року CERN оголосила, що World Wide Web буде вільною

і безкоштовною для всіх. Це був дуже важливий крок, адже CERN мав повне право використовувати розробку в комерційних цілях. В такому випадку, сьогоднішнього Інтернет просто не існувало б.

13 березня 1989 року Тім Бернерс-Лі звернувся до свого керівника Майку Сендаллу з пропозицією створити систему управління інформацією.

Даний лист фактично став стартовою точкою початку створення системи, що пізніше отримала назву World Wide Web - «Всесвітня павутина» або Інтернет.

Слід зазначити, що теоретичні передумови для створення гіпертексту були закладені ще за півстоліття до створення першого сайту. На початку 1940-х років Ванневар Буш висунув ідею про можливість розширення людської пам'яті за допомогою технічних пристроїв. А також (що більш важливо, з позиції технології веб) можливість індексації накопиченої людством інформації для швидкого пошуку. Далі Даг Енгельбарт і Теодор Нельсон запропонували ідею «ветвящегося текста» - власне, технологію гіпертексту. Технологія передбачала можливість надання читачеві різні варіанти читання. Свою систему для зберігання і пошуку тексту автори назвали Xanadu. Система так і залишилася незавершеною.

1.3 Класифікація веб сайтів

Сайт візитка:

Сайт візитка - найпростіший вид сайту. Сайт такого типу можна зробити навіть на простому HTML, без використання системи управління сайтом. Зазвичай сайт-візитка містить від 1 до 5 сторінок. Сайти цього виду як правило включають в себе тільки загальну інформацію про власника сайту і його контактні дані. Простота розробки такого виду сайту робить вартість його створення порівняно дешевої, що є очевидною перевагою для замовника.

Корпоративні сайти:

Корпоративні сайти - це повнофункціональні представництва компаній в інтернеті. Цей тип сайту найкраще підходить для серйозних середніх і великих

фірм. Корпоративні сайти містять повну інформацію про компанію та її діяльність. Такий тип сайту іноді називають віртуальним офісом, так як відвідування такого сайту можна порівняти з спілкуванням з менеджером по роботі з клієнтами. Корпоративні сайти потрібні, в першу чергу, для формування іміджу компанії і надання відвідувачам і клієнтам якнайповнішої інформації.

Інтернет-вітрини:

Інтернет-вітрина або інтернет-каталог товарів - це вид сайтів, основне завдання яких - продавати. На таких сайтах розміщується інформація про товари і контакти, зазвичай телефони, за якими слід дзвонити бажаючим придбати пропонований товар. На таких сайтах розміщуються технічні характеристики товарів, відгуки, рекомендації експертів і т.д.

Інтернет магазини:

Цей вид сайтів аналогічний інтернет-вітрин, але має додатковий функціонал: можливість замовити пропонований товар прямо через сайт.

Промо-сайти:

Сайти цього типу призначені для розкрутки і просування будь-якого товару або бренду.

Тематичні сайти:

Даний тип інтернет сайтів характеризується тим, що містить інформацію з будь-якої конкретної тематики. Сюди ж можна віднести інтернет-енциклопедії.

Інтернет-портали:

Портали - це тип сайтів, що містять велику кількість різноманітної інформації. Як правило, портали схожі за структурою з тематичними сайтами, але мають більш розвинений функціонал і більшу кількість сервісів і розділів. Також на порталах часто бувають розділи для спілкування користувачів: чати,

блоги і форуми.

Блоги:

Блог - це тип сайтів, на яких власник або редактор блогу пише пости зі своїми новинами, ідеями або інший постійно надходить інформацією. Відмінною особливістю блогів є актуальність інформації, що публікується інформації.

Каталоги сайтів:

Це вид сайтів, основним вмістом яких є структуровані посилання на інші сайти, а також їх короткі описи.

Пошукові системи:

Вид сайтів, призначених для пошуку сторінок в інтернеті по певних запитах.

Поштові сервіси:

Цей тип сайтів надає інтерфейс для роботи з електронною поштою.

Інтернет-форуми:

На сайтах цього виду користувачі можуть створювати теми, а також коментувати їх. Як правило, форуми обмежені однією специфічною тематикою, хоча зустрічаються і форуми «про все».

Сайти-хостинги:

На сайтах цього типу реалізована функція зберігання будь-яких файлів. Також часто зустрічаються сайти-хостинги з можливістю перегляду завантажених файлів прямо через браузер.

Дошки оголошень:

На таких сайтах користувачі можуть розміщувати або шукати інформацію у вигляді будь-яких оголошень, наприклад - про купівлю-продаж.

Соціальні мережі:

Тип сайтів, створених для спілкування користувачів між собою. Як правило, на таких сайтах є рейтинги, сторінки користувачів, групи і безліч інших сервісів.

1.4 Досліджувані мови програмування

Для дослідження обрано чотири мови програмування: Python, Ruby, Javascript, PHP. Ці мови э самими популярними мовами для розробки в веб середовищі за даними сайту Github.com.

Python – інтерпретуєма об'єктно-орієнтована мова програмування високого рівня з динамічною типізацією, автоматичним управлінням пам'яттю і зручними високорівневими структурами даних, такими як словники (хеш-таблиці), списки, кортежі. Підтримує класи, модулі (які можуть бути об'єднані в пакети), обробку винятків, а також багатопотокові обчислення. Пітон має простий і виразним синтаксисом. Мова підтримує кілька парадигм програмування: структурний, об'єктно-орієнтоване, функціональне і аспектно-орієнтоване.

Розробка мови Python була почата в кінці 1980-х років співробітником голландського інституту CWI Гвідо ван Россум. Для розподіленої ОС Amoeba потрібний розширюваний скриптова мова, і Гвідо почав писати Python на дозвіллі, запозичивши деякі напрацювання для мови ABC (Гвідо брав участь в розробці цієї мови, орієнтованого на навчання програмування). У лютому 1991 року Гвідо опублікував вихідний текст в групі новин alt.sources. З самого початку Python проектувався як об'єктно-орієнтована мова.

Назва мови пішла не від виду плазунів. Автор назвав мову на честь популярного британського комедійного телешоу 1970-х «Літаючий цирк Монті Пайтона». Втім, все одно назву мови частіше пов'язують саме зі змією, ніж з передачею - піктограми файлів в KDE або в Microsoft Windows і навіть емблема на сайті python.org (до виходу версії 2.5) зображують зміїні голови. Важлива мета розробників Python - створювати його забавним для використання. Це відображено в його назві, яке прийшло з Монті Пайтона.

Наявність дружелюбного, чуйного спільноти користувачів вважається поряд з дизайнерської інтуїцією Гвідо одним з факторів успіху Python. Розвиток мови відбувається згідно чітко регламентованим процесу створення, обговорення, відбору та реалізації документів PEP (англ. Python Enhancement Proposal) - пропозицій щодо розвитку Python.

3 грудня 2008, після тривалого тестування, вийшла перша версія Python 3000. В Python 3000 усунені багато недоліків архітектури з максимально можливим (але не повним) збереженням сумісності зі старими версіями Python. На сьогодні підтримуються обидві гілки розвитку (Python 3.x і 2.x).

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

В області веб-програмування, зокрема серверної частини, PHP - один з популярних сценарних мов Популярність в області побудови веб-сайтів визначається наявністю великого набору вбудованих засобів для розробки веб-додатків.

Основні з них:

- автоматичне вилучення POST і GET-параметрів, а також змінних оточення веб-сервера;
- взаємодія з великою кількістю різних систем управління базами даних (MySQL, MySQLi, SQLite, PostgreSQL);
- автоматизована відправка HTTP-заголовків;
- робота з HTTP-авторизацією;
- робота з cookies і сесіями;
- робота з локальними і віддаленими файлами, сокетами;
- обробка файлів, що завантажуються на сервер;
- робота з XForms.

В даний час PHP використовується сотнями тисяч розробників. Згідно з

рейтингом корпорації TIOBE, що базується на даних пошукових систем, в травні 2016 року PHP знаходився на 6 місці серед мов програмування. До найбільших сайтів, які використовують PHP, відносяться Facebook, Wikipedia та ін. Входить в LAMP - поширений набір програмного забезпечення для створення та хостингу веб-сайтів (Linux, Apache, MySQL, PHP).

У 1994 році данський програміст Расмус Лердорф створив набір скриптів на Perl / CGI для висновку і обліку відвідувачів його онлайн-резюме, що обробляє шаблони HTML-документів. Лердорф назвав набір Personal Home Page (Особиста Домашня Сторінка). Незабаром функціональності і швидкості Perl - інтерпретатора скриптів - перестало вистачати, і Лердорф розробив з використанням мови C новий інтерпретатор шаблонів PHP / FI (англ. Personal Home Page / Forms Interpreter - «персональна домашня сторінка / інтерпретатор форм»).

У 2014 році було проведено голосування, за результатами якого наступна версія отримала назву PHP 7. Вихід нової версії планувався в середині жовтня 2015 года. У березні 2015 року Zend представили інфографіку в якій описані основні нововведення PHP 7.

Нова версія грунтується на експериментальній гілки PHP, яка спочатку називалася phpng (PHP Next Generation - наступне покоління), і розроблялася з упором на збільшення продуктивності і зменшення споживання пам'яті. У новій версії додана можливість вказувати тип повертаються з функції даних, доданий контроль переданих типів для скалярних даних, а також нові оператори.

JavaScript – підтримує об'єктно-орієнтована, імперативний і функціональний стилі. Є реалізацією мови ECMAScript (стандарт ECMA-262).

JavaScript зазвичай використовується як вбудований мова для програмного доступу до об'єктів додатків. Найбільш широке застосування знаходить в браузерах як мова сценаріїв для додання інтерактивності веб-сторінок.

Основні архітектурні риси: динамічна типізація, слабка типізація, автоматичне керування пам'яттю, прототипне програмування, функції як об'єкти першого класу.

На JavaScript вплинули багато мов, при розробці була мета зробити мову схожим на Java, але при цьому легким для використання непрограмістів. Мовою JavaScript не володіє будь-яка компанія або організація, що відрізняє його від ряду мов програмування, використовуваних в веб-розробці.

Назва «JavaScript» є зареєстрованим товарним знаком компанії Oracle Corporation.

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

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

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

Ruby – динамічна, рефлективна, що інтерпретується високорівнева мова програмування. Мова має незалежної від операційної системи реалізацією багатопоточності, суворої динамічною типізацією, складальником сміття і багатьма іншими можливостями. За особливостями синтаксису він близький до мов Perl і Eiffel, по об'єктно-орієнтованого підходу - до Smalltalk. Також деякі риси мови взяті з Python, Lisp, Dylan і Клу.

Творець Ruby - Юкіхіро Мацумото (Matz) – цікавився мовами

програмування, ще будучи студентом, але ідея про розробку нової мови з'явилася пізніше. Ruby почав розроблятися 23 лютого 1993 року і вийшов у світ в 1995 році.

Метою розробки було створення «справжнього об'єктно-орієнтованого», легкого в розробці, інтерпретується мови програмування. З листа автора:

Ruby народився 23 лютого 1993 року. В той день я розмовляв зі своїм колегою про можливість існування об'єктно-орієнтованого сценарного мови.

В Японії Ruby став популярним з моменту появи першої загальнодоступної версії в 1995 році, проте наявність документації тільки на японській мові стримувало його подальше поширення. Лише в 1997 році з'явився опис Ruby англійською мовою, а в 1998 році відкрився форум «ruby-talk». Це поклало початок росту популярності мови в іншому світі. Видано кілька книг на різних мовах, в тому числі російською. Зараз Ruby входить в більшість дистрибутивів ОС Linux, поставляється разом з Mac OS X, інші користувачі операційних систем.

24 лютого 2014 виповнився 21 рік з моменту анонса мови програмування Ruby. Така подія розробники вирішили відзначити випуском патча для Ruby 2.1, який назвали Ruby 2.1.1.

1.5 Веб термінологія

В ході роботи використовуються слова, які можуть ускладнити розуміння матеріалу. Тому в даному розділі будуть описані основні терміни та визначення до них:

Адаптивний веб-дизайн - підхід до розробки сайтів, який має на увазі створення декількох окремих макетів сайту з фіксованими розмірами і набором особливих функціональних можливостей для різних категорій пристроїв і завантаження відповідного макета при кожному відвідуванні.

Браузер - програма для перегляду web-сторінок, один з основних інструментів користувача Інтернету. На сьогоднішній день найбільш поширений браузер Microsoft Internet Explorer (MSIE), що поставляється в

складі операційної системи MS Windows. Досить часто використовуються браузери Opera, Chrome і Firefox. Різні браузери можуть по-різному відображувати одні і ті ж веб-сторінки (особливо створені непрофесіоналами) і по-різному виконувати сценарії (скрипти) на них. Професійні веб-розробники завжди враховують особливості браузерів при HTML верстці сторінок, завдяки чому такі сторінки однаково добре відображуються і працюють у всіх найбільш поширених браузерах.

Адреса сайту (URL, Uniform Resource Locator) - унікальна адреса веб-сторінки або якогось іншого ресурсу в Інтернеті.

Логін - це частина реквізитів доступу до закритих даних. Логін майже завжди супроводжується паролем. У кожного користувача може бути багато логінів для доступу до різних даних. Вони можуть використовуватися для доступу до системи управління сайтом, електронної пошти та інших ресурсів.

Сервер / Server - Це потужний комп'ютер (цілодобово підключений до мережі Інтернет), на якому розміщується сайт (або сайти). Найчастіше клієнтові не потрібен свій сервер (це виключно дорого) і йому простіше і зручніше розміщувати свій сайт на спеціальному сервері хостинг-провайдера.

Блокчейн – вибудувана за певними правилами безперервна послідовна ланцюжок блоків (зв'язний список), що містять інформацію. Найчастіше копії ланцюжків блоків зберігаються і незалежно один від одного (надзвичайно паралельно) обробляються на безлічі різних комп'ютерів.

Мікросервіси – сучасне уявлення сервіс-орієнтованої архітектури (SOA), що використовується для створення розподілених програмних систем. Як і в SOA, модулі в архітектурі мікросервісов взаємодіють по мережі один з одним для виконання мети. Ще одна схожість в тому, що мікросервіси використовують протокол-незалежну технологію. Дана архітектура є першою реалізацією SOA, що з'явилася після впровадження DevOps, і вона поступово стає стандартом для безперервно розвиваються систем.

В архітектурі мікросервісов модулі повинні бути невеликими і протоколи

повинні бути легкими. Перевагою розподілу різних функцій системи в різні невеликі модулі є те, що це підсилює ступінь зачеплення і зменшує зв'язаність. Це дозволяє простіше додавати і змінювати функції в системі в будь-який час.

Властивості, характерні для архітектури мікросервісов:

- модулі можна легко замінити в будь-який час;

- модулі організовані навколо функцій, наприклад, призначений для користувача інтерфейс, логістика, виставлення рахунку і т. д;

- модулі можуть бути реалізовані з використанням різних мов програмування, баз даних, апаратних засобів і програмного забезпечення, в залежності від того, що підходить найкраще.

Термін «мікросервіси» існує вже більше десяти років. На конференції Cloud Computing Expo в 2005 Пітер Роджерс використовує його в своїй презентації. Тоді він використовував термін «мікро-веб-сервіс» для іменування компонентів програмного забезпечення. Юваль Леві мав схожу думку щодо мікросервісов, він так-же вважав, що саме мікросервіси є наступним етапом розвитку архітектури Microsoft. Він описав, як добре розроблена сервісна платформа застосовує основні архітектурні принципи веб і веб-служб разом з Unix-подібними, щоб забезпечити радикальну гнучкість і простоту, надаючи платформу для застосування сервіс-орієнтованої архітектури у всій середовищі додатків. У майстерні архітекторів програмного забезпечення, що проводиться недалеко від Венеції в 2011 році, використовувався термін «мікросервіс» для опис загального архітектурного стилю, який розвивали учасники останні роки. І, нарешті, в 2012 році, все та ж команда, що і в 2011, вирішила прийняти цей термін, як остаточний.

Філософія мікросервісов фактично копіює філософію UNIX, а саме «Роби щось одне і роби це добре». Суть полягає в наступному:

- сервіси – невеликі, виконують кожен свою, єдину функцію;

- організаційна культура включає в себе автоматизацію розробки і тестування, це знижує навантаження на управління;

- принципи культури і дизайну повинні включати в себе «обхід» колишніх помилок;

- кожен сервіс – еластичний, легко змінюється, елементарний і при цьому закінчений.

1.6 Опис математичних алгоритмів

Для дослідження було обрано математичні алгоритми, завдяки яким можна було б виміряти таки характеристики:

- Стек виклику;
- Швидкість виконання;
- Кількість операцій в секунду.

Числа Фібоначчі – елементи числової послідовності в якій перші два числа рівні або 1 і 1, або 0 і 1, а кожне наступне число дорівнює сумі двох попередніх чисел. Названі на честь середньовічного математика Леонардо Пізанського (відомого як Фібоначчі).

Послідовність Фібоначчі була добре відома в стародавній Індії, де вона застосовувалася в метричних науках (просодії, іншими словами - віршуванні) набагато раніше, ніж стала відома в Європі.

Зразок довжиною n може бути побудований шляхом додавання S до зразка довжиною n-1, або L до зразка довжиною n-2; і просодіцісти показали, що число зразків довжиною n є сумою двох попередніх чисел в послідовності. Дональд Кнут розглядає цей ефект в книзі «Мистецтво програмування».

На Заході ця послідовність була досліджена Леонардо Пізанським, відомим як Фібоначчі, в його праці «Liber Abaci» (1 202).

Цей алгоритм має кілька привабливих особливостей:

- якщо час розрахунку n-го числа t, то (n + 1) -го - t * φ, де φ - це золотий перетин (√5 + 1) / 2;

- саме обчислюється n-е число округлене до ближайщего цілого величиною φn / √5;

- розрахунок fib (n + 1) вимагає n-й вложеності викликів.

Перша особливість дозволяє за невеликий час протестувати транслятори, швидкості роботи яких розрізняються в сотні тисяч разів. Друга особливість дозволяє швидко перевіряти правильність розрахунків. Третя особливість теоретично дозволяє досліджувати ємність стека, але через те, що розрахунок при n> 50 стає дуже повільним навіть на суперкомп'ютері, практично використовувати цю особливість не представляється можливим.

Функція Аккермана – простий приклад всюди певної обчислюваної функції, яка не є примітивно рекурсивної. Вона приймає два невід'ємних цілих числа в якості параметрів і повертає натуральне число. Ця функція зростає дуже швидко.

Наприкінці 20-х років XX століття математики-учні Давида Гільберта-Габріель Судан і Вільгельм Аккерман вивчали основи обчислень. Судан і Аккерман відомі за відкриття всюди певної обчислюваної функції, котра є примітивно рекурсивної. Судан відкрив менш відому функцію Судану, після чого, незалежно від нього, в 1928 році Аккерман опублікував свою функцію. Крім її історичної ролі як першої всюди безумовно не примітивно рекурсивної обчислюваної функції, оригінальна функція Аккермана розширювала основні арифметичні операції за спорудження до рівня, хоча і не так добре, як спеціально призначені для цього функції на кшталт послідовності гіпероператор Гудстейн.

У статті «On the infinite» Гільберт висловив гіпотезу про те, що функція Аккермана не є примітивно рекурсивної, а Аккерман довів цю гіпотезу в статті «On hilbert's construction of the real numbers». Роза Петер і Рафаель Робінсон пізніше представили двухаргументную версію функції Аккермана, яку тепер багато авторів вважають за краще оригінальної.

Ця функція з ростом n росте дуже швидко, але вважається дуже повільно. Остання властивість теоретично зручно для тестування швидкодії. Однак, розрахунок цієї функції вимагає значного числа рекурсивних викликів і більшість тестованих мов виявилося не в змозі їх підтримувати для обчислень,

що мають помітну тривалість. Відомо, що обчислення цієї функції не можна звести до ітерації. Розрахунок по цьому завданню дозволив досліджувати максимальну ємність стека досліджуваних мов: розрахунок ack (1,1, n-1) вимагає n-й вкладеності викликів і дуже швидкий.

РОЗДІЛ 2. АНАЛІЗ ТЕХНОЛОГІЙ ТА МЕТОДІВ РОЗРОБКИ ОРІЄНТОВАНИХ НА WEB

2.1 Принципи створення сучасних web-додатків за допомогою JS

Рендеринг сторінок на сервері здійснюється не заради SEO, а для продуктивності. Потрібно приймати до уваги додаткові запити для отримання скриптів, стилів і наступні запити до API. В майбутньому, також потрібно приймайте до уваги використання методу HTTP 2.0 Push.

Перш за все, потрібно звернути увагу на загальноприйняту помилку розділяти «додатки з рендерингом на сервері» і «односторінкові додатки». Якщо ми хочемо домогтися найкращого сприйняття з точки зору користувача, то не повинні обмежувати себе такими рамками і відмовлятися від однієї альтернативи на користь іншої.

Причини цілком очевидні. Сторінки передаються по інтернету, у якого є фізичні обмеження, що незабутньо проілюстрував Стюарт Чешир в знаменитому есе “Це latency, дурник”

Сучасне обладнання інтернету передає сигнал зі швидкістю 0,5 від швидкості світла. Це особливо важливо у зв'язку з ростом популярності JavaScript-додатків, які зазвичай містять тільки розмітку <script> і <link> поруч з порожнім полем <body>. Так звані односторінкові додатки (Single Page Applications, SPA) – сервер повертає одну сторінку, а все інше викликається кодом на стороні клієнта.

Уявімо сценарій, коли користувач безпосередньо заходить за адресою app.com/orders. До моменту, коли додаток отримує і обробляє цей запит, у нього вже є важлива інформація про те, що потрібно показувати на сторінці. Воно може, наприклад, довантажити замовлення з бази даних і додати його в відповідь. А ось більшість SPA в такій ситуації повертає порожню сторінку і тег <script>. Потім доведеться ще раз обмінятися запитами для отримання вмісту скрипта, і ще раз - для отримання контенту.

Багато розробники свідомо йдуть на таку жертву. Вони намагаються

гарантувати, що додаткові мережеві хопи для користувача відбудуться тільки один раз, відправляючи правильні заголовки для кешування в відповідях зі скриптами і CSS. Загальноприйнята думка полягає в тому, що це прийнятна угода, тому що після завантаження всіх файлів на комп'ютер більшість дій користувача (як переходи в інші розділи) здійснюються без запитів додаткових сторінок або скриптів.

Однак, навіть з урахуванням кеша, є певний програш в продуктивності, якщо врахувати час на парсинг та виконання скрипта. Що ще гірше, зазвичай користувач не отримує ніякого фідбек в той час, як завантажуються скрипти. Результат - чиста сторінка на екрані, яка потім раптово перетворюється в повністю завантажену сторінку.

Найголовніше, зазвичай забувається, що найбільш поширені транспорт для передачі інтернет-даних (TCP) стартує повільно. Це майже напевно гарантує, що більшість комплектів зі скриптами ні передані за один раз, роблячи вищеописану ситуацію ще гірше.

TCP-з'єднання починається з обміну пакетами для рукостискання. Якщо ви використовуєте SSL, що важливо для безпечної передачі скриптів, відбувається два додаткових обміну пакетами (один, якщо клієнт відновлює сесію). Тільки після цього сервер може почати відправку даних, але практика показує, що він робить це повільно і порційно.

Механізм контролю заторів під назвою Slow Start вбудований в протокол TCP, щоб відправляти дані, поступово нарощуючи кількість сегментів. Це має два серйозних виведення для SPA.

Великі скрипти завантажуються набагато довше, ніж здається. Потрібно «чотири обміну пакетами і сотні мілісекунд затримки, щоб вийти на 64 КБ обміну даними між клієнтом і сервером». Наприклад, в разі швидкого інтернет-з'єднання між Лондоном і Нью-Йорком, потрібно 225 мс, перш ніж TCP зможе вийти на максимальний розмір пакету.

Оскільки це правило діє також для первинного завантаження сторінки, то

дуже важливо, який контент вантажиться для рендеринга на сторінці в першу чергу.

Веб-сайти, яким вдається доставити контент (нехай навіть базову розмітку без даних) в цьому напрямі, здаються виключно чуйними. Насправді, багато авторів швидких серверних додатків сприймають JavaScript як щось непотрібне або що потрібно використовувати з великою обережністю.

У деяких випадках, неактуальну в даний момент для користувача частина сторінки краще виключити з початкового відповіді і залишити на потім. Деякі додатки, наприклад, вважають за краще здійснити рендеринг тільки «ядра» сторінки для забезпечення негайного відгуку. Потім вони запитують різні частини сторінки паралельно. Це забезпечує кращу чуйність навіть в ситуації з повільним застарілим бекенд. Для деяких сторінок хорошим варіантом буде рендеринг тільки видимої частини сторінки.

Винятково важливою є якісна оцінка скриптів і стилів з урахуванням інформації, яка у сервера є про сесії, клієнтів і URL. Скрипти, які здійснюють сортування замовлень, очевидно будуть важливішими для /orders, ніж логіка сторінки налаштувань. Можливо, не настільки очевидна, але є різниця в завантаженні «структурного CSS» і «CSS для оформлення». Перший може знадобитися для коду JavaScript, так що потрібно блокування, а другий завантажується асинхронно.

Досить гнучка система, яка розділяє рендеринг між браузером і сервером і надає інструменти для поступової завантаження скриптів і стилів, цілком може стерти межу між веб-сайтами та веб-додатками. І те, і інше використовує URL'и, навігацію, демонструє дані користувачеві. Навіть додаток з електронними таблицями, яке традиційно покладається на функціональність з клієнтської сторони, спочатку має показати клієнту інформацію, яку потрібно редагувати.

Найбільший недолік продуктивності в багатьох популярних системах в наш час пояснюється прогресивним накопиченням складності в стеці.

Негайна відповідь на дії користувача. JavaScript дозволяє взагалі заховати мережеву затримку. Використовуючи це як принцип дизайну, ми можемо навіть забрати з програми майже всі індикатори завантаження і повідомлення "loading". PJAX або TurboLinks упускають можливості по збільшенню суб'єктивної швидкості інтерфейсу.

Основне завдання полягає в максимальному прискоренні реакції на дії користувача. Скільки б зусиль не вкладали в зменшення числа хопов при роботі з веб-додатком, але є речі поза контролем розробника. Це теоретична межа швидкості світла і мінімальний пінг між клієнтом і сервером.

Якщо якість зв'язку погане, то буде відбуватися повторна передача пакетів. Там, де контент повинен завантажуватися за пару секунд, може знадобитися набагато більше.

У цьому головна перевага JavaScript для поліпшення UX. Якщо на стороні клієнта інтерфейс управляється за допомогою скриптів, розробники можуть заховати мережеву затримку.

JavaScript дозволяє реагувати негайно і оптимістично на дії користувача. Натискання на посилання або кнопку призводить до негайної реакції, без звернення в Мережу. Відомий приклад - це інтерфейс Gmail (або Google Inbox), в якому архівація поштового повідомлення відбувається негайно, тоді як відповідний запит до сервера відправляється і обробляється асинхронно.

У випадку з формою, замість очікування якогось коду HTML як відповідь на її заповнення, ми можемо реагувати відразу, як тільки користувач натиснув "Enter". Або навіть краще, як робить пошук Google, ми можемо реагувати ще раніше, готуючи розмітку для нової сторінки завчасно.

Така поведінка – приклад того, що називається адаптацією розмітки. Основна ідея полягає в тому, що сторінка «знає» свою майбутню розмітку, так що може переключитися на неї тоді, коли ще не має даних для вказівки на це. Це «оптимістичне» поведінка, тому що все ще залишається ризик, що дані ніколи не надійдуть, і доведеться виводити повідомлення про помилку, але це,

очевидно, трапляється рідко.

Головна сторінка google цілком підходить в якості прикладу, тому що вона дуже чітко демонструє перші два принципи.

По-перше, пакетний дамп tcp-з'єднання з www.google.com показує, що вони спеціально намагаються відправити всю сторінку цілком відразу після отримання запиту. Весь обмін пакетами, включаючи закриття з'єднання, займає 64 мс. Ймовірно, це було актуально для них з самого початку.

В кінці 2004 року, компанія google стала піонером у використанні javascript для видачі підказок в реальному часі в процесі набору пошукового запиту (цікаво, що цю функцію співробітник розробив в вільні від основної роботи 20% часу, так само як і gmail). Це навіть стало фундаментом для появи ajax. І в 2010 вони представили instant search, в якому js грає центральну роль.

Реакція на зміну даних. Коли на сервері оновлюються дані, клієнта слід повідомляти без затримки. Це така форма підвищення продуктивності, коли користувача звільняють від необхідності здійснювати додаткові дії (натискати F5, оновлювати сторінку).

Третій принцип відноситься до реагування UI на зміну даних в джерелі, звичайно, в одному або декількох серверах баз даних.

Відходить у минуле модель передачі по HTML даних, які залишаються статичними до тих пір, поки користувач не оновить сторінку (традиційні веб-сайти) або не взаємодіє з нею (Ajax).

Це критично важливо в світі з наростаючим потоком інформації з різних джерел, включаючи годинники, телефони, планшети і носяться пристрої, які з'являться в майбутньому.

Було б невірним, однак, припустити, що переваги миттєвого оновлення UI обмежуються тільки на багато користувачів додатками. Ось чому потрібно говорити про узгоджені дата-поінти, замість користувачів.

Контроль обміну даними з сервером. Потрібно переконуватись в обробці помилок, повторних запитах на користь клієнта, синхронізації даних у

фоновому режимі і збереженні кеша в офлайні. Коли з'явився веб, обмін даними між клієнтом і сервером був обмежений кількома способами – натискання на посилання відправить GET для отримання нової сторінки і її рендеринга, відправлення форми відправить POST або GET з подальшим рендерингом нової сторінки, впровадження зображення або об'єкта відправить GET асинхронно з подальшим рендерингом.

Простота такої моделі дуже приваблива, і зараз все виразно ускладнилося, коли мова йде про розуміння, як отримувати і відправляти інформацію.

Головні обмеження стосуються другого пункту. Неможливість відправити дані без обов'язкової завантаження нової сторінки було недоліком з точки зору продуктивності. Але найголовніше, що це повністю ламало кнопку “Назад”.

Саме тому веб як платформа для додатків залишався неповноцінним без JavaScript. Ajax був величезний стрибок вперед з точки зору зручності в частині публікації інформації користувачем.

Зараз є безліч API (XMLHttpRequest, WebSocket, EventSource, це лише деякі з них), які дають повний і чіткий контроль над потоком даних. Крім можливості публікувати призначені для користувача дані через форму, з'явилися нові можливості щодо поліпшення UX.

Пряме відношення до попереднього принципу має показ стану з'єднання. Якщо користувач очікує, що дані будуть оновлюватися автоматично, то потрібно його інформувати про факти втрати зв'язку і спробах її відновлення.

При виявленні дисконнекта, корисно зберегти дані в пам'яті (а ще краще, в localStorage), так що їх можна відправити пізніше. Це особливо важливо в світлі майбутнього використання ServiceWorker, який дозволяє додаткам JavaScript працювати у фоновому режимі. Якщо ваш додаток не відкрите, ви все ще можете продовжувати спроби синхронізувати дані з сервером у фоновому режимі.

Потрібно враховувати можливість таймаутів і помилок при відправці даних, такі ситуації повинні вирішуватися на користь клієнта. Якщо з'єднання

відновлено, потрібно спробувати відправити дані знову. У разі постійної помилки, потрібно повідомити про це користувачеві.

Деякі помилки потрібно обробляти особливо уважно. Наприклад, несподівана 403 може означати, що призначена для користувача сесія визнана недійсною. У таких випадках, є можливість відновити сеанс, якщо показати користувачеві вікно для введення логіна і пароля.

Важливо ще переконатися, що користувач, бува, не перерве потік даних. Це може статися в двох ситуаціях. Перший і найочевидніший випадок - закриття браузера або вкладки, що потрібно намагатися запобігти оброблювачем beforeunload.

Інший (і менш очевидний) випадок – спроба переходу на іншу сторінку, наприклад, натискання на посилання. У цьому випадку програма може зупинити користувача іншими методами, на розсуд розробника.

Якщо браузер не буде керувати URL'амі і історією, виникнуть нові проблеми. Якщо не брати до уваги відправки форм, то при використанні в веб-додатку одних тільки гіперпосилань у розробника буде повністю функціональна навігація «Вперед / Назад» у браузері.

Наприклад, типову «нескінченну» сторінку зазвичай роблять за допомогою кнопки на JavaScript, яка запитує додаткові дані/HTML і вставляє їх. На жаль, мало хто при цьому пам'ятають про необхідність виклику history.pushState або replaceState як обов'язкового кроку.

2.2 Принципи розробки сучасних web-додатків на PHP

PHP розробка тоді і зараз. Для більшості розробників робота над PHP додатками зараз і десять років тому відрізняється кардинально.

За останні роки картина досить сильно змінилася. Прості сайти тепер вдають із себе цілі SaaS додатки – навіщо створювати черговий двигун для блогу, вбудовувати CMS або ecommerce систему, коли вже десятки різних продуктів існують. Іншими словами, все що можна було автоматизувати вже оптимізовано. Але залишається один момент не піддається цьому процесу –

потреби замовників.

Тому веб-розробники стали спеціалізуватися саме на тих ділянках, які неможливо оптимізувати – веб-додатках. У парі з цим прийшли основні ідеї коректної розробки на PHP.

Багато застосовують фреймворки як Laravel5, CakePHP3, Symfony2 і т.д. Вони вже містять в собі стилі та шаблони проектування.

Дотримання одного стилю – є дуже важливим елементом при розробці. Один стиль простіше читати колегам, відповідно нові співробітники легко зможуть освоїти код.

Відкривати вихідний код програми (або його модулів) може дуже позитивно позначитися: безкоштовна допомога в розробці, безкоштовна реклама.

Автоматичне тестування завжди відставало в PHP в порівнянні з іншими мовами програмування. Але, на щастя, ситуація змінилася в кращу сторону. Думка, що написання тестів подвоює витрачений час, практично викорінено, і додавання нового функціоналу стало немислимо без відповідного тесту.

Добре покрите тестами додаток дає ряд переваг. Ось список моїх трьох улюблених:

- рефакторинг такого ПО значно простіше;
- юніт тестування підштовхує розробника використовувати модульний стиль;
- тести надають не тільки приклади коду, але і документацію до нього.

Залежностями однозначно повинен управляти Composer. Так як код дуже тісно пов'язаний з певними бібліотеками та їх версіями, то заявлені залежності (composer.lock) повинен також знаходиться в GIT.

2.3 Розробка на Ruby

Ruby – проста і потужна мова програмування, можливість мета-програмінгу, блоків, ітераторів, а також обробки виключень робить мову чудовою основою для фреймворка. Найбільш популярним фреймворков на мові

програмування Ruby є Ruby on Rails. Побачити, як допомагає мета-програмування Ruby, можна побачити в ORM компоненті RoR Active Record. Грунтуючись на імені класу, RoR зчитує схему (schema) і нальоту створює об'єкти класу на базі таблиці БД. RoR реалізує MVC

Active Record – це Модель в RoR. Модель зберігає дані і надає базу для роботи з даними. Крім цього Active Record також є ORM фреймворком. ORM значить Object-relational mapping (Об'єктно-реляційна проекція). Власне Active Record робить багато речей. Проекція таблиці на клас. Кожна таблиця проектується на один або кілька класів за принципом convention over configuration (угода вище конфігурації). Одне з таких угод – ім'я таблиці повинно бути у множині, а назва класу – в єдиному. Атрибути таблиці нальоту проектуються в атрибути примірника Ruby. Після того, як всі проекції зроблені, кожен об'єкт ORM класу представляє певну рядок таблиці, з якої клас був спроектований.

З'єднання з БД. Можна підключитися до бази даних, використовуючи API, що надається Active Record, який створює необхідний запит безпосередньо в двигун БД за допомогою адаптерів. У Active Record є адаптери для MySQL, Postgres, MS SQLServer, DB2, і SQLite. Необхідно лише записати параметри доступу до БД.

Операції CRUD. Це операції create (створення), retrieve (отримання), update (оновлення) і delete (видалення) над таблицею. Так як Active Record – це ORM фреймворк, він завжди працюєте з об'єктами. Щоб створити новий рядок таблиці, створюєте новий об'єкт класу і заповнюєте його змінні екземпляра значеннями. Варто зауважити, що все це Active Record робить за вас.

Перевірка даних. Перевірка даних перед приміщенням їх в таблицю – це перший крок в безпеці проекту. Active Record надає перевірку Моделі. Дані можуть бути перевірені автоматично за допомогою безлічі готових методів, які, в разі необхідності, можна переписати під власні потреби.

RoR заохочує використовувати окремі середовища для кожного з етапів життєвого циклу програми – розробка (development), тестування (testing) та експлуатація (production), для кожного з яких створюється окрема БД.

У середовищі розробки ставка робиться на негайне відображення нового варіанту при зміні коду – досить оновити сторінку в браузері. Швидкість в цьому середовищі не важлива. Коли трапляється помилка, вона виводиться на екран.

При тестуванні зазвичай кожен раз наповнюється БД якимось текстом, щоб переконатися, що нормальна поведінка не залежить від змісту БД. Процедури юніт-тестінгу і тесту функціональності в RoR автоматизовані і здійснюються через консоль. Тестова середовище надає окремий простір, в яких оперують ці процедури.

Зрештою додаток виходить до фінальної межі, пройшовши тести і позбувшись від багів. Тепер поновлення коду відбуватимуться рідко і можна сконцентруватися на продуктивності, включити кешування. Немає необхідності писати величезні логи помилок і лякати користувачів повідомленнями про ці помилки в браузері.

РОЗДІЛ 3. АНАЛІЗ МОВ ПРОГРАМУВАННЯ

3.2 Опис структури проекту

Проект складається з наступних логічних частин:

- php – каталог с контролерами;
    - fibaonachi_test.php – Тест Фібоначчі;
    - akkerman_test.php – Тетс Аккермана;
    - ml_test.php – Тетс машинного навчання;
- ruby – містить всі ассети проекту;
    - fibaonachi_test.rb – Тест Фібоначчі;
    - akkerman_test.rb – Тетс Аккермана;
    - ml_test.rb – Тетс машинного навчання;
- js – містить контролери, які викликаються з командного рядку;
    - fibaonachi_test.js – Тест Фібоначчі;
    - akkerman_test.js – Тетс Аккермана;
    - ml_test.js – Тетс машинного навчання;
- python – містить конфігураційні фали проекту;
    - fibaonachi_test.py – Тест Фібоначчі;
    - akkerman_test.py – Тетс Аккермана;
    - ml_test.py – Тетс машинного навчання;

3.2 Аналіз технічніх характеристик мов програмування

Першим тестом буде тест алгоритммом Фибоначчі. В рамках цього тесту розраховуеться розмур стеку та час на виконання розрахунку послідовності Фібоначчі до 50.

Спочатку буде проведенодослідження мови PHP, потім Ruby, JS та Python. Для виконання тестів беруться налаштування за замовченням для кожного середовища мови програмування.

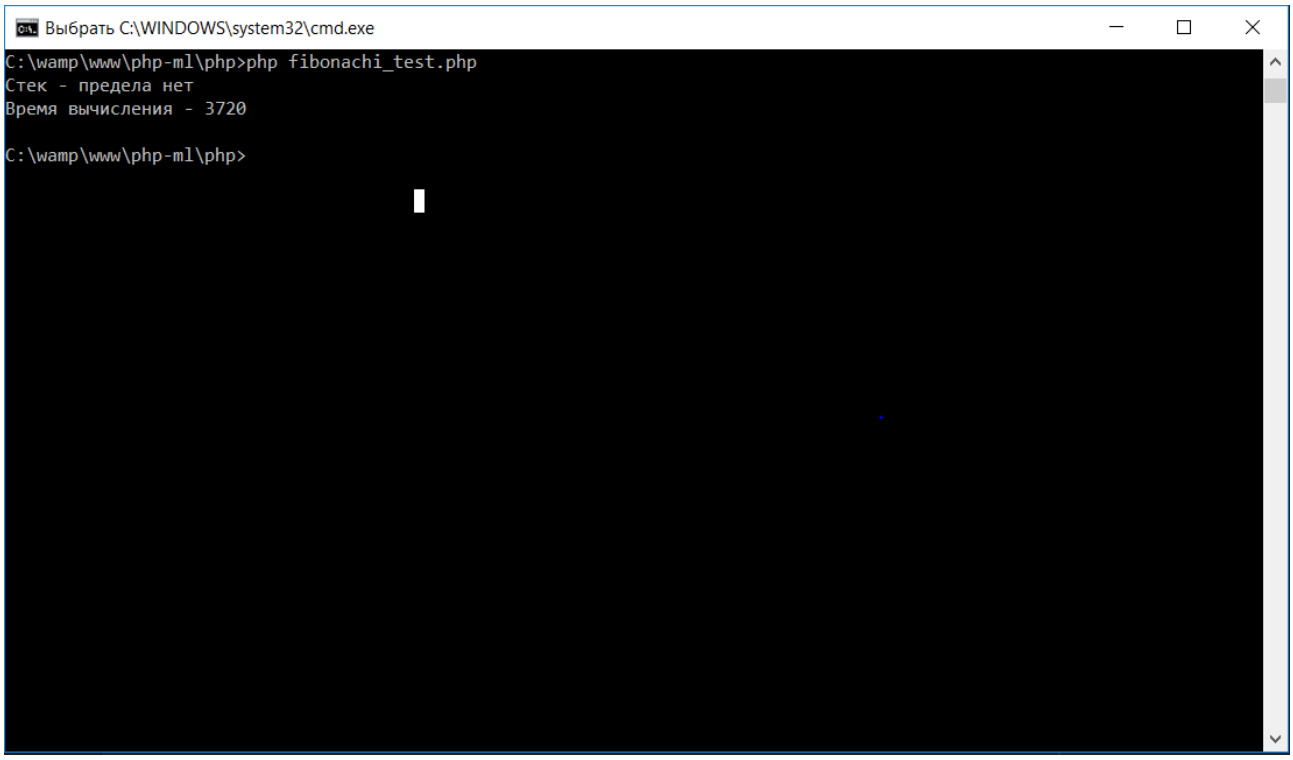

Рис. 3.1 – Результат тесту Фібоначчі для мови PHP

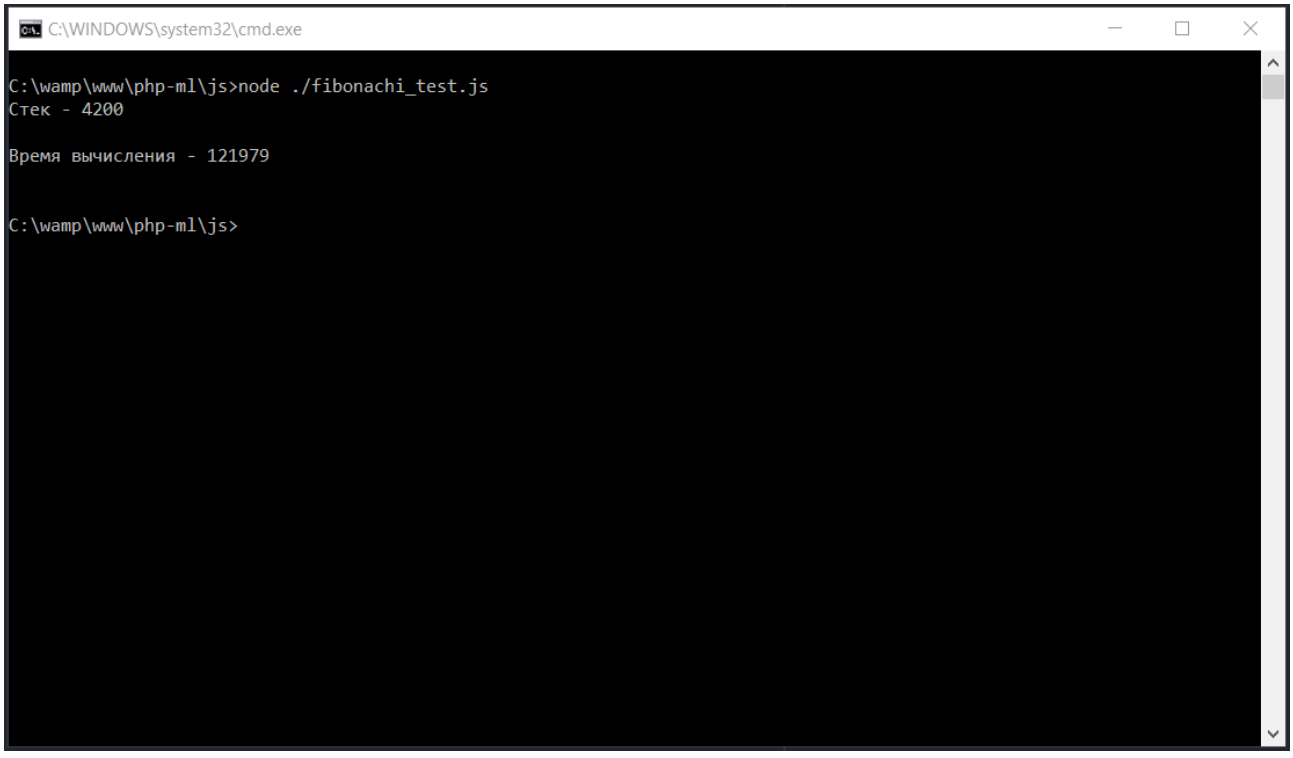

Рис. 3.2 – Результат тесту Фібоначчі для мови JS

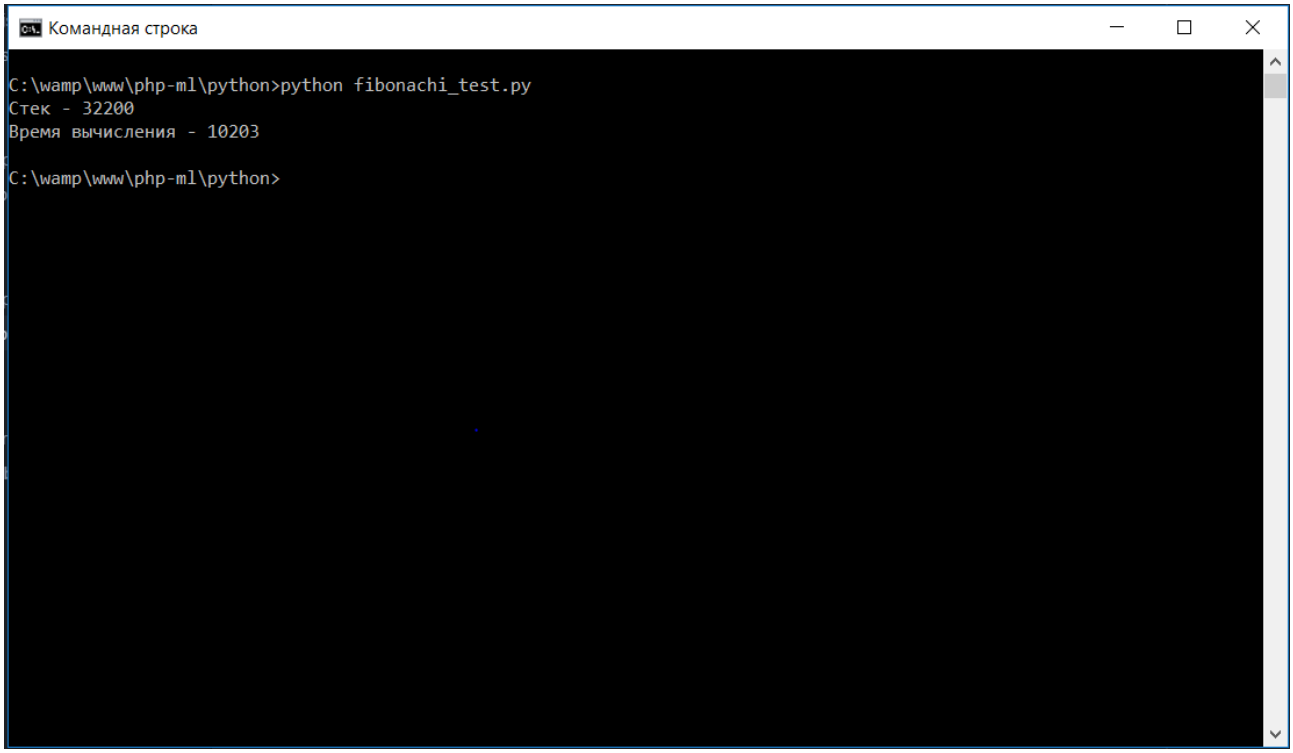

Рис. 3.3 – Результат тесту Фібоначчі для мови Python

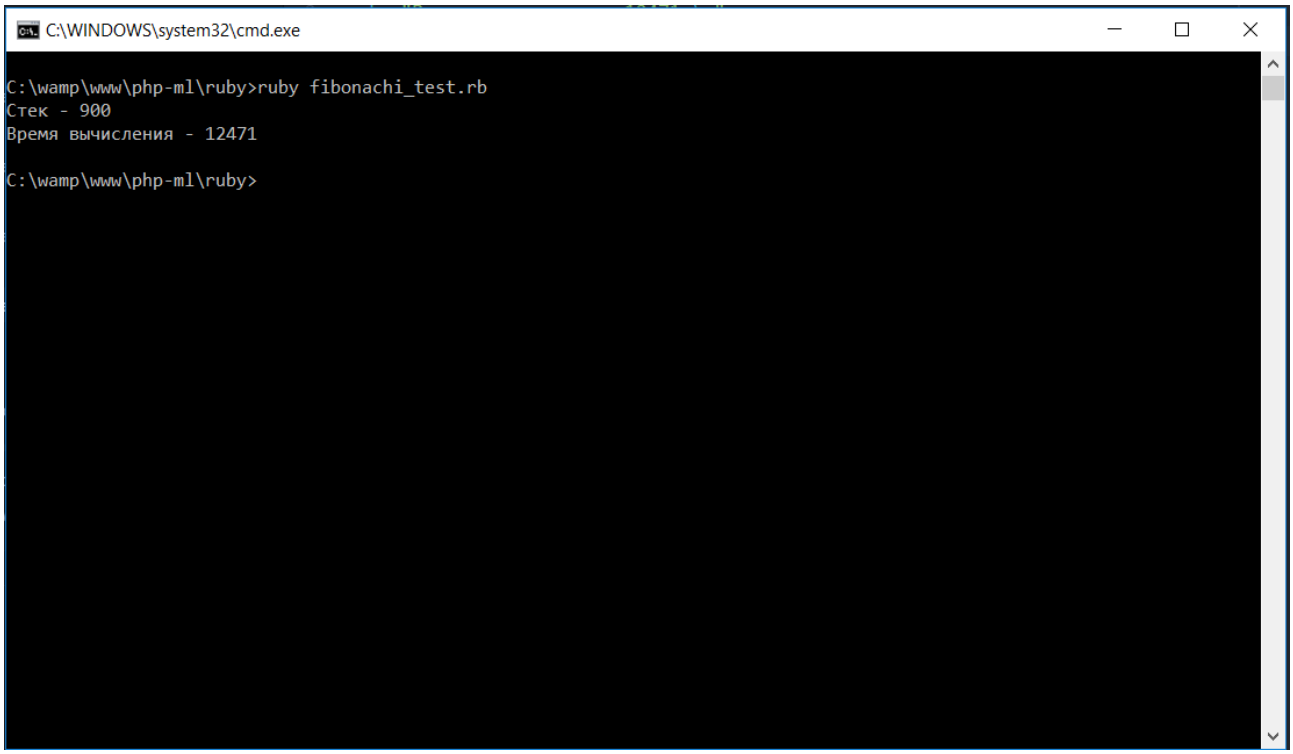

Рис. 3.4 – Результат тесту Фібоначчі для мови Ruby

Таблиця 3.1 – Результати тесту алгоритмом Фібоначі

Мова	**Стек**	**Час виконання, с.**
Python	32200	10203
Ruby	900	12471
JS	4200	129979
PHP	-	13200

З преведених результатів можна з'ясувати, що PHP не має межі стеку при кінечному значенні для послідовності акермана в 50. Також PHP досить гарну швидкысть. Серед мов, які мають межу стеку найкраще себе показав Python.

Наступний тест – тест алгоритмом Акермана. Завдяки ньому тестується бістро дія та кількість операцій на такт.

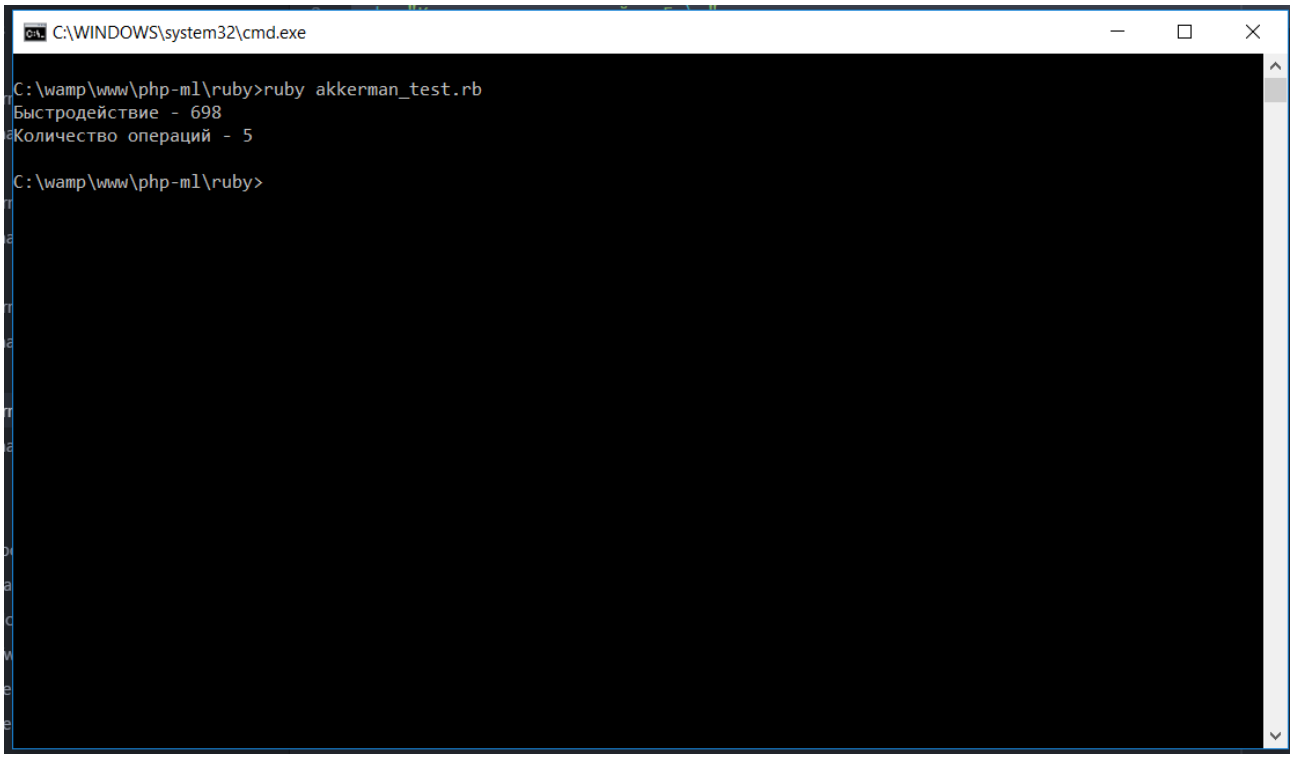

Рис. 3.5 – Результат тесту Аккермана для мови Ruby

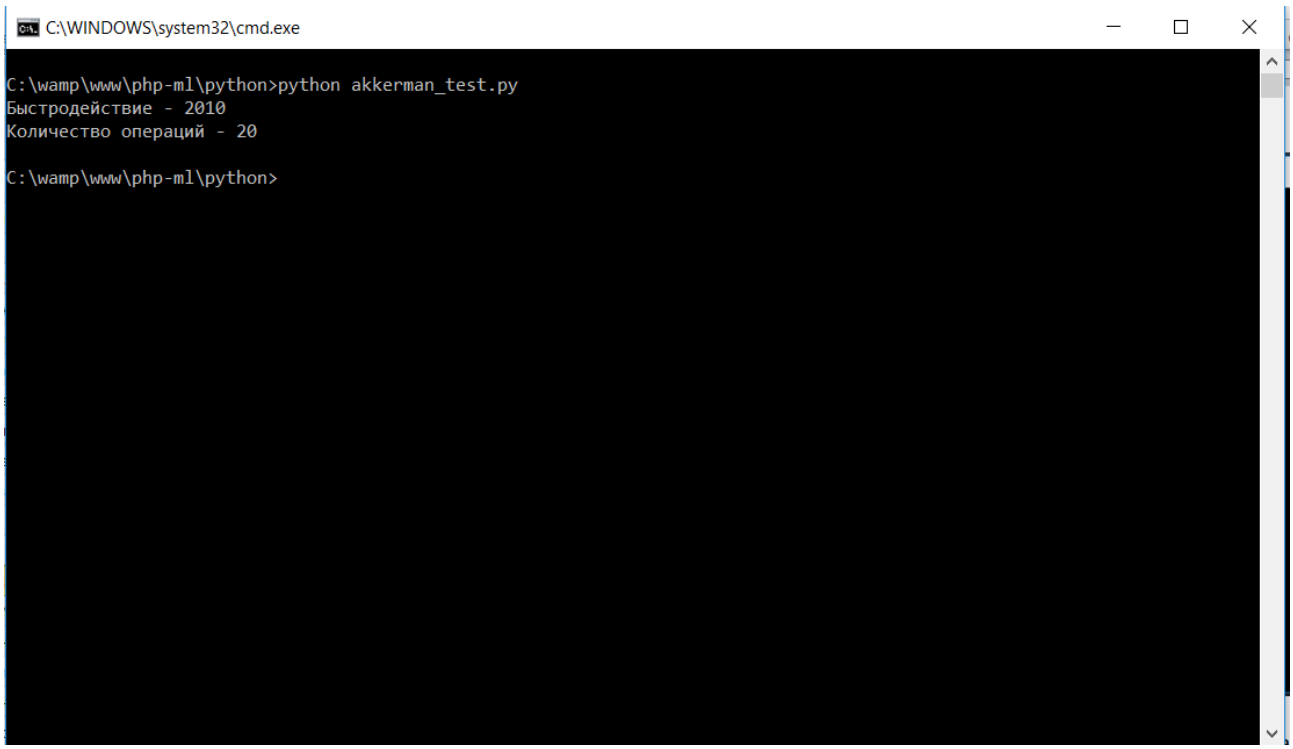

Рис. 3.6 – Результат тесту Аккермана для мови Python

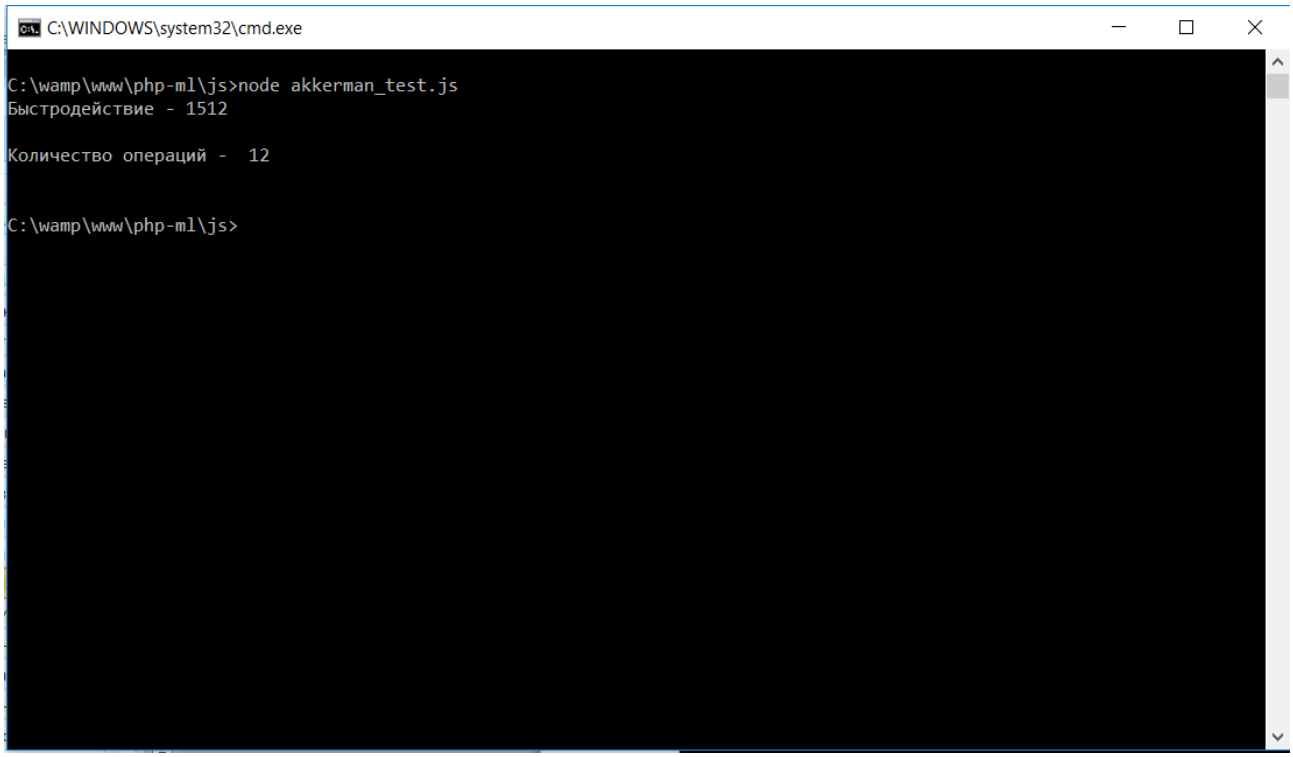

Рис. 3.7 – Результат тесту Аккермана для мови JS

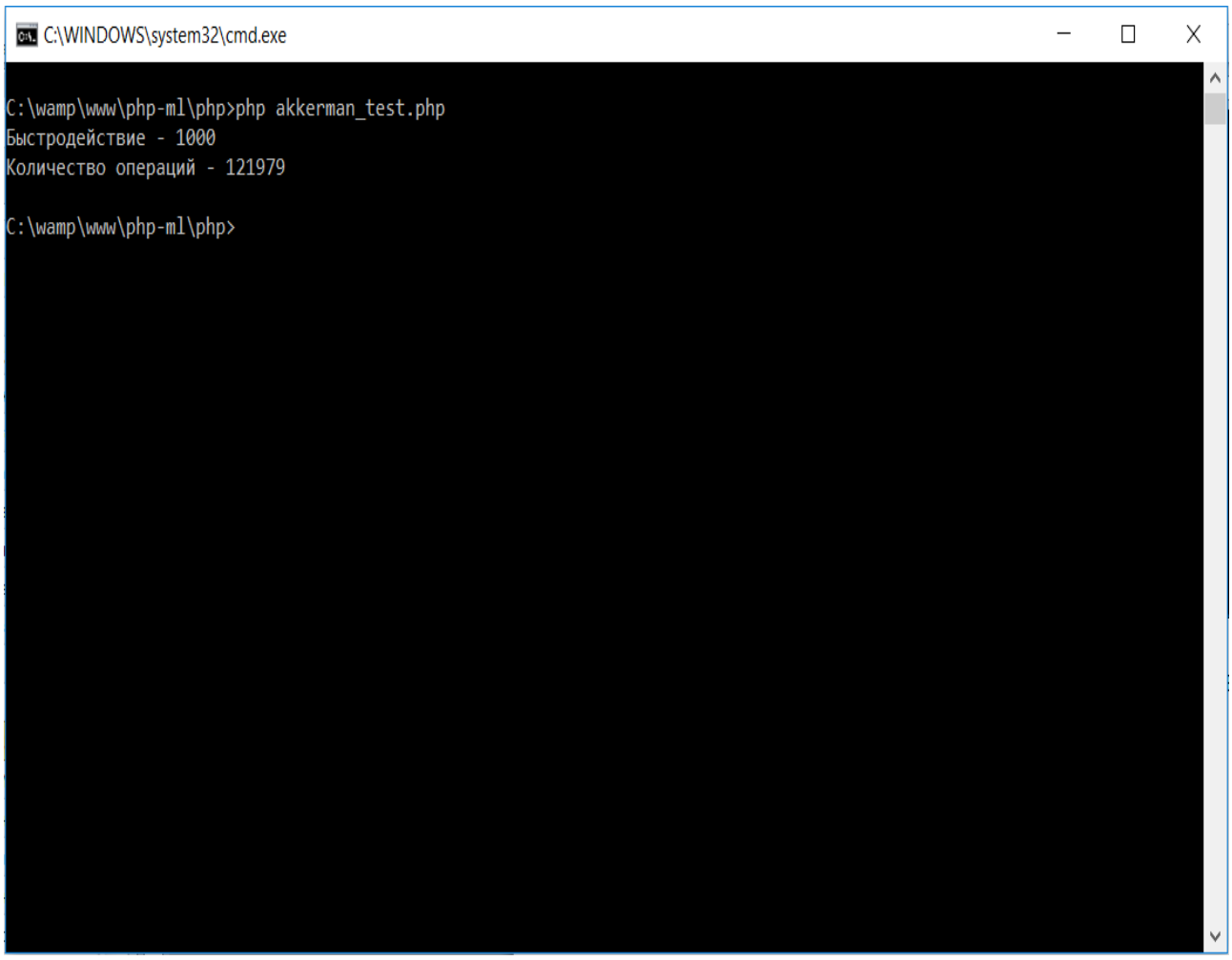

Рис. 3.8 – Результат тесту Аккермана для мови PHP

Таблиця 3.2 – Результати тесту алгоритмом Аккермана

Мова	**Кількість операцій**	**Швидкість, с.**
Python	2010	20
Ruby	695	5
JS	12	1512
PHP	12979	1000

3.3 Аналіз можливостей мови до машинного навчання

В якості датасету для машинного навчання використовувався стандартний сет з позитивними та ннегативними повідомленнями в твітері.

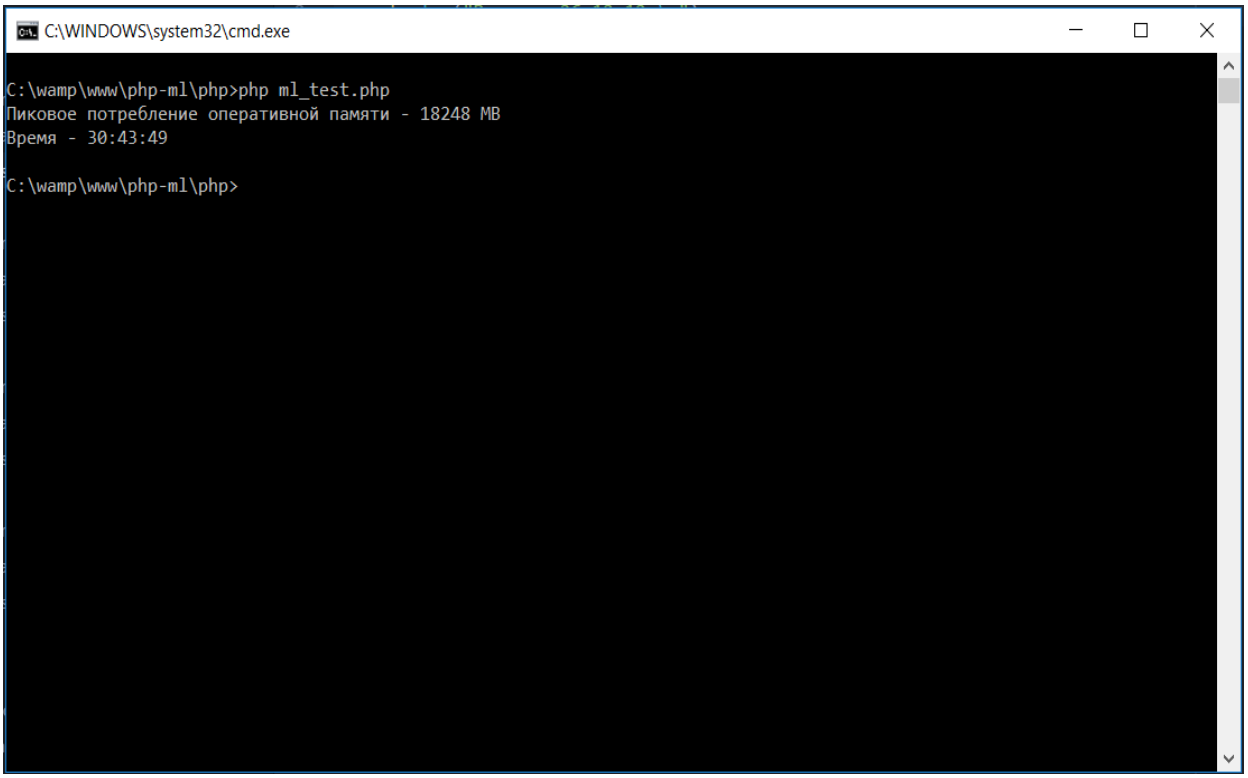

Рис. 3.9 – Результат машинного навчання мовою PHP

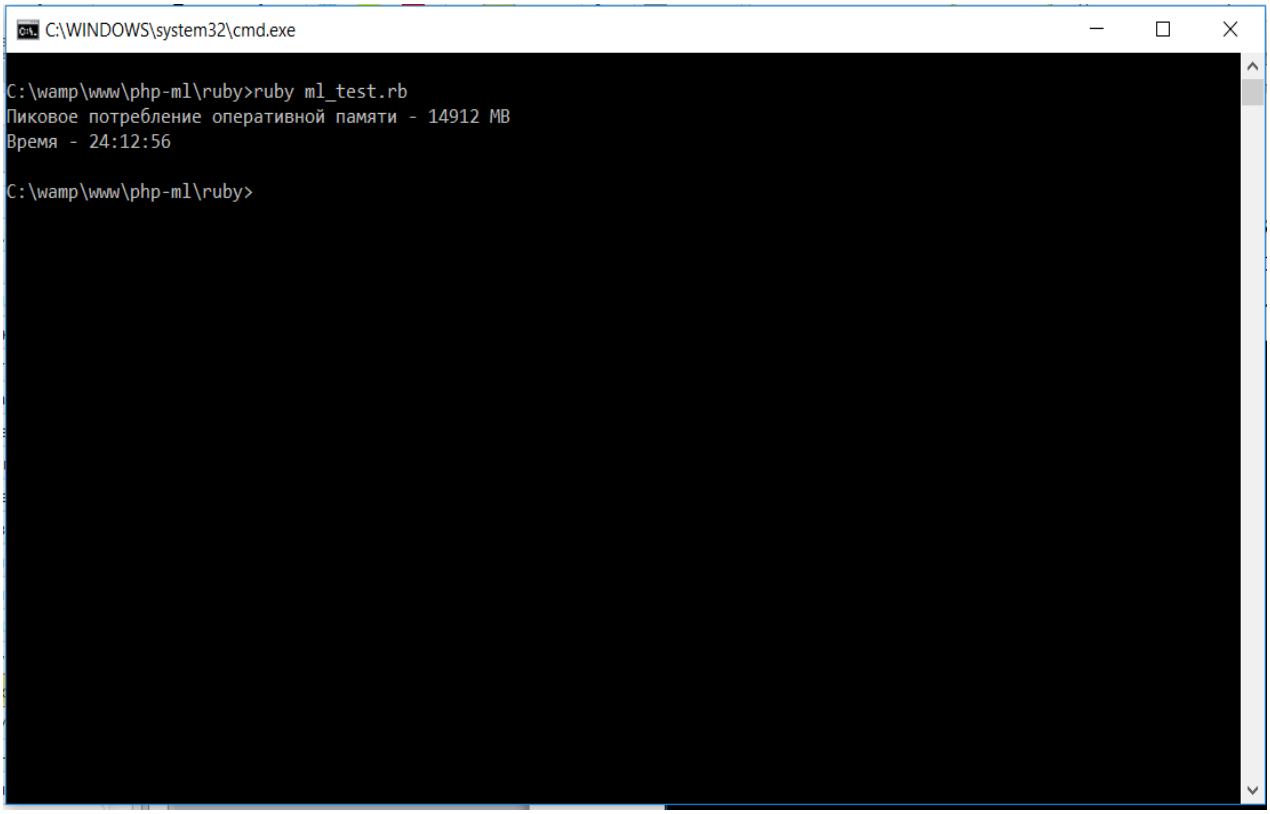

Рис. 3.10 – Результат машинного навчання мовою Ruby

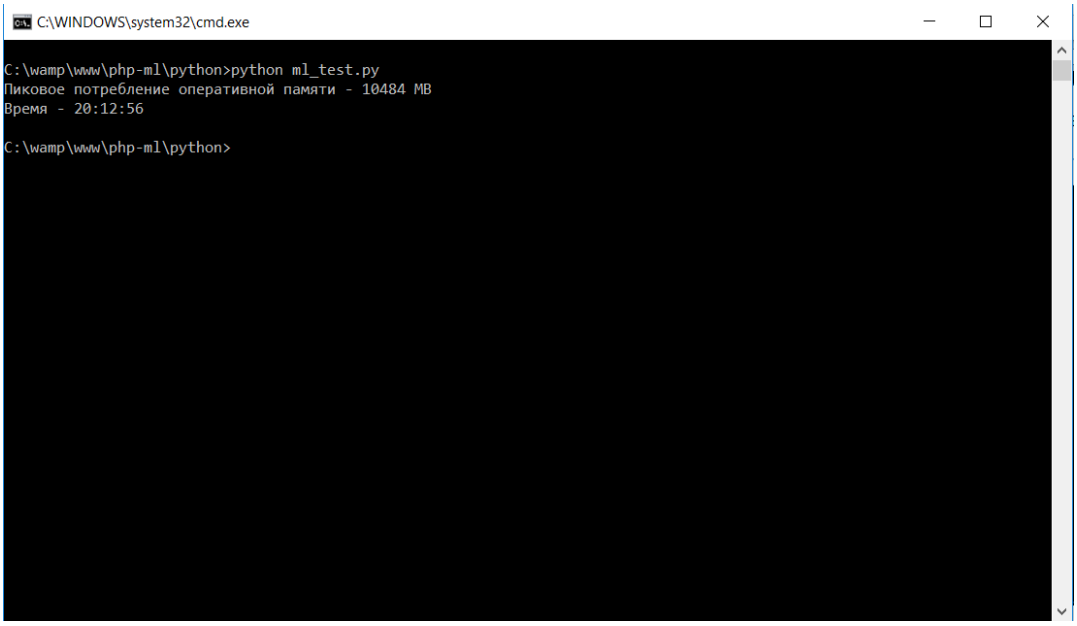

Рис. 3.11 – Результат машинного навчання мовою Python

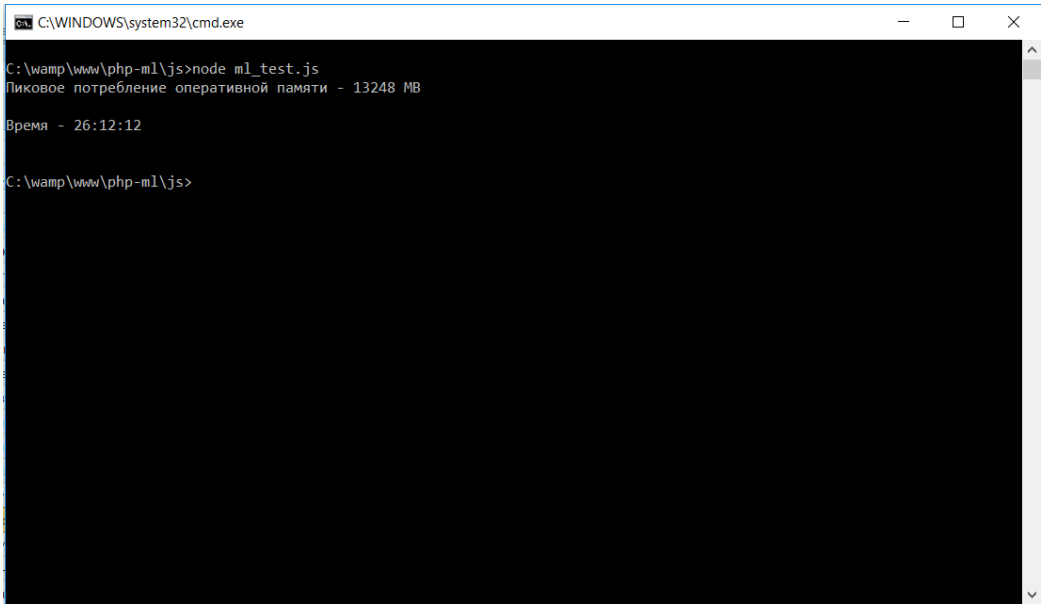

Рис. 3.12 – Результат машинного навчання мовою JS

Таблиця 3.3 – Результати тесту машинним навчанням

Мова	**Кількість ОЗУ, MB**	**Час, с.**
Python	10484	20:12:56
Ruby	14912	24:12:56
JS	13248	26:12:12
PHP	18248	30:43:49

РОЗІЛ 4. ЕКОНОМІЧНИЙ

Дані для розрахунків:

Передбачуване число операторів – 1757.

Коефіцієнт складності програми – 1.50.

Коефіцієнт корекції програми – 0.20.

Коефіцієнт кваліфікації програміста – 0.8.

Годинна заробітна плата програміста – 31.

Вартість машино- години ЕОМ – 5.

Коефіцієнт збільшення витрат праці внаслідок поганого опису – 1.3.

4.1 Визначення трудомісткості розробки програмного забезпечення

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста, тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість можна розрахувати за формулою:

$$t = t_о + t_и + t_а + t_п + t_{отл} + t_д \text{ люд.-годин} \qquad (4.1)$$

де $t_о$ – витрати праці на підготовку і опис поставленого завдання (приймається = 50);

t_u – витрати праці на дослідження алгоритму розв'язання задачі;

$t_а$ – витрати праці на розробку блок-схеми алгоритму;

$t_п$ – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються виходячи з умовного числа операторів в розробляється ПО.

Умовне число операторів (підпрограм):

$$Q = qC(1 + p) \qquad (4.2)$$

де q – передбачувана кількість операторів (q = 1757).

С – коефіцієнт складності програми. Для розроблюваного програмного продукту візьмемо коефіцієнт складності завдання 1,5.

Р – коефіцієнт корекції програми в ході її розробки. Коефіцієнт корекції програми р – збільшення обсягу робіт за рахунок внесення змін до алгоритму або програму за результатами уточнення постановок і описів її, зміни складу і структури інформації, а також уточнень, внесених розробниками для поліпшення якості самої програми без зміни постановки завдання. Візьмемо коефіцієнт рівний 0,2.

$$Q = 1757 * 1,5 * (1 + 0,2) = 3163$$

Витрати праці на вивчення опису завдання визначається з урахуванням уточнення опису і кваліфікації програміста.

$$t_u = \frac{QB}{(75…85)K}, \text{ люд.-годин} \qquad (4.3)$$

В – коефіцієнт збільшення витрат праці унаслідок недостатнього опису завдання. Коефіцієнт збільшення витрат праці в залежності від складності завдання приймається від 1,2 до 1,5, приймемо В = 1,3.

К – коефіцієнт кваліфікації програміста, який визначається від стажу роботи за даною спеціальністю. Коефіцієнт становить: для працюючих до двох років – 0,8; від двох до трьох років – 1,0; від трьох до п'яти років – 1,1 - 1,2; від п'яти до семи – 1,3 - 1,4; понад сім років – 1,5 - 1,6. Тому приймемо К = 0,8.

$$t_u = \frac{3163 * 1,3}{77 * 0,8} = 66,7 \text{ люд.-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20…25)K}, \qquad (4.4)$$

$$t_a = \frac{3163}{23 * 0,8} = 171,9 \text{ люд.-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20..25)K}, \qquad (4.5)$$

$$t_n = \frac{3163}{22*0,8} = 179,7 \text{ люд.-годин.}$$

Витрати праці на налагодження програми за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)K}, \quad (4.6)$$

$$t_{отл} = \frac{3163}{4*0,8} = 988,4 \text{ люд.-годин.}$$

Витрати праці на налагодження програми за умови комплексного налагодження завдання:

$$t^{к}_{отл} = 1,5*t_{отл}, \quad (4.7)$$

$$t^{к}_{отл} = 1,5 * 988,4 = 1482,6 \text{ люд.-годин.}$$

Витрати на підготовку документації:

$$t_д = t_{др} + t_{до}, \quad (4.8)$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукописи;

$t_{до}$ – трудомісткість редагування, друку і оформлення документації.

Трудомісткість підготовки матеріалів і рукописи визначається за формулою:

$$t_{др} = \frac{Q}{(15..20)K}, \quad (4.9)$$

$$t_{др} = \frac{3163}{17*0,8} = 232,5 \text{ люд.-годин.}$$

Трудомісткість редагування, друку і оформлення документації:

$$t_{до} = 0,75t_{др}, \quad (4.10)$$

$$t_{до} = 0,75 * 232,5 = 174,3 \text{ люд.-годин.}$$

Витрати на складання документації становлять:

$$t_д = 232,5 + 174,3 = 406,8 \text{ люд.-годин.}$$

Отримуємо трудомісткість розробки ПЗ:

$$t = 50 + 67,7 + 171,9 + 179,7 + 988,4 + 406,8 = 1864,5 \text{ люд.-годин}$$

Таким чином, трудомісткість розробки програмного забезпечення складає 1864,5 люд.-годин.

4.2. Витрати на створення програмного забезпечення

Витрати на створення ПО ($К_{по}$) включають витрати на заробітну плату розробників програми ($З_з$ / п), яка визначається множенням сумарної трудомісткості розробки ПЗ (t) на середню заробітну плату програміста з нарахуваннями та вартості машинного часу на налагодження.

$$К_{ПО} = З_{ЗП} + З_{МВ} \tag{4.11}$$

Заробітна плата розробників визначається за формулою:

$$З_{зп} = tC_{пр}, \tag{4.12}$$

де t – загальна трудомісткість, люд.-годин.

$C_{пр}$ – середня годинна заробітна плата програміста, грн / год.

$C_{пр}$ = 31 грн. / Год.

$$З_{зп} = 1864,5 * 31 = 57800 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{мв} = t_{отл}C_{мч}, \tag{4.13}$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, г.

$C_{МЧ}$ – вартість машинного часу ЕОМ грн / год.

$$З_{мв} = 988,4 * 5 = 4942 \text{ грн.}$$

Витрати на створення програмного забезпечення складуть:

$$К_{по} = 57800 + 4942 = 62742 \text{ грн.}$$

Визначені таким чином витрати на створення програмного забезпечення є одноразовими капітальними витратами на створення АС.

Очікуваний період створення ПО:

$$Т = \frac{t}{B_k F_р} \text{ міс}, \tag{4.14}$$

де B_k – число розробників;

F_p – місячний фонд робочого часу (при 40 годинному робочому тижні F_p = 176 годин).

$$T = \frac{186465}{1*176} = 10 \text{ міс.}$$

Висновок:

Таким чином, очікувана тривалість проведення аналізу ефективності використання мов програмування спрямованих на веб в залежності від поставленого завдання становить, приблизно, 10 місяців, а витрати – 62742 грн.

4.3. Маркетингові дослідження ринку збуту розробленого програмного продукту

В дипломному проекті було досліджено ефективність використання мов програмування орієнтованих на web за вирішуваними завданнями, саме цьому ринок web розробки є единим ринком для збуту програмних напрацювань. В результаті роботи над дипломним проектом, розроблено набір скриптів для збору інформації про мови програмування при вирішенні різних завдань. При дослідженні ринку схожих програмних додатків не було виявлено, саме тому прийнята спроба охарактеризувати студіїї на цьому ринку для яких можливе використання розробленого набору скриптів, їх можна розділити на сектори за величиною.

На вересень 2017 р. число активних постійно діючих студій становить 789. Існує чотири основні цінові сектора, в яких діють студії. Охарактеризувати ці сектори можна наступним чином:

Демпінг-сектор характеризується невеликими розмірами студій (1-2 людини), часто відсутність офісу, тільки електронний контакт. Основу витрат складає заробітна плата. Призначення наднизької ціни за проекти становить основу просування на ринку. Термін життя студій в демпінг-секторі залежить від амбіцій засновника – якщо студія фактично представляє особистий фріланс-проект, то вона може існувати досить довго, якщо ж робиться спроба

організувати повноцінний бізнес (приміщення, наймані співробітники, юрособа та бухгалтерія), то термін життя в демпінг-секторі зазвичай складає не більше півтора року.

В економ-секторі діють як «персональні» студії, так і студії, що складаються з декількох співробітників. У порівнянні з демпінг-сектором, кількість друге тут істотно більше. Гонорар економ-сектора дає більше можливостей для розвитку студії, однак висуває серйозні вимоги до ефективності бізнес-процесів і скорочення витрат. Також робота в економ-секторі вимагає постійного завантаження проектами, щоб забезпечити рентабельність компанії. Компанії економ-сектора дуже чутливі до тимчасових труднощів - через низьку рентабельність бізнесу навіть невеликий простий у роботі може погубити компанію.

Найбільш комфортний для існування студії сегмент. З одного боку, ще порівняно велика кількість замовників, з іншого - гонорари бізнес-сектора дають можливість найму кваліфікованих співробітників і забезпечення комфортних умов роботи. Зрозуміло, сказане справедливо для студії, досить завантаженій проектами.

Преміум-сектор, орієнтований на великих замовників з великими бюджетами. З точки зору «простого програміста» в цьому секторі незаслужено беруться величезні гроші за просту роботу. Однак головна компетенція в цьому секторі – врахування специфіки великих клієнтів, забезпечення їм комфортного взаємодії з виконавцем. Крім того, значну частину витрат студії преміум-сектора становлять витрати на брендинг (формування «імені» студії). Рентабельність роботи в цьому секторі досить висока, однак це гідність обмежується порівняно невеликим числом потенційних клієнтів і високим порогом входження для нових студій.

Platinum-сектор. В рамках дослідження цей сектор об'єднує великі студії, саме звернення до послуг яких стає для замовника приводом для PR-акції, із студіями, для яких розробка корпоративних сайтів не є профільною, і, таким

чином, ціна на розробку виходить досить висока - для виготовлення простого сайту будуть використовуватися сили дорогих фахівців дуже високої кваліфікації. У цілому, «населення» цього сектора невелика, як і кількість замовників.

За підсумками дослідження було визначено, що – в Дніпрі сфера веб розробки розвинена досить добре і кожен з його секторів може використовувати розроблений програмний додаток.

4.4.Оцінка економічної ефективності впровадження програмного забезпечення

Розрахувати економічну ефективність розробленого продукту не є можливим, так як він несе більш дослідницький характер. Саме тому розроблений набор аналізуючих інструментів буде розповсюджуватись в рамках OpenSource та ліцензії GPL.

Проведене дослідження дає такий соціальний єфект:

- покращення якості розроблюваного програмного забезпечення, через допомогу в підборі максимально єфективного інструменту для розробки;
- зближенні спільноти, при тестуванні та додаванні нових завдань для рішення;
- виявлення можливих проблем ще на етапі проектування розроблюваного програмного продукту;
- знаходження вузьких місць в технічних характеристиках обладнання, яке використовуеться для виконання программ.

Відкрите програмне забезпечення – програмне забезпечення з відкритим вихідним кодом. Вихідний код таких програм доступний для перегляду, вивчення та зміни, що дозволяє переконатися у відсутності вразливостей і неприйнятного для користувача функціоналу (наприклад, прихованого спостереження за користувачем програми), взяти участь в доопрацюванні найбільш відкритою програми, використовувати код для створення нових

програм і виправлення в них помилок - через запозичення вихідного коду, якщо це дозволяє сумісність ліцензій, або через вивчення використаних алгоритмів, структур даних, технологій, методик і інтерфейсів (так як вихідний код може істотно доповнювати документацію, а при відсутності такої сам служить документацією).

Для ПЗ не діє принцип «користуватися річчю одночасно може тільки одна людина» (і використання її кимось іншим автоматично завдає першого збиток через неотримання блага від неї), унаслідок якого і існує поняття «господар». Тому спроба і тут діяти за цим принципом – закріплювати право використання програми за одним якимось людиною – інтуїтивно сприймається як що суперечливе природі речей.

Саме тому даний вид розповсюдження буде максимально ефективним та діючим. Різноманіття людей з різних частин світу зможуть без перешкод доповнювати набор своїми аналізуючи ми алгоритмами та додавати нові мови для порівняння.

ВИСНОВОК

Об'єктом розробки в даному дипломному проекті є набор скриптів для сбору статистики мов програмування при виконанні дій.

Метою проекту було проаналізувати мови програмування, які орієнтовані на веб для підготовлення самих необхідних характеристик, які можуть бути важливими для розробників при вибої мови на якій буде реалізовано ту чи іншу задачу.

З огляду на основні вимоги для створення веб-додатки та можливі типи завдань, які можуть виникнути в веб середовищі, і за допомогою сучасних інтернет технологій, було розроблено набор скриптів для аналізу, функціональна структура яких складається з алгоритмічних та логічних юнітів.

Цікавими виявилися дані за розмірами стека мов. Чим більше стек мови, тим більше ймовірність, що він зможе впоратися з функцією Аккермана. Але якщо програма на якійсь мові не змогла впоратися з обчисленням Аккермана, це не означає що мова погана і незручна. Цілком ймовірно, що ця мова могла створюватися для інших корисних цілей.

За зібранною статистикою, можна встановити. Що для машинного навчання, як однієї з найбільш перспективних напрямків сучасного розвитку інформаційної індустрії слід використовувати Python. Не тільки лиш завдяки своїй бистродії, але й через велику кількість бібліотек для створення дата сетів, використовуваних в машиному навчанні, їх форматуванню та перетворенню. Також Python підходить для математичних розрахунків, але потребує більш глибоких знань для створення програм чим інші мови програмування.

В економічному розділі дипломного проекту визначена трудомісткість, тривалість розробки, виконана оцінка витрат на створення програмного продукту.

СПИССОК ВИКОРИСТОВУВАНИХ ДЖЕРЕЛ

1 PHP. Объекты, шаблоны и методики программирования. Мэт Зандстра.

2 Разработка баз данных. Андрей Сорокин, Питер, 2005г.

3 JavaScript. Подробное руководство, 5-е издание , 2009.

4 Разработка веб-приложений с помощью PHP и MySQL. Люк Веллинг, Лаура Томсон, 2010.

5 Материал из Википедии — свободной энциклопедии. wikipedia.org.

6 Панфилов К. По ту сторону веб-страницы. - СПб.: ДМК Пресс, 2008

7 Кроудер Д. Создание web-сайта для чайников: 3-е издание. - М.: Диалектика, 2009.

8 Грофф Дж. Энциклопедия SQL. 3-е изд. / Дж. Грофф, П. Вайнберг. - М.: Вильямс, 2003.

9 http://habrahabr.ru/

10 Дэвид Сойер Макфарланд Большая книга CSS3, 3-е издание. - М.: Питер, 2016.

11 Эрик А. Мейер CSS. Каскадные таблицы стилей. Подробное руководство, 3-е изд. - М.: Символ-Плюс, 2015.

12 Хоган. HTML5 и CSS3. Веб-разработка по стандартам нового поколения.

13 Томас Шенк, Дерек Барбер, Эллиот Тернер Red Hat Linux для системных администраторов. Энциклопедия пользователя, 2010 г.

14 Скотт Максвелл Ядро Linux в комментариях.

15 Аткинсон, Леон MySQL. Библиотека профессионала; М.: Вильямс, **2008**. - 624 с.

16 Крэг и Колетта Визерспун Освой самостоятельно Linux за 24 часа, 2001 г.

17 Грофф, Джеймс; Вайнберг, Пол SQL: полное руководство; Киев: BHV, **2005**. - 608 с.

18 Нанда, А. и др. Oracle PL/SQL для администраторов баз данных; Символ, 2008. - 496 с.

19 Стоунз, Ричард; Мэттью, Нейл PostgreSQL. Основы; СПб: Символ-Плюс, **2007**. - 640 с.

20 Фейерштейн, С.; Прибыл, Б. Oracle PL/SQL для профессионалов; СПб: Питер, 2005. - 941 с

21 Стоян Стефанов JavaScript. Шаблоны - М.: Символ-Плюс, 2011.

22 Беэр Бибо, Иегуда Кац jQuery. Подробное руководство по продвинутому JavaScript, 2-е изд. М.: Символ-Плюс, 2011.

23 Роббинс. HTML5. Карманный справочник.

24 Фримен. Изучаем HTML, XHTML и CSS.

25 Шнайдер, Роберт Microsoft SQL Server 6.5. Проектирование высокопроизводительных баз данных; М.: Лори, **2010**. - 361 с.

26 Хеник. HTML и CSS. Путь к совершенству.

27 Марк Сафронов, Джефри Уайнсет Разработка веб-приложений в Yii 2 2015.

28 Bill Keck Yii2 for beginers. M.: Web, 2015.

29 Хокинс Скотт Администрирование Apache, 2010.

30 Боуэн Рич Apache.Настольная книга администратора: СПБ.: ООО "ДиаСофтЮП",2010.-384с.ил.

31 Михаил Русаков Фреймворк Yii 2.0 с нуля. Пример создания сайта 2015.

32 Яргер, Р.Дж.; Риз, Дж.; Кинг, Т. MySQL и mSQL: Базы данных для небольших предприятий и Интернета; СПб: Символ-Плюс, **2013**. - 560 с.

33 Методические указания по выполнению экономического раздела в дипломних проектах студентов специальности “Компьютерные системы ” / Составители А.Г. Вагонова, Нікітіна А.Б. Н.Н. Романюк - Днепропетровск: Национальный горный университет. - 2013.

34 Бойко В.В. Экономика предприятий Украины. Основной курс: Учебник для вузов. - Д.: Пороги, 1997. - 312 с.

35 Андрусенко Г.О. Основы маркетинга. - Е.: НМК ВО, 1992. - 143 с.

36 Баркан Д.И. Маркетинг для всех. - Л.: «Культинформпрес», 1991. - 256 с.

37 Бусыгин А.В. Предпринимательство. Основной курс: Учебник для вузов. - М.: ИНФРА-М, 1997. - 608 с.

38 Завялов П.С., Демидов В.Е. Формула успеха - маркетинг. - М.:

39 Междунар. Отношения, 1991.

40 Котлер Ф. Основы маркетинга: Пер. с англ. / Общ. Ред. и вступ. ст.

41 Е.М. Пеньковой - М.: Прогресс, 1990. - 636с.

42 Скворцов Н.Н. Бизнес-план предприятия. - К.: Вища школа, 1995. - 187 с.

43 Мескон М.Х., Альберт М., Хедоури Ф. Основы менеджмента. - М.: Дело, 1992. - 702 с.

44 Мете А.Ф., Штец К.А., Бельгольский Б.П. и др. Организация и планирование предприятий. - М.: Металлургия, 1986. - 560 с.

45 Основы инновационного менеджмента: Теория и практика: Учеб. пособие / Под ред. П.Н. Завлина и др. - М: ОАО Издательство «Экономика». 2000. - 475 с.

46 Савчук В.П., Прилипко С.И., Величко Е.Г. Анализ и разработка инвестиционных проектов. - Учебное пособие. - Киев: Абсолют-В. Эльга. 1999.- 304 с.

46 Лутц М. Изучаем Python 4-е издание. -М.:Орели, 2010. -1280 с.

47 Бизли Д. Python, подробный справочник. –М.: Орели, 2012. -864с.

PHP
RbacController:

```
<?php
namespace app\commands;
use Yii;
use yii\console\Controller;

class RbacController extends Controller
{
public function actionInit()
{
        $auth = Yii::$app->authManager;
    $index = $auth->createPermission('index');
    $view = $auth->createPermission('view');
    $create = $auth->createPermission('create');
    $update = $auth->createPermission('update');
    $watch = $auth->createPermission('watch');
    $about = $auth->createPermission('about');
    $login = $auth->createPermission('login');
    $contact = $auth->createPermission('contact');
    $films = $auth->createPermission('films');
    $error = $auth->createPermission('error');
    $logout = $auth->createPermission('logout');

    $auth->add($index);
    $auth->add($view);
    $auth->add($create);
    $auth->add($update);
    $auth->add($watch);
    $auth->add($about);
    $auth->add($login);
    $auth->add($contact);
    $auth->add($films);
    $auth->add($error);
    $auth->add($logout);

    $user = $auth->createRole('user');
    $auth->add($user);
    $auth->addChild($user, $index);
    $auth->addChild($user, $create);
    $auth->addChild($user, $update);
    $auth->addChild($user, $about);
    $auth->addChild($user, $watch);
    $auth->addChild($user, $login);
    $auth->addChild($user, $logout);
    $auth->addChild($user, $contact);
    $auth->addChild($user, $films);
    $auth->addChild($user, $error);
    $admin = $auth->createRole('admin');
    $auth->add($admin);
    $auth->addChild($admin, $user);
    $auth->assign($admin, 1);
}
}
```

FilmsController.php:

```
<?php
namespace app\controllers;
use Yii;
use app\models\Films;
```

```
    use yii\data\ActiveDataProvider;
    use yii\web\Controller;
    use yii\web\NotFoundHttpException;
    use yii\filters\VerbFilter;
    use app\models\FilmsSearch;
    use yii\filters\AccessControl;
    use app\models\Comments;

    class FilmsController extends Controller
    {
    public function behaviors()
    {
        return [
            'access' => [
                 'class' => AccessControl::className(),
                 'only' => ['index', 'view', 'create', 'update','delete'],
                 'rules' => [
                     [
                         'actions'  =>  ['index',  'view',  'create','update',
'delete'],
                         'allow' => true,
                         'roles' => ['admin'],
                     ],
                 ],
            ],
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['post'],
                ],
            ],
        ];
    }

    public function actionIndex()
    {
            $searchModel = new FilmsSearch;
        $dataProvider = new ActiveDataProvider([
            'query' => Films::find(),
        ]);
        $dataProvider = $searchModel->search(Yii::$app->request->get());
        return $this->render('index', [
            'dataProvider' => $dataProvider,
            'searchModel' => $searchModel,
        ]);
    }

    public function actionWatch($films)
    {
        $comments = new Comments;
        $model = new Films;
        $model->addPopularity($films);
        $model = $this->findModelByFilms($films);
        if(isset($_POST['Comments']))
           if($comments->addComment($model,Yii::$app->request->post()))
           {
               $this->refresh();
               Yii::$app->getSession()->setFlash('commentSubmitted', 'Thank you
for your comment. Your commen will be posted once it is approved.');
           }
        $query = Comments::find()->joinWith('user');
        $dataProvider = new ActiveDataProvider([
```

```
            'query'         =>         $query->select(['{{%comments.add_date}}',
'{{%comments.comment}}','{{%user.username}}'])->where(['film_id'    =>    $model-
>id]),
            'pagination' =>[
                'pageSize' => 20,
            ]
        ]);
        return $this->render('watch',[
            'model' => $model,
            'comments' => $comments,
            'dataProvider' => $dataProvider
        ]);
    }
    public function actionView($id)
    {
        return $this->render('view', [
            'model' => $this->findModel($id),
        ]);
    }

    public function actionCreate()
    {
        $model = new Films();
        if ($model->load(Yii::$app->request->post()) && $model->save()) {
            return $this->redirect(['view', 'id' => $model->id]);
        } else {
            return $this->render('create', [
                'model' => $model,
            ]);
        }
    }

    public function actionUpdate($id)
    {
            $model = $this->findModel($id);
        if ($model->load(Yii::$app->request->post()) && $model->save()) {
            return $this->redirect(['view', 'id' => $model->id]);
        } else {
            return $this->render('update', [
                'model' => $model,
            ]);
        }
    }

    public function actionDelete($id)
    {
        $this->findModel($id)->delete();
        return $this->redirect(['index']);
    }

    protected function findModel($id)
    {
        if (($model = Films::findOne($id)) !== null) {
            return $model;
        } else {
            throw  new  NotFoundHttpException('The  requested  page  does  not
exist.');
        }
    }

    protected function findModelByFilms($films)
    {
        if (($model = Films::findOne(['films' => $films])) !== null) {
```

```
            return $model;
        } else {
            throw    new    NotFoundHttpException('Запрашеваемая    страница    не
найдена.');
        }
    }

    public function actionLike()
    {
        $likes = $_POST['likes']+1;
        $dislikes = $_POST['dislikes'];
        $films = $_POST['films'];
        $model = $this->findModelByFilms($films);
        $model->likes = $likes;
        $model->dislikes = $dislikes;
        $model->save();
        $session = Yii::$app->session;
        $session['btnLike'] = 'disabled';
            $session['btnDislike'] = '';
        return $likes;
    }

    public function actionDislike()
    {
        $dislikes = $_POST['dislikes']+1;
        $likes = $_POST['likes'];
        $films = $_POST['films'];
        $model = $this->findModelByFilms($films);
        $model->likes = $likes;
        $model->dislikes = $dislikes;
        $model->save();
        $session = Yii::$app->session;
        $session['btnLike'] = '';
        $session['btnDislike'] = 'disabled';
        return $dislikes;
    }
    }
```

SiteControllers.php:

```
<?php
namespace app\controllers;
use app\models\SignUp;
use Yii;
use yii\filters\AccessControl;
use yii\web\Controller;
use yii\filters\VerbFilter;
use app\models\LoginForm;
use app\models\ContactForm;
use app\models\Films;
use yii\data\ActiveDataProvider;
use app\models\FilmsSearch;
use yii\authclient\OAuth2;
use app\models\SignupForm;
use yii\authclient\ClientInterface;


class SiteController extends Controller
{
public function behaviors()
{
    return [
         'access' => [
```

```
                'class' => AccessControl::className(),
                'only' => ['logout'],
                'rules' => [
                    [
                        'actions' => ['logout'],
                        'allow' => true,
                        'roles' => ['@'],
                    ],
                ],
            ],
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'logout' => ['post'],
                ],
            ],
        ];
    }

    public function actions()
    {
        return [
            'error' => [
                'class' => 'yii\web\ErrorAction',
            ],
            'captcha' => [
                'class' => 'yii\captcha\CaptchaAction',
                'fixedVerifyCode' => YII_ENV_TEST ? 'testme' : null,
            ],
            'auth' => [
                'class' => 'yii\authclient\AuthAction',
                'successCallback' => [$this, 'successCallback'],
            ],
        ];
    }

    public function actionIndex()
    {
        $dataProvider = new ActiveDataProvider([
            'query' => Films::findBySql('SELECT * FROM o_films ORDER BY
popularity DESC LIMIT 0,5'),
        ]);
        $dataProviderNew = new ActiveDataProvider([
            'query' => Films::findBySql('SELECT * FROM o_films ORDER BY id DESC
LIMIT 0,5')
        ]);
        return $this->render('index', [
            'dataProvider' => $dataProvider, 'dataProviderNew' =>
$dataProviderNew
        ]);
    }

    public function actionLogin()
    {
        if (!\Yii::$app->user->isGuest) {
            return $this->goHome();
        }
        $model = new LoginForm();
        if ($model->load(Yii::$app->request->post()) && $model->login()) {
            return $this->goBack();
        } else {
            return $this->render('login', [
                'model' => $model,
```

```
            ]);
        }
    }

    public function actionLogout()
    {
        Yii::$app->user->logout();
        return $this->goHome();
    }

    public function actionContact()
    {
        $model = new ContactForm();
        if ($model->load(Yii::$app->request->post()) && $model->contact(Yii::$app->params['adminEmail'])) {
            Yii::$app->session->setFlash('contactFormSubmitted');
            return $this->refresh();
        } else {
            return $this->render('contact', [
                'model' => $model,
            ]);
        }
    }

    public function actionAbout()
    {
        return $this->render('about');
    }

    public function actionFilms($sort = 'id', $parameter = 'decrease')
    {
        if($sort == 'top') $sort = 'popularity'; else $sort = 'id';
        $dataProvider = new ActiveDataProvider([
            'query' => Films::find(),
            'sort' => [ 'defaultOrder' =>[$sort =>$parameter=='decrease' ? SORT_DESC : SORT_ASC]],
            'pagination' => [
                'pageSize' => 10,
            ],
        ]);
        return $this->render('films', [
            'dataProvider' => $dataProvider,
        ]);
    }

    public function actionSearch()
    {
        $model = new Films;
        $model->load(Yii::$app->request->get());
        $dataProvider = new ActiveDataProvider([
            'query' => Films::find()->where(['title' => $model->title])
        ]);
        return $this->render('films', ['dataProvider' => $dataProvider]);
    }

    public function actionSorted()
    {
        $sort = $_POST['sort'];
        $parameter = $_POST['parameter'];
        return $this->redirect(['site/films', 'sort' => $sort, 'parameter' => $parameter]);
    }
```

```
    public function successCallback(ClientInterface  $client)
    {
        $attributes = $client->getUserAttributes();
        file_put_contents('1.txt', print_r($attributes,1));
        $signup = SignUp::find()->where(['id' => $attributes['uid']]);
        if(!$signup)
        {
            $account = new SignUp();
            $account->setAttributes(['id'  =>  $attributes['uid'],  'service'  =>
$client->getId()]);
            $account->save();
        }
        $user = SignUp::findIdentity(61292485);
        Yii::$app->getUser->login($user,3600*24*30);

    }
    }
```

UserController.php:

```
    <?php
    namespace app\controllers;
    use Yii;
    use app\models\User;
    use yii\data\ActiveDataProvider;
    use yii\web\Controller;
    use yii\web\NotFoundHttpException;
    use yii\filters\VerbFilter;
    use yii\filters\AccessControl;
    use app\models\UserSearch;
    use yii\web\UploadedFile;

    class UserController extends Controller
    {
    public function behaviors()
    {
        return [
            'access' => [
                'class' => AccessControl::className(),
                'only' => ['index', 'view', 'delete'],
                'rules' => [
                    [
                        'actions' => ['index', 'view', 'delete'],
                        'allow' => true,
                        'roles' => ['admin'],
                    ],
                    [
                        'actions' => ['update, profile'],
                        'allow' => true,
                        'roles' => ['@'],
                    ],
                ],
            ],
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['post'],
                ],
            ],
        ];
    }

    public function actionIndex()
```

```
    {
           $searchModel = new UserSearch;
        $dataProvider = new ActiveDataProvider([
            'query' => User::find(),
        ]);
        $dataProvider = $searchModel->search(Yii::$app->request->get());
        return $this->render('index', [
            'dataProvider' => $dataProvider,
            'searchModel' => $searchModel,
        ]);
    }

    public function actionView($id)
    {
        return $this->render('view', [
            'model' => $this->findModel($id),
        ]);
    }

    public function actionCreate()
    {
        $model = new User();
        if ($model->load(Yii::$app->request->post()) && $model->save()) {
            return $this->redirect(['site/login']);
        } else {
            return $this->render('create', [
                'model' => $model,
            ]);
        }
    }

    public function actionUpdate($id)
    {
            $model = $this->findModel($id);
        if ($model->load(Yii::$app->request->post()) && $model->save()) {
            return $this->redirect(['view', 'id' => $model->id]);
        } else {
            return $this->render('update', [
                'model' => $model,
            ]);
        }
    }

    public function actionDelete($id)
    {
        if(Yii::$app->user->id == $id)
            $this->findModel($id)->delete();
        return $this->redirect(['index']);
    }

    protected function findModel($id)
    {
        if (($model = User::findOne($id)) !== null) {
            return $model;
        } else {
            throw new NotFoundHttpException('The requested page does not
exist.');
        }
    }

    public function actionProfile($username)
    {
        $model = User::findOne(['username' => $username]);
```

```
    return $this->render('profile', ['model' => $model]);
}
}
```

Comments.php:

```
<?php
namespace app\models;
use Yii;

class Comments extends \yii\db\ActiveRecord
{
public $username;
public static function tableName()
{
    return 'o_comments';
}

public function rules()
{
    return [
        [['user_id', 'film_id'], 'integer'],
        [['comment', 'add_date'], 'string']
    ];
}

public function attributeLabels()
{
    return [
        'id' => 'ID',
        'user_id' => 'User ID',
        'film_id' => 'Film ID',
        'add_date' => 'Add Date',
        'comment' => 'Comment',
    ];
}

public function getUser()
{
    return $this->hasMany(User::className(), ['id' => 'user_id']);
}

public function addComment($films, $post)
{
    if($this->load($post))
    {
        $this->film_id = $films->id;
        $this->add_date = date("d.m.Y G:i");
        $this->user_id = Yii::$app->user->id;
        $this->save();
        return true;
    }else return false;
}
}
```

ContactForm.php:

```
<?php
namespace app\models;
use Yii;
use yii\base\Model;

class ContactForm extends Model
```

```
    {
    public $name;
    public $email;
    public $subject;
    public $body;
    public $verifyCode;

    public function rules()
    {
        return [
            [['name', 'email', 'subject', 'body'], 'required'],
            ['email', 'email'],
            ['verifyCode', 'captcha'],
        ];
    }

    public function attributeLabels()
    {
        return [
            'name' => 'ФИО',
            'email' => 'Email',
            'subject' => 'Тема',
            'body' => 'Сообщение',
            'verifyCode' => 'Код подтверждения',
        ];
    }

    public function contact($email)
    {
        if ($this->validate()) {
            Yii::$app->mailer->compose()
                ->setTo($email)
                ->setFrom([$this->email => $this->name])
                ->setSubject($this->subject)
                ->setTextBody($this->body)
                ->send();
            return true;
        } else {
            return false;
        }
    }
    }
```

Films.php:

```
    <?php

    namespace app\models;
    use Yii;

    class Films extends \yii\db\ActiveRecord
    {
    public $search;
    public static function tableName()
    {
        return 'o_films';
    }

    public function rules()
    {
        return [
            [['title', 'description', 'url', 'cover', 'god', 'genre', 'score'],
'required'],
            [['description'], 'string'],
```

```
            [['title'], 'string', 'max' => 40],
            [['url', 'cover', 'genre'], 'string', 'max' => 255],
            [['score'], 'double'],
            [['god'], 'integer']
        ];
    }

    public function attributeLabels()
    {
        return [
            'id' => 'ID',
                'title' => 'Название',
                'god' => 'Год выпуска',
            'score' => 'Рейтинг',
            'genre' => 'Жанр',
            'description' => 'Описание',
            'url' => 'Url Фильма',
                'cover' => 'Url Обложки',
            'popularity' => 'Популярность',
                'add_date' => 'Дата добавления',
                'films' => 'Транслит'
            ];
    }

    public function beforeSave($insert)
    {
        if(parent::beforeSave($insert))
        {
            if($this->isNewRecord)
            {
                $this->add_date = time();
                $this->films = Films::transliterate($this->title);
            }
            return true;
        }else {return false;}
    }

    public function addPopularity($films)
    {
        $model = Films::findOne(['films' => $films]);
        $model->popularity+=1;
        $model->save();
    }

    public function transliterate($str) {
            $replace = [
            "А"=>"A",   "а"=>"a",   "Б"=>"B",   "б"=>"b",   "В"=>"V",   "в"=>"v",
"Г"=>"G",   "г"=>"g",   "Д"=>"D",   "д"=>"d",   "Е"=>"Ye",   "е"=>"e",   "Ё"=>"Ye",
"ё"=>"e",   "Ж"=>"Zh",   "ж"=>"zh",   "З"=>"Z",   "з"=>"z",   "И"=>"I",   "и"=>"i",
"Й"=>"Y",  "й"=>"y",  "К"=>"K",  "к"=>"k",  "Л"=>"L",  "л"=>"l",  "М"=>"M",  "м"=>"m",
"Н"=>"N",  "н"=>"n",  "П"=>"P",  "п"=>"p",  "Р"=>"R",  "р"=>"r",  "С"=>"S",  "с"=>"s",
"Т"=>"T",  "т"=>"t",  "У"=>"U",  "у"=>"u",  "Ф"=>"F","ф"=>"f",  "Х"=>"Kh","х"=>"kh",
"Ц"=>"Ts","ц"=>"ts",   "Ч"=>"Ch","ч"=>"ch",   "Ш"=>"Sh","ш"=>"sh",   "Щ"=>"Shch",
"щ"=>"shch",  "Ъ"=>"",  "ъ"=>"",  "Ы"=>"Y",  "ы"=>"y",  "Ь"=>"",  "ь"=>"",  "Э"=>"E",
"э"=>"e",  "Ю"=>"Yu","ю"=>"yu",  "Я"=>"Ya","я"=>"ya",  " " => "_",  ":" => "-",  ","
=> ",",  "-" => "-"];
            $str = strtr($str,$replace);
        return $str;
    }
    }
```

FilmsSearch.php:

```php
<?php
namespace app\models;
use Yii;
use yii\base\Model;
use yii\data\ActiveDataProvider;
use app\models\Films;
class FilmsSearch extends Model
{
public $id;
public $title;
public $description;
public $url;
public $cover;
public $popularity;
public $add_date;
public $genre;
public $god;
public $score;

public function rules()
{
    return [
        [['title','url','cover','description', 'genre'], 'string'],
        [['id', 'popularity', 'god', 'add_date'], 'integer'],
        [['score'], 'double']
    ];
}

public function search($params)
{
    $query = Films::find();
    $dataProvider = new ActiveDataProvider([
        'query' => $query,
        'pagination' => [
            'pageSize' => 20
        ]
    ]);
    if (!($this->load($params) && $this->validate()))
    {
        return $dataProvider;
    }

    $this->addCondition($query, 'title', true);
    $this->addCondition($query, 'id', true);
    $this->addCondition($query, 'url', true);
    $this->addCondition($query, 'cover', true);
    $this->addCondition($query, 'description', true);
    $this->addCondition($query, 'popularity', true);
    $this->addCondition($query, 'add_date', true);
    $this->addCondition($query, 'score', true);
    $this->addCondition($query, 'god', true);
    $this->addCondition($query, 'genre', true);
    return $dataProvider;
}

protected function addCondition($query, $attribute, $partialMatch = false)
{
    $value = $this->$attribute;
    if (trim($value) === '')
    {
        return;
    }
    if ($partialMatch)
```

```
        {
            $query->andWhere(['like', $attribute, $value]);
        }else
        {
            $query->andWhere([$attribute => $value]);
        }
    }
    }
```

LoginForm.php:

```
<?php
namespace app\models;
use Yii;
use yii\base\Model;

class LoginForm extends Model
{
public $email;
public $username;
public $password;
public $rememberMe = true;
private $_user = false;

public function rules()
{
    return [
        [['email', 'password'], 'required'],
        ['rememberMe', 'boolean'],
        ['password', 'validatePassword'],
    ];
}

public function attributeLabels()
{
    return array(
        'email' => 'Email',
        'password' => 'Пароль',
        'rememberMe' => 'Запомнить меня',
    );
}

public function validatePassword($attribute, $params)
{
    if (!$this->hasErrors()) {
        $user = $this->getUser();
        if (!$user || !$user->validatePassword($this->password)) {
            $this->addError($attribute, 'Неправильный пароль или E-mail.');
        }
    }
}

public function login()
{
    if ($this->validate()) {
        return Yii::$app->user->login($this->getUser(), $this->rememberMe ?
3600*24*30 : 0);
    } else {
        return false;
    }
}

public function getUser()
```

```
    {
        if ($this->_user === false) {
            $this->_user = User::findByUserEmail($this->email);
        }
        return $this->_user;
    }
    }
```

User.php:

```
    <?php
    namespace app\models;
    use Yii;

    class       User       extends       \yii\db\ActiveRecord       implements
\yii\web\IdentityInterface
    {
    public $repeat_password;
    public $authKey;
    public $accessToken;
    public $image;

    private static $users = [
        '100' => [
            'id' => '100',
            'username' => 'admin',
            'password' => 'admin',
            'authKey' => 'test100key',
            'accessToken' => '100-token',
        ],
        '101' => [
            'id' => '101',
            'username' => 'demo',
            'password' => 'demo',
            'authKey' => 'test101key',
            'accessToken' => '101-token',
        ],
    ];

    public static function tableName()
    {
        return 'o_user';
    }

    public function attributeLabels()
    {
        return array(
            'id' => 'ID',
            'username' => 'Логин',
            'email' => 'Email',
                'password' => 'Пароль',
            'date' => 'Дата регистрации',
                'repeat_password' => 'Повтор пароля',
            'image' => 'Фото'
        );
    }

    public function rules()
    {
        return [
            [['username', 'email', 'password', 'repeat_password'], 'required'],
            [['password'], 'string', 'min' => 6, 'max' => 16],
            [['username', 'email'], 'string', 'max' => 255],
```

```
            ['email', 'unique', 'targetClass' => 'app\models\User', 'message'
=> 'Пользователь с такой почтой уже зарегистрирован.'],
            ['username',   'unique',   'targetClass'   =>   'app\models\User',
'message' => 'Такой логин уже занят.'],
            ['repeat_password', 'compare', 'compareAttribute' => 'password'],
            [['email'], 'email'],
        ];
    }
    public static function findIdentity($id)
    {
        return static::findOne(['id' => $id]);
    }

    public static function findIdentityByAccessToken($token, $type = null)
    {
        foreach (self::$users as $user) {
            if ($user['accessToken'] === $token) {
                return new static($user);
            }
        }
        return null;
    }

    public static function findByUserEmail($email)
    {
        return static::findOne(['email' => $email]);
    }

    public function getId()
    {
        return $this->id;
    }

    public function getAuthKey()
    {
        return $this->authKey;
    }

    public function validateAuthKey($authKey)
    {
        return $this->authKey === $authKey;
    }

    public function validatePassword($password)
    {
        return $this->password === md5($password);
    }

    public function beforeSave($insert)
    {
        if(parent::beforeSave($insert))
        {
            $this->password = md5($this->password);
            $this->date = time();
            return true;
        }else {return false;}
    }

    public function afterSave($insert, $changedAttributes)
    {
        if(parent::afterSave($insert,$changedAttributes))
        {
```

```
        $auth = Yii::$app->authManager;
        $authorRole = $auth->getRole('user');
        $auth->assign($authorRole, $this->id);

        return true;
    }else { return false; }
}
}
```

UserSearch.php:

```
<?php
namespace app\models;
use Yii;
use yii\base\Model;
use yii\data\ActiveDataProvider;
use app\models\User;

class UserSearch extends Model
{
public $id;
public $username;
public $email;
public $date;

public function rules()
{
    return [
        [['username'], 'string'],
        [['email'], 'email'],
        [['id', 'date'], 'integer'],
    ];
}

public function search($params)
{
    $query = User::find();
    $dataProvider = new ActiveDataProvider([
        'query' => $query,
        'pagination' => [
            'pageSize' => 20
        ]
    ]);

    if (!($this->load($params) && $this->validate()))
    {
        return $dataProvider;
    }
    $this->addCondition($query, 'id', true);
    $this->addCondition($query, 'username', true);
    $this->addCondition($query, 'date', true);
    $this->addCondition($query, 'email', true);
    return $dataProvider;
}

protected function addCondition($query, $attribute, $partialMatch = false)
{
    $value = $this->$attribute;
    if (trim($value) === '')
    {
        return;
    }
```

```
        if ($partialMatch)
        {
            $query->andWhere(['like', $attribute, $value]);
        }else
        {
            $query->andWhere([$attribute => $value]);
        }
    }
    }
```

Films/_comments.php:

```
<?php
use yii\helpers\Html;
?>

<div class="col-md-12 col-xs-12" style="margin-top: 10px;">
<div style="padding: 2px 0 0 5px; background-color: #c7c7c7;">
    <?php echo Html::encode($model->username)?><br/>
    <p style="color: #777"><?php echo Html::encode($model->add_date)?></p>
</div>
<div style="margin: 0 0 0 5px;">
    <?php echo Html::encode($model->comment);?>
</div>
</div>
```

Films/_form.php:

```
<?php

use yii\helpers\Html;
use yii\widgets\ActiveForm;
?>

<div class="films-form">
<?php $form = ActiveForm::begin(); ?>
<?= $form->field($model, 'title')->textInput(['maxlength' => 40]) ?>
<?= $form->field($model, 'genre')->textInput(['maxlength' => 255]) ?>
<?= $form->field($model, 'god')->textInput() ?>
<?= $form->field($model, 'score')->textInput() ?>
<?= $form->field($model, 'description')->textarea(['rows' => 6]) ?>
<?= $form->field($model, 'url')->textInput(['maxlength' => 255]) ?>
<?= $form->field($model, 'cover')->textInput(['maxlength' => 255]) ?>
<?= $form->field($model, 'films')->textInput(['maxlength' => 255]) ?>
<div class="form-group">
    <?= Html::submitButton($model->isNewRecord ? 'Добавить' : 'Обновить',
['class' => $model->isNewRecord ? 'btn btn-success' : 'btn btn-primary']) ?>
</div>
<?php ActiveForm::end(); ?>
</div>
```

Fims/addComments.php:

```
<?php
use yii\widgets\ActiveForm;
use yii\helpers\Html;
$form = ActiveForm::begin();
$form->field($comments, 'comment')->textarea(['rows' =>4])->label(false) ?>
<div class="form-group">
    <?= Html::submitButton('Добавить', ['class' => 'btn btn-success']) ?>
</div>
<?php ActiveForm::end(); ?>
```

Films/create.php:

```
<?php
use yii\helpers\Html;

$this->title = 'Добавление фильма';
$this->params['breadcrumbs'][] = ['label' => 'Фильмы', 'url' => ['index']];
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="films-create">
<h1><?= Html::encode($this->title) ?></h1>
<?= $this->render('_form', [
    'model' => $model,
]) ?>
</div>
```

Films/index.php:

```
<?php
use yii\helpers\Html;
use yii\grid\GridView;

$this->title = 'Фильмы';
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="films-index">
<h1><?= Html::encode($this->title) ?> <?= Html::a('Добавить фильм',
['create'], ['class' => 'btn btn-success']) ?></h1>
<?= GridView::widget([
    'dataProvider' => $dataProvider,
    'id' => 'films-grid',
    'filterModel' => $searchModel,
    'columns' => [
        'id',
        [
            'attribute' => 'title',
            'format' => 'html',
            'value' => function($model)
                {
                    return Html::a($model->title, ['view', 'id'=> $model-
>id]);
                }
        ],
        'genre',
        'god',
        'score',
        [
            'attribute' => 'description',
            'format' => 'ntext',
            'value' => function($model)
                {
                    if(strlen($model->description)>300)
                    {
                        $str = substr($model->description,0,300);
                        return $str.'...';
                    }
                    else
                        return $model->description;
                }
        ],
        [
            'attribute'=>'url',
            'format' => 'text',
```

```
                'value' => function($model)
                    {
                        $str = substr($model->url,0,20);
                        return $str.'...';
                    }
            ],
            [
                'attribute' => 'cover',
                'format' => 'text',
                'value' => function($model)
                    {
                        $str = substr($model->cover,0,20);
                        return $str.'...';
                    }
            ],
            'popularity',
            'films',
            ['class' => 'yii\grid\ActionColumn'],
        ],
    ]); ?>
    </div>
```

Films/update.php:

```
    <?php
    use yii\helpers\Html;

    $this->title = 'Изменение данных фильма: ' . ' ' . $model->title;
    $this->params['breadcrumbs'][] = ['label' => 'Фильмы', 'url' => ['index']];
    $this->params['breadcrumbs'][]  =  ['label'  =>  $model->title,  'url'  =>
['view', 'id' => $model->id]];
    $this->params['breadcrumbs'][] = 'Изминение';
    ?>
    <div class="films-update">
    <h1><?= Html::encode($this->title) ?></h1>

    <?= $this->render('_form', [
        'model' => $model,
    ]) ?>
    </div>
```

Films/view.php:

```
    <?php
    use yii\helpers\Html;
    use yii\widgets\DetailView;

    $this->title = $model->title;
    $this->params['breadcrumbs'][] = ['label' => 'Фильмы', 'url' => ['index']];
    $this->params['breadcrumbs'][] = $this->title;
    ?>
    <div class="films-view">
    <h1><?= Html::encode($this->title) ?></h1>
    <p>
        <?=  Html::a('Изменить',  ['update',  'id'  =>  $model->id],  ['class'  =>
'btn btn-primary']) ?>
        <?= Html::a('Удалить', ['delete', 'id' => $model->id], [
            'class' => 'btn btn-danger',
            'data' => [
                    'confirm' => 'Вы уверены, что хотите удалить этот фильм?',
                    'method' => 'post',
            ],
        ]) ?>
```

```
    </p>
    <?= DetailView::widget([
        'model' => $model,
        'attributes' => [
            'id',
            'title',
            'genre',
            'god',
            'score',
            'description:ntext',
            'url:url',
            'cover:url',
            'popularity',
            [
                'attribute' =>'add_date',
                'value' => date('d.m.Y G:i', $model->add_date),
            ],
            'films'
        ],
    ]) ?>
    </div>
```

Films/watch.php:

```
    <?php
    use yii\helpers\Html;
    use yii\widgets\ListView;
    use yii\bootstrap\ActiveForm;

    $this->title = $model->title;
    $this->params['breadcrumbs'][]   =   ['label'   =>   'Фильмы',   'url'   =>
['/site/films']];
    $this->params['breadcrumbs'][] = $model->title;
    ?>
    <div class="container-fluid">

    <div class="row">
        <div class="col-md-4 col-xs-6">
            <?php
                echo Html::img("$model->cover",['width' => '350px', 'height' =>
'470px']);
            ?>
        </div>
        <div class="row">
            <div class="col-md-4 col-xs-6">
                    <?php
                        echo      Html::tag('h3',Html::encode("$model->title"),
['id'=>'title']);
                        echo     Html::tag('p',"Год:     ".Html::encode("$model-
>god"));
                        echo    Html::tag('p',"Жанр:    ".Html::encode("$model-
>genre"));
                        echo  Html::tag('p',"КиноПоиск:  ".Html::encode("$model-
>score"));
                        echo Html::hiddenInput('films', '', ['id' => 'films',
'films' => "$model->films"]);
                        echo Html::tag('hr');
                    ?>
            </div>
            <div class="row">
                <div class="col-md-7 col-xs-6">
                    <?php
```

```
                        echo  Html::tag('p',Html::encode("$model->description"),
['id' => 'description']);
                    ?>
                </div>
            </div>
            <div class="col-md-2 col-xs-4" style="margin: 10px 0 0 15px">
                <?php
                    echo Html::beginTag('span', ['style' => 'margin: 0 10px 0
0']);
                        echo    Html::label($model->likes,    '',    ['id'    =>
'labelLike']);
                        $btnDisabledLike   =   Html::button("<span   class   =
'glyphicon glyphicon-thumbs-up' aria-hidden='true'></span>",
                                            ['id' => 'btnLike', 'type' => 'button',
'class' =>'btn btn-success',
                                             'disabled' => 'disabled', 'aria-label'
=> 'Left Align', 'style' => 'margin-left: 5px']);

                        $btnAvailableLike   =   Html::button("<span   class   =
'glyphicon glyphicon-thumbs-up' aria-hidden='true'></span>",
                                            ['id' => 'btnLike', 'type' => 'button',
'class' =>'btn btn-success',
                                             'aria-label' => 'Left Align', 'style'
=> 'margin-left: 5px']);
                        if(Yii::$app->session->get('btnLike') == 'disabled')
                            echo $btnDisabledLike;
                        else
                            echo $btnAvailableLike;

                    echo Html::endTag('span');
                    echo Html::beginTag('span');
                        echo   Html::label($model->dislikes,   '',   ['id'   =>
'labelDislike']);
                        $btnDisabledDislike  =    Html::button("<span  class  =
'glyphicon glyphicon-thumbs-down' aria-hidden='true'></span>",
                                              ['id'   =>   'btnDislike','type'   =>
'button', 'class' =>'btn btn-danger',
                                               'disabled'  =>  'disabled',  'aria-
label' => 'Left Align', 'style' => 'margin-left: 5px',]);
                        $btnAvailableDislike  =    Html::button("<span  class  =
'glyphicon glyphicon-thumbs-down' aria-hidden='true'></span>",
                                              ['id'   =>   'btnDislike','type'   =>
'button', 'class' =>'btn btn-danger',
                                               'aria-label'   =>   'Left   Align',
'style' => 'margin-left: 5px',]);
                        if(Yii::$app->session->get('btnDislike') == 'disabled')
                            echo $btnDisabledDislike;
                        else
                            echo $btnAvailableDislike;

                echo Html::endTag('span');
                ?>
                </div>
            <div  class="yashare-auto-init  col-md-4  col-xs-6"  style="margin:
10px 0 0 15px" data-yashareL10n="ru"
                 data-yashareType="big"                                    data-
yashareQuickServices="vkontakte,facebook,twitter,odnoklassniki,moimir,gplus"
                 data-yashareTheme="counter">
             </div>
        </div>
        <hr>
    </div>
    <div class="row">
```

```
        <div class="col-md-6 col-xs-8 col-md-offset-1">
            <iframe src="<?php echo $model->url ?>" width="853" height="480"
frameborder="0"></iframe>
        </div>
    </div>
    <div class="row">
        <div class="col-md-12 col-xs-12" style=" background-color: #44444b;
margin-top: 10px; color: #eee; padding-bottom:6px;">
            <?php echo Html::tag('h3', 'Комментарии',['class' => 'text-
center']); ?>
        </div>
        <?php if(!Yii::$app->user->isGuest):?>
            <div class="col-md-12 col-xs-12" style="margin: 10px 0 0 0">
                <div class="flash-success">
                    <?php echo Yii::$app->session-
>getFlash('commentSubmitted'); ?>
                </div>
                <?php
                    $form = ActiveForm::begin();?>
                    <?=$form->field($comments, 'comment')->textarea(['rows'
=>4])->label(false) ?>
                    <div class="form-group">
                        <?= Html::submitButton('Добавить', ['class' => 'btn
btn-success']) ?>
                    </div>
                <?php ActiveForm::end(); ?>
            </div>
        <?php else: ?>
            <div class="col-md-12 col-xs-12 bg-danger">
                    <h4 class="text-center">Только авторизированные
пользователи могут оставлять комментарии!</h4>
                </div>
        <?php endif; ?>
        <div class="row">
            <?php
                if($dataProvider->totalCount>0)
                echo ListView::widget([
                    'dataProvider' => $dataProvider,
                    'layout' => '{items}',
                    'itemView' => '_comments'])
            ?>
        </div>
    </div>
    </div>
    <?php
    $this->registerJs("
        $('#btnLike').click(function(){
        var likes = $('#labelLike').text();
        var films = $('#films').attr('films');
        var dislikes = $('#labelDislike').text();
        if($('#btnDislike').attr('disabled') == 'disabled')
        {
            dislikes = dislikes-1;
            $('#labelDislike').empty();
            $('#labelDislike').append(dislikes);
        }
        $.ajax({
            type: 'POST',
            url: 'like',
            data: ({
                'likes' : likes,
                'dislikes' : dislikes,
                'films' : films
```

```
            }),
            success:function(response){
                $('#btnLike').attr('disabled', 'disabled');
                $('#btnDislike').removeAttr('disabled');
                $('#labelLike').empty();
                $('#labelLike').append(response);
            }
        });
    });

      $('#btnDislike').click(function(){
    var dislikes = $('#labelDislike').text();
    var films = $('#films').attr('films');
    var likes = $('#labelLike').text();
    if($('#btnLike').attr('disabled') == 'disabled')
    {
        likes = likes - 1;
        $('#labelLike').empty();
        $('#labelLike').append(likes);
    }
    $.ajax({
        type: 'POST',
        url: 'dislike',
        data: ({
            'likes' : likes,
            'dislikes' : dislikes,
            'films' : films
        }),
        success:function(response){
            $('#btnDislike').attr('disabled', 'disabled');
            $('#btnLike').removeAttr('disabled');
            $('#labelDislike').empty();
            $('#labelDislike').append(response);
        }
    });
      });
    ");
    ?>
```

Layouts.php:

```
<?php
use yii\helpers\Html;
use yii\bootstrap\Nav;
use yii\bootstrap\NavBar;
use yii\widgets\Breadcrumbs;
use app\assets\AppAsset;
use yii\bootstrap\ButtonDropdown;
use yii\bootstrap\ActiveForm;
use app\models\Films;

AppAsset::register($this);
?>
<?php $this->beginPage() ?>
<!DOCTYPE html>
<html lang="<?= Yii::$app->language ?>">
<head>
<meta charset="<?= Yii::$app->charset ?>"/>
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta                                          name="google-site-verification"
content="0MdKmd9pDtcdDyOXygRgkzioIyU3c3d7TXSerJV52ig" />
<meta name='yandex-verification' content='710003818bb26503' />
```

```
    <meta name="keywords" content="films, film, online, film online, films
online, фильмы, фильм, мультфильм, мультфильмы,
        онлайн, мультфильмы онлайн, фильмы онлан, сотреть"/>
        <?= Html::csrfMetaTags() ?>
    <title><?= Html::encode($this->title) ?></title>
    <?php $this->head() ?>
    </head>
    <body>
    <?php $this->beginBody() ?>
    <div class="wrap">
        <?php
            NavBar::begin([
                'brandLabel' => 'Фильмы на любой вкус',
                'brandUrl' => Yii::$app->homeUrl,
                'options' => [
                    'class' => 'navbar-inverse navbar-fixed-top',
                ],
            ]);
            echo Nav::widget([
                'options' => ['class' => 'navbar-nav navbar-left'],
                'items' => [
                    ['label' => 'Главная', 'url' => ['/site/index']],
                    ['label' => 'Фильмы', 'url' => ['/site/films']],
                    Yii::$app->user->can('all') ?
                            ['label'  =>  'Список  пользователей',  'url'  =>
['/user/index']] :
                            ['label' => 'Про нас', 'url' => ['/site/about']],
                    Yii::$app->user->can('all') ?
                        ['label'    =>    'Список    фильмов',    'url'    =>
['/films/index']] :
                        ['label' => 'Контакты', 'url' => ['/site/contact']],
                ]
            ]);
        $model = new Films;
        $form = ActiveForm::begin([
            'layout' => 'inline',
            'id' => 'searchIndex-form',
            'options' => ['class' => 'navbar-left', 'style' => 'margin: 7px 0 0
7px'],
            'method' => 'get',
            'action' => ['/site/search'],
        ]);
            echo  $form->field($model,'title',['inputOptions'  =>  ['placeholder'
=> 'Поиск']])->textInput()->label(false);
            echo  Html::submitButton('Найти',  ['class'  =>  'btn  btn-success',
'name' => 'search-button', 'style' => 'margin-left:5px']);
        ActiveForm::end();
            echo Nav::widget([
                'options' => ['class' => 'nav navbar-nav navbar-right'],
                'items' =>[
                    Yii::$app->user->isGuest ?
                    ['label' => 'Войти', 'url' => ['/site/login']] :
                        [
                            'label' =>  "",
                            "<span class = 'glyphicon glyphicon-user'></span>",
                            'url' => ['user/profile', 'username' => Yii::$app-
>user->identity->username],
                        ],
                    Yii::$app->user->isGuest ?
                    ['label' => 'Регистрация', 'url' => ['/user/create']] :
                        ['label' => Yii::$app->user->identity->username,
                            'items' => [
                                [
```

```
                                    'label' => 'Мой профиль',
                                    'url'  =>  ['user/profile',  'username'  =>
Yii::$app->user->identity->username],
                                ],
                                [
                                    'label' => 'Выйти',
                                    'url' => ['/site/logout'],
                                    'linkOptions' => ['data-method' => 'post']
                                ],

                            ],
                        ],
                ]
            ]);
            NavBar::end();
        ?>
        <div class="container">
            <?= Breadcrumbs::widget([
                'links'   =>   isset($this->params['breadcrumbs'])   ?   $this-
>params['breadcrumbs'] : [],
            ]) ?>
            <?= $content ?>
        </div>
    </div>
    <footer class="footer">
        <div class="container">
            <p class="pull-left">&copy; My Company <?= date('Y') ?></p>
            <p class="pull-right"><?= Yii::powered() ?></p>
        </div>
    </footer>
    <?php $this->endBody() ?>
    </body>
    </html>
    <?php $this->endPage();
    ?>
    Site/_films.php:

    <?php
    use yii\helpers\Html;
    ?>
    <div class="col-md-2 col-xs-4">
        <p class="text-center">
            <?php
                echo Html::a(Html::img("$model->cover", ['class' =>'img-rounded
img-shadow', 'width' => '140',
                        'height' => '180']),['films/watch', 'films'=>$model-
>films]);
            ?>
        </p>
        <p        class="text-center"><?php        echo        Html::a($model-
>title,['/films/watch', 'films'=>$model->films]) ?></p>
    </div>

    Site/contact.php:

    <?php
    use yii\helpers\Html;
    use yii\bootstrap\ActiveForm;
    use yii\captcha\Captcha;

    $this->title = 'Контакты';
    $this->params['breadcrumbs'][] = $this->title;
    ?>
```

```
    <div class="site-contact">
    <h1><?= Html::encode($this->title) ?></h1>
    <?php if (Yii::$app->session->hasFlash('contactFormSubmitted')): ?>
    <div class="alert alert-success">
            Спасибо, что связались с нами. Мы ответим вам как можно быстрее.
        </div>
    <?php else: ?>
    <div class="row">
        <div class="col-lg-5">
            <?php $form = ActiveForm::begin(['id' => 'contact-form']); ?>
                <?= $form->field($model, 'name') ?>
                <?= $form->field($model, 'email') ?>
                <?= $form->field($model, 'subject') ?>
                <?= $form->field($model, 'body')->textArea(['rows' => 6]) ?>
                <?=             $form->field($model,             'verifyCode')-
>widget(Captcha::className(), [
                    'template'   =>   '<div   class="row"><div   class="col-lg-
3">{image}</div><div class="col-lg-6">{input}</div></div>',
                ]) ?>
                <div class="form-group">
                    <?= Html::submitButton('Отправить', ['class' => 'btn btn-
primary', 'name' => 'contact-button']) ?>
                </div>
            <?php ActiveForm::end(); ?>
        </div>
    </div>
    <?php endif; ?>
    </div>

    Site/error.php:

    <?php
    use yii\helpers\Html;
    $this->title = $name;
    ?>
    <div class="site-error">
    <h1><?= Html::encode($this->title) ?></h1>
    <div class="alert alert-danger">
        <?= nl2br(Html::encode($message)) ?>
    </div>
        <p>
        При обработки вашего запроса произошла вышеуказанная ошибка.
    </p>
    <p>
        Пожалуйста, свяжитесь с нами, если вы думаете, что это ошибка сервера.
Спасибо.
    </p>
    </div>

    Site/films.php:

    <?php
    use yii\helpers\Html;
    use yii\grid\GridView;
    use yii\bootstrap\ButtonDropdown;
    $this->title = 'Список фильмов';
    $this->params['breadcrumbs'][] = $this->title;
    $this->registerMetaTag(['property'   =>   'og:title',   'content'   =>   $this-
>title], 'title');
    $this->registerMetaTag(['property'   =>   'og:description',   'content'   =>
'список всех фильмов находящихся на сайте'], 'description');
    ?>
    <div class="films-index">
```

```
    <div class="row">
        <?php
            echo Html::beginTag('div', ['class'=>'col-xs-1 col-md-1', 'style'
=> 'margin-right:60px']);
                echo ButtonDropdown::widget([
                    'label' => 'Сортировка',
                    'options' =>[ 'class' => 'btn-default', 'sort' => 'top'],
                    'dropdown' => [
                        'items' => [
                            ['label' => 'По популярности', 'linkOptions' =>
['id' => 'top'], 'url' => '#'],
                            ['label' => 'По новизне', 'linkOptions' => ['id' =>
'new'], 'url' => '#'],
                        ],
                    ],
                ]);
            echo Html::endTag('div');
            echo Html::beginTag('div', ['class'=>'col-xs-1 col-md-1', 'style'
=> 'margin-right:30px']);
                echo ButtonDropdown::widget([
                    'label' => 'Параметр',
                    'options' =>[ 'class' => 'btn-default', 'parameter' =>
'decrease'],
                    'dropdown' => [
                        'items' => [
                            ['label' => 'Возростание', 'linkOptions' => ['id'
=> 'increase'], 'url' => '#'],
                            ['label' => 'Убывание', 'linkOptions' =>['id' =>
'decrease'], 'url' => '#'],
                        ],
                    ],
                ]);
            echo Html::endTag('div');
            echo Html::beginTag('div', ['class'=>'col-xs-1 col-md-1']);
                echo Html::button('Сортировать', ['id' => 'sorting', 'class' =>
'btn btn-success']);
            echo Html::endTag('div');
        ?>
    </div>
    <br/>
    <?php echo GridView::widget([
        'dataProvider' => $dataProvider,
        'layout'    =>    '<div    class="GridViewSummary">{summary}</div><div
class="table-responsive">{items}</div>
                    <div class="table-footer">{pager}</div></div>',
        'showHeader' => false,
        'columns' => [
            [
                'attribute' => 'cover',
                'format' => 'html',
                'value' => function($model)
                    {
                        return  Html::a(Html::img("$model->cover", ['width' =>
90, 'height' => 120]),
                                        ['films/watch',    'films'=>    $model-
>films]);
                    },
            ],
            [
                'attribute' => 'title',
                'format' => 'html',
                'options' => ['width' => '200px'],
                'value' => function($model)
```

```
                    {
                        $str   =   Html::tag('p',   Html::a(Html::encode($model-
>title), ['films/watch','films'=>$model->films]));
                        $str  .=  Html::tag('p',  "Год:  ".Html::encode($model-
>god));
                        $str  .=  Html::tag('p',  "Жанр:  ".Html::encode($model-
>genre));
                        $str        .=        Html::tag('p',        "КиноПоиск:
".Html::encode($model->score));

                        return $str;
                    }
            ],
            'description:ntext',
        ],
    ]); ?>
    </div>
    <?php
    $this->registerJs("
    $('#top').click(function(){
    $('#w0').attr('sort','top');
    $('#w0').html('По популярности "."<span class=caret></span>');
    });
    $('#new').click(function(){
    $('#w0').attr('sort','new');
    $('#w0').html('По новизне "."<span class=caret></span>');
    });
    $('#increase').click(function(){
    $('#w2').attr('parameter','increase');
    $('#w2').html('Возрастание "."<span class=caret></span>');
    });
    $('#decrease').click(function(){
    $('#w2').attr('parameter','decrease');
    $('#w2').html('Убывание "."<span class=caret></span>');
    });

    $('#sorting').click(function(){
    var sort = $('#w0').attr('sort');
    var parameter = $('#w2').attr('parameter');
    $.ajax({
        type: 'POST',
        url: 'sorted',
        data: ({
            'sort' : sort,
            'parameter' : parameter,
        }),
        success:function(response){
            alert('sdas');
        }
    });
    });
    ");
    ?>
```

Site/index.php:

```
    <?php
    use yii\helpers\Html;
    use yii\widgets\ListView;
    $this->title = 'Фильмы';
    $this->registerMetaTag(['property'  =>  'og:title',  'content'  =>  $this-
>title], 'title');
```

```
    $this->registerMetaTag(['property'   =>   'og:description',   'content'   =>
'Фильмы на любой вкус'], 'description');
    ?>
        <div class="container-fluid">
        <div class="row"><h3>Популярное</h3><hr></div>
        <div class="row">
            <?php echo ListView::widget(['dataProvider' => $dataProvider,
                    'layout' => '{items}',
                    'itemView' => '_films']);
            echo  Html::a('<div  class="col-md-2  col-xs-4"  style="width:  140px;
border-radius: 8px; height: 180px; line-height: 180px;
                           background-color:     #eee">     <p     class="text-
center"><strong>Все</strong></p></div>',
                           ['films', 'sort' => 'top']);
            ?>
        </div>
        <div class="row"><h3>Новое</h3><hr></div>
        <div class="row">
            <?php echo ListView::widget([
                'dataProvider' => $dataProviderNew,
                'layout' => '{items}',
                'itemView' => '_films']);
            echo  Html::a(' <div  class="col-md-2  col-xs-4"  style="width:  140px;
border-radius: 8px; height: 180px; line-height: 180px; background-color: #eee">
<p   class="text-center"><strong>Все</strong></p></div>',['films',   'sort'   =>
'new']);
            ?>
        </div>
    </div>

    Site/login.php:

    <?php
    use yii\helpers\Html;
    use yii\bootstrap\ActiveForm;
    $this->title = 'Вход';
    $this->params['breadcrumbs'][] = $this->title;
    ?>
    <div class="site-login">
    <h1><?= Html::encode($this->title) ?></h1>
        <p>Пожалуйста заполние поля для входа:</p>
        <?php $form = ActiveForm::begin([
        'id' => 'login-form',
        'options' => ['class' => 'form-horizontal'],
        'fieldConfig' => [
            'template' => "{label}\n<div class=\"col-lg-3\">{input}</div>\n<div
class=\"col-lg-8\">{error}</div>",
            'labelOptions' => ['class' => 'col-lg-1 control-label'],
        ],
    ]); ?>
    <?= $form->field($model, 'email') ?>
    <?= $form->field($model, 'password')->passwordInput() ?>
    <div class="col-lg-offset-1">
    <?= $form->field($model, 'rememberMe', [
        'template' => "<div class=\"col-lg-3\">{input}</div>\n<div class=\"col-
lg-8\">{error}</div>",
    ])->checkbox() ?>
    </div>
    <div class="form-group">
        <div class="col-lg-offset-0 col-lg-11">
            <?=  Html::submitButton('Войти',  ['class'  =>  'btn  btn-primary',
'name' => 'login-button']) ?>
        </div>
```

```
    </div>
    <?php ActiveForm::end(); ?>
    <div class="col-lg-offset-0 col-lg-11">
        <?= yii\authclient\widgets\AuthChoice::widget([
            'baseAuthUrl' => ['site/auth']
        ]) ?>
    </div>
    <div class="col-lg-offset-0 col-lg-11">
    You can enter using these data:
    email: vova@v.com
    password: 123456
    </div>
    </div>

    User/_form.php:

    <?php
    use yii\helpers\Html;
    use yii\widgets\ActiveForm;
    ?>

    <div class="user-form">
    <?php $form = ActiveForm::begin(); ?>
    <?= $form->field($model, 'username')->textInput(['maxlength' => 16])?>
    <?= $form->field($model, 'email')->textInput(['maxlength' => 30]) ?>
    <?= $form->field($model, 'password')->passwordInput(['maxlength' => 16]) ?>
    <?=  $form->field($model,  'repeat_password')->passwordInput(['maxlength'  =>
16]) ?>
    <div class="form-group">
        <?=    Html::submitButton($model->isNewRecord    ?    'Регистрация'    :
'Изменить',  ['class'  =>  $model->isNewRecord  ?  'btn  btn-success'  :  'btn  btn-
primary']) ?>
    </div>
    <?php ActiveForm::end(); ?>
    </div>

    Site/create.php:

    <?php

    use yii\helpers\Html;
    $this->title = 'Регистрация';
    $this->params['breadcrumbs'][] = $this->title;
    ?>
    <div class="user-create">
    <h1><?= Html::encode($this->title) ?></h1>
    <?= $this->render('_form', [
        'model' => $model,
    ]) ?>
    </div>

    User/index.php:

    <?php
    use yii\helpers\Html;
    use yii\grid\GridView;
    $this->title = 'Пользователи';
    $this->params['breadcrumbs'][] = $this->title;
    ?>
    <div class="user-index">
    <h1><?= Html::encode($this->title) ?></h1>
    <?= GridView::widget([
        'dataProvider' => $dataProvider,
```

```
        'id' => 'user-group',
        'filterModel' => $searchModel,
        'columns' => [

            [
                'attribute' => 'id',
                'format' => 'integer',
                'options' => ['width' => '100px'],

            ],
            'username',
            'email:email',
            [
                'attribute' => 'date',
                'format' => 'html',
                'value' => function($model)
                    {
                        return date('d.m.Y', $model->date);
                    }
            ],
            'password',
            ['class' => 'yii\grid\ActionColumn'],
        ],
    ]); ?>
    </div>

    User/profile.php:

    <?php
    use yii\helpers\Html;
    use yii\widgets\DetailView;
    if( isset(Yii::$app->user->identity->username))
    {
    if($model->username == Yii::$app->user->identity->username)
        $view = false;
    else
        $view = true;
    }
    else
    $view = true;
    $this->title = $model->username;
    $this->params['breadcrumbs'][] = 'Профиль';
    ?>
    <div class="user-view">
    <div class="col-md-4 col-xs-6">
        <?php
            echo Html::img("/films/web/image/notFoto.png", ['width' => '350px',
'height' => '350px' ]);
        ?>
    </div>
    <h1><?= Html::encode($this->title) ?></h1>
    <p>
        <?php
            if($view == false)
                echo  Html::a('Изменить',  ['update',  'id'  =>  $model->id],
['class' => 'btn btn-primary']);
        ?>
    </p>
    </div>

    User/update.php:

    <?php
```

```
use yii\helpers\Html;
$this->title = 'Именение данных пользователя: ' . ' ' . $model->username;
$this->params['breadcrumbs'][] = 'Изменение';
?>
<div class="user-update">
<h1><?= Html::encode($this->title) ?></h1>
<?= $this->render('_form', [
    'model' => $model,
]) ?>
</div>
```

Python

```
 """Benchmark for memory reduction in deep resnet."""

 import argparse
 import os
 import sys
 import time

 parser = argparse.ArgumentParser(description='deep resnet benchmark')
 parser.add_argument('--name', type=str, default='deep',
                     help="name of benchmark run")
 parser.add_argument('--max_blocks', type=int, default=10,
                     help="maximum number of blocks to add to resnet")
 parser.add_argument('--outdir', type=str, default='.',
                     help="where to save results")
 parser.add_argument('--disable_batch_norm', type=int, default=0,
                     help="where to save results")
 args = parser.parse_args()

 module_path=os.path.dirname(os.path.abspath(__file__))
 sys.path.append(module_path+'/..')

 os.environ['TF_CUDNN_USE_AUTOTUNE']='0'  # autotune adds random memory
spikes
 os.environ['TF_CPP_MIN_LOG_LEVEL']='2'  # silence tf init messages

 import math
 import numpy as np
 import os
 import pytest
 import sys
 import tensorflow as tf
 import tensorflow.contrib.graph_editor as ge
 import time

 import memory_saving_gradients
 import resnet_model
 import mem_util

 pytestmark = pytest.mark.skipif(not tf.test.is_gpu_available(),
                                 reason="needs gpu")
 resnet_model._DISABLE_BATCH_NORM=bool(args.disable_batch_norm)

 # resnet parameters
 #HEIGHT = 32
 #WIDTH = 32
 #DEPTH = 3
 #NUM_CLASSES = 10
 #BATCH_SIZE=128
 _WEIGHT_DECAY = 2e-4
```

```
      # valid resnet sizes
      #  200 # 18, 34 , 50 , 101, 152, 200
      BATCH_SIZE=32
      RESNET_SIZE=18

      USE_TINY = True

      HEIGHT=224
      WIDTH=224

      _INITIAL_LEARNING_RATE = 0.1 * BATCH_SIZE / 128
      _MOMENTUM = 0.9

      DEPTH = 3
      NUM_CLASSES = 1001

      # add_2:0, add_7:0, add_12:0, add_17:0, add_22:0, add_27:0, add_32:0,
add_37:0, add_42:0, add_47:0, add_52:0, add_57:0,
      USE_TINY = False

      BATCH_SIZE=32
      RESNET_SIZE=18

      _WEIGHT_DECAY = 2e-4
      _INITIAL_LEARNING_RATE = 0.1 * BATCH_SIZE / 128
      _MOMENTUM = 0.9


      # debug parameters
      DUMP_GRAPHDEF = False

      def create_session():
        optimizer_options =
tf.OptimizerOptions(opt_level=tf.OptimizerOptions.L0)
        config = tf.ConfigProto(operation_timeout_in_ms=150000,
graph_options=tf.GraphOptions(optimizer_options=optimizer_options))
        #  config.graph_options.rewrite_options.constant_folding =
rewriter_config_pb2.RewriterConfig.OFF
        config.graph_options.place_pruned_graph = True
        return tf.Session(config=config)

      def create_loss():
        """Creates loss tensor for resnet model."""
        images = tf.random_uniform((BATCH_SIZE, HEIGHT, WIDTH, DEPTH))
        labels = tf.random_uniform((BATCH_SIZE, NUM_CLASSES))
        network = resnet_model.resnet_v2(resnet_size=RESNET_SIZE,
                                         num_classes=NUM_CLASSES)

        inputs = tf.reshape(images, [BATCH_SIZE, HEIGHT, WIDTH, DEPTH])
        logits = network(inputs,True)
        cross_entropy = tf.losses.softmax_cross_entropy(logits=logits,
                                                        onehot_labels=labels)
        l2_penalty = tf.add_n([tf.nn.l2_loss(v) for v in
tf.trainable_variables()])
        loss = cross_entropy + _WEIGHT_DECAY * l2_penalty
        return loss

      GLOBAL_PROFILE = True
      DUMP_TIMELINES = False
      run_metadata = True
      def sessrun(*args, **kwargs):
        global sess, run_metadata
```

```
  if not GLOBAL_PROFILE:
    return sess.run(*args, **kwargs)

  run_metadata = tf.RunMetadata()

  kwargs['options'] = tf.RunOptions(trace_level=tf.RunOptions.FULL_TRACE)
  kwargs['run_metadata'] = run_metadata
  result = sess.run(*args, **kwargs)
  first_entry = args[0]
  if isinstance(first_entry, list):
    if len(first_entry) == 0 and len(args) == 1:
      return None
    first_entry = first_entry[0]

  if DUMP_TIMELINES:
    name = first_entry.name
    name = name.replace('/', '-')

    tl = timeline.Timeline(run_metadata.step_stats)
    ctf = tl.generate_chrome_trace_format()
    with open('timelines/%s.json'%(name,), 'w') as f:
      f.write(ctf)
    with open('timelines/%s.pbtxt'%(name,), 'w') as f:
      f.write(str(run_metadata))

  return result

RESNET_SIZE=-1
def memory_test(size):
  """Evaluates gradient, returns memory in MB's and gradient eval time in
  seconds."""
  global sess, RESNET_SIZE

  RESNET_SIZE = size

  start_time0 = time.perf_counter()
  tf.reset_default_graph()

  loss = create_loss()

  start_time = time.perf_counter()
  grads = tf.group(tf.gradients(loss, tf.trainable_variables()))

  sess = create_session()
  sessrun(tf.global_variables_initializer())
  times = []
  memories = []
  for i in range(3):
    start_time = time.perf_counter()
    sessrun(grads)
    elapsed_time = time.perf_counter() - start_time
    times.append(elapsed_time)
    mem_use = mem_util.peak_memory(run_metadata)['/gpu:0']/1e6
    memories.append(mem_use)

  return np.min(memories), np.min(times)


def main():
  old_gradients = tf.gradients

  # automatic checkpoint selection
```

```
        def gradients_memory(ys, xs, grad_ys=None, **kwargs):
          return memory_saving_gradients.gradients(ys, xs, grad_ys,
                                                   checkpoints='memory',
**kwargs)
        print("Running with checkpoints")
        tf.__dict__["gradients"] = gradients_memory
        memories, times = [],[]

        outf = open(args.outdir+'/'+args.name+'.csv', 'w')

        valid_sizes = [18,34,50,101,152,200]
        for i in range(args.max_blocks):
          size = valid_sizes[i]
          memory_cost, time_cost = memory_test(size=size)
          print("%-10d %10d"%(i, memory_cost))

          memories.append(memory_cost)
          times.append(time_cost)

        def tostr(l): return [str(e) for e in l]
        outf.write(','.join(str(i) for i in range(1, max_blocks))+'\n')
        outf.write(','.join(tostr(memories))+'\n')
        outf.write(','.join(tostr(times))+'\n')


        # restore old gradients
        print("Running without checkpoints")
        tf.__dict__["gradients"] = old_gradients
        memories, times = [],[]
        for i in range(1, args.max_blocks):
          size = valid_sizes[i]
          memory_cost, time_cost = memory_test(i)
          print("%-10d %10d"%(i, memory_cost))
          memories.append(memory_cost)
          times.append(time_cost)

      #  print(memories)
      #  print(times)

        outf.write(','.join(tostr(memories))+'\n')
        outf.write(','.join(tostr(times))+'\n')
        outf.close()

      if __name__=='__main__':
        main()

      # test chain gradient, using 4 different ways of obtaining memory usage
      # 3001088 VLOG_MEMORY
      # 3003648 MaxBytesInUse
      # 3000576 metadata
      # 3003648 metadata max

      import pytest
      pytestmark = pytest.mark.skip(reason="needs memory_util")

      import os, sys
      os.environ['TF_CUDNN_USE_AUTOTUNE']='0'  # autotune adds random memory
spikes
      module_path=os.path.dirname(os.path.abspath(__file__))
      sys.path.append(module_path+'/..')

      import numpy as np
      import tensorflow as tf
```

```
      import util

      def create_session():
        config = tf.ConfigProto(log_device_placement=False,
graph_options=tf.GraphOptions(optimizer_options=tf.OptimizerOptions(opt_level=tf
.OptimizerOptions.L0)))
        return tf.InteractiveSession(config=config)


      def make_chain_tanh(length=100, name_prefix="a", node_mbs=1):
        """Creates chain of length length. First node is Variable, rest are
tanh.
        Returns nodes. Note, if length is 1, there are no non-linearities in the
        graph, hence gradients do not need to store any activations."""

        node_mbs = 1
        dtype = np.float32
        n = node_mbs * 250000
        #  a0_ = tf.ones((n,), dtype=dtype)
        #  a0 = tf.Variable(a0_, name=name_prefix+"00")
        val = tf.constant(1, dtype=dtype)
        a0 = tf.fill((n,), val)
        a = a0
        nodes = [a]
        for i in range(1, length):
          name = "%s%02d"%(name_prefix, i)
          a = tf.tanh(a, name=name)
          nodes.append(a)

        return nodes


      def main():
        import memory_util
        memory_util.vlog(1)   # vlog=2 on GPU machine will spam gpu "polling"
msgs

        tf.reset_default_graph()
        n = 3

        # TODO: fix edge case with n=2
        nodes = make_chain_tanh(n)
        a0 = nodes[0]
        a = nodes[-1]
        #grad = memory_saving_gradients.gradients_memory([a], [a0])[0]
        grad = tf.gradients(a, [a0])[0]

        sess = create_session()
        sess.run(tf.global_variables_initializer())

      #  feed_dict = {a0,
        with memory_util.capture_stderr() as stderr:
          sess.run(grad.op)

        peak_memory1 = memory_util.peak_memory(stderr.getvalue())
        # 20 mem used with following tensors picked automatically as bottlenecks
        # ['a10:0', 'a19:0', 'a28:0', 'a37:0', 'a46:0', 'a55:0', 'a64:0',
'a73:0',
        # 'a82:0', 'a91:0']

        # method 2
        mem_op = tf.contrib.memory_stats.MaxBytesInUse()
        peak_memory2 = sess.run(mem_op)
```

```
        # method 3
        run_metadata = tf.RunMetadata()
        run_options = tf.RunOptions(trace_level=tf.RunOptions.FULL_TRACE)


        sess.run(grad.op, run_metadata=run_metadata, options=run_options,)
        print(run_metadata)
        peak_memory3 = memory_util.peak_from_metadata(run_metadata)['gpu']
        print(peak_memory1, "VLOG_MEMORY")
        print(peak_memory2, "MaxBytesInUse")
        print(peak_memory3, "metadata")

        cpu,gpu=memory_util._retrieve_cpu_gpu_stats(run_metadata)
        if cpu:
          bytes_in_use_cpu = [node.memory[0].allocator_bytes_in_use for node in
cpu]
        if gpu:
          bytes_in_use_gpu = [node.memory[0].allocator_bytes_in_use for node in
gpu]

        peak_memory4 = max(bytes_in_use_gpu)
        print(peak_memory4, "metadata max")

        # fourth way would parse "allocator_bytes_in_use
         # node_stats {
         #    node_name: "Square"
         #    all_start_micros: 1509664297214870
         #    op_start_rel_micros: 4
         #    op_end_rel_micros: 115
         #    all_end_rel_micros: 136
         #    memory {
         #      allocator_name: "GPU_0_bfc"
         #      allocator_bytes_in_use: 6013952
         #    }
        expected_peak = 3 * 10**6
        util.report_memory(peak_memory1, expected_peak)

        assert abs(peak_memory3 - expected_peak) < 10000, "Difference too
large."

      if __name__=='__main__':
        assert tf.test.is_gpu_available(), "Memory tracking only works on GPU"
        main()
```

JavaScript

```
'use strict';

const shell = require('shelljs');
const util = require('../fixtures/util');
const kill = require('../utils/psKill');
const path = require('path');
const fs = require('fs');

shell.config.silent = true;

describe('razzle start', () => {
  describe('razzle basic example', () => {
    beforeAll(() => {
      shell.cd(path.join(util.rootDir, 'examples/basic'));
    });
```

```
    jasmine.DEFAULT_TIMEOUT_INTERVAL = 1000000; // eslint-disable-line no-
undef

    it('should start a dev server', () => {
      let outputTest;
      const run = new Promise(resolve => {
        const child = shell.exec('./node_modules/.bin/razzle start', () =>
{
          resolve(outputTest);
        });
        child.stdout.on('data', data => {
          if (data.includes('Compiled successfully')) {
            shell.exec('sleep 5');
            const devServerOutput = shell.exec(
              'curl -sb -o "" localhost:3001/static/js/bundle.js'
            );
            outputTest = devServerOutput.stdout.includes('React');
            kill(child.pid);
          }
        });
      });
      return run.then(test => expect(test).toBe(true));
    });

    jasmine.DEFAULT_TIMEOUT_INTERVAL = 400000; // eslint-disable-line no-
undef

    it('should build and run', () => {
      let outputTest;
      shell.exec('./node_modules/.bin/razzle build');
      const run = new Promise(resolve => {
        const child = shell.exec('node build/server.js', () => {
          resolve(outputTest);
        });
        child.stdout.on('data', data => {
          if (data.includes('> Started on port 3000')) {
            shell.exec('sleep 5');
            const output = shell.exec('curl -I localhost:3000');
            outputTest = output.stdout.includes('200');
            kill(child.pid);
          }
        });
      });
      return run.then(test => expect(test).toBe(true));
    });

    afterAll(() => {
      shell.rm('-rf', 'build');
      shell.cd(util.rootDir);
    });
  });
});

'use strict';

const shell = require('shelljs');
const util = require('../fixtures/util');

shell.config.silent = true;

const stageName = 'stage-build';
```

```
      describe('razzle build', () => {
        it('should compile files into a build directory', () => {
          util.setupStageWithFixture(stageName, 'build-default');
          const output = shell.exec('yarn build');
          // Create asset manifest
          expect(shell.test('-f', 'build/assets.json')).toBe(true);

          // Create server.js
          expect(shell.test('-f', 'build/server.js')).toBe(true);
          expect(shell.test('-f', 'build/server.js.map')).toBe(true);

          // Should copy static assets from src/public directory
          expect(shell.test('-f', 'build/public/nothing.txt')).toBe(true);

          // Should compile client bundle to js directory
          expect(shell.test('-d', 'build/public/static/js')).toBe(true);
          expect(shell.ls('build/public/static/js/bundle.*.js').code).toBe(0);

expect(shell.ls('build/public/static/js/bundle.*.js.map').code).toBe(0);

          // should compile client image assets to media directory
          expect(shell.test('-d', 'build/public/static/media')).toBe(true);
          expect(shell.ls('build/public/static/media/logo.*.png').code).toBe(0);

          // should compile client css to css directory
          expect(shell.test('-d', 'build/public/static/css')).toBe(true);
          expect(shell.ls('build/public/static/css/bundle.*.css').code).toBe(0);

          expect(output.code).toBe(0);
        });

        it('should compile files with a custom .babelrc', () => {
          util.setupStageWithFixture(stageName, 'build-with-babelrc');
          const output = shell.exec('yarn build');
          // Create asset manifest
          expect(shell.test('-f', 'build/assets.json')).toBe(true);

          // Create server.js
          expect(shell.test('-f', 'build/server.js')).toBe(true);
          expect(shell.test('-f', 'build/server.js.map')).toBe(true);

          // Should copy static assets from src/public directory
          expect(shell.test('-f', 'build/public/nothing.txt')).toBe(true);

          // Should compile client bundle to js directory
          expect(shell.test('-d', 'build/public/static/js')).toBe(true);
          expect(shell.ls('build/public/static/js/bundle.*.js').code).toBe(0);

expect(shell.ls('build/public/static/js/bundle.*.js.map').code).toBe(0);

          // should compile client image assets to media directory
          expect(shell.test('-d', 'build/public/static/media')).toBe(true);
          expect(shell.ls('build/public/static/media/logo.*.png').code).toBe(0);

          // should compile client css to css directory
          expect(shell.test('-d', 'build/public/static/css')).toBe(true);
          expect(shell.ls('build/public/static/css/bundle.*.css').code).toBe(0);

          expect(output.code).toBe(0);
        });

        it('should compile files with a custom razzle.config.js', () => {
          util.setupStageWithFixture(stageName, 'build-with-custom-config');
```

```
          const output = shell.exec('yarn build');
          // Create asset manifest
          expect(shell.test('-f', 'build/assets.json')).toBe(true);

          // We modify the default server output filename -> custom.js
          expect(shell.test('-f', 'build/custom.js')).toBe(true);
          expect(shell.test('-f', 'build/custom.js.map')).toBe(true);

          // Should compile client bundle to js directory
          expect(shell.test('-d', 'build/public/static/js')).toBe(true);
          expect(shell.ls('build/public/static/js/bundle.*.js').code).toBe(0);

expect(shell.ls('build/public/static/js/bundle.*.js.map').code).toBe(0);

          // should compile client image assets to media directory
          expect(shell.test('-d', 'build/public/static/media')).toBe(true);
          expect(shell.ls('build/public/static/media/logo.*.png').code).toBe(0);

          // should compile client css to css directory
          expect(shell.test('-d', 'build/public/static/css')).toBe(true);
          expect(shell.ls('build/public/static/css/bundle.*.css').code).toBe(0);

          expect(output.code).toBe(0);
        });

        afterEach(() => {
          util.teardownStage(stageName);
        });
      });
```

Додаток Б

ВІДГУК

на дипломний проект магістра на тему:
«Дослідження ефективності використання мов програмування, орієнтованих на web в залежності від вирішуваної задачі»

студента групи 121М-16-1 Барвіненка Дмитра Миколайовича

1. Мета дипломного проекту – проаналізувати мови програмування, які орієнтовані на веб за параметрами є важливими для реалізації задач різного типу.

2. Обрана тема актуальна в зв'язку з наявності множини мов програмування за якими можливе вирішення задач одного типу.

3. Тема дипломного проекту безпосередньо зв'язана із об'єктом діяльності магістра напрямку "Інженерія програмного забезпечення"

4. Вимоги та завдання дипломного проекту, що стосуються дослідження мов програмування орієнтованих на web, віднесені в освтньо-кваліфікацйній характеристиці магістра до класу стереотипних, вирішення яких основане на знакових та понятійних вміннях.

5. Оригінальність технічних та програмних вирішень полягає у розробці послідовності тестуючи методів, завдяки яким відбувався збір необхідних характеристик.

6. Практичне значення роботи полягає у аналізі результатів дослідження які дозволяють розробляти программи с кращими характеристиками по бістро дії та використанні машинних ресурсів за рахунок вибору найбільш єфективної мови програмування для задач відповідного типу.

7. Оформлення матеріалів дипломного проекту виконано на сучасному рівні та відповідає вимогам, що пред'явлені до кваліфікаційних робіт фахівця з інформаційних технологій.

8. Рівень самостійності виконання дипломного проекту гарний.

9. Дипломний проект в цілому заслуговує оцінки «добре», а студент Барвіненко Д.М. – присвоєння кваліфікації «інженер програміст».

Керівник дипломного
проекту бакалавра,
д.т.н., проф. к В.М. Куваєв

Додаток В

Рецензія

на дипломный проект магістра на тему:
«Дослідження ефективності використання мов програмування орієнтованих на web в залежності від вирішуваної задачі»

студента групи 121М-16-1 Барвіненка Дмитра Миколайовича

В результаті проведеної роботи було проаналізовано одні з основних характеристик мов програмування та порівняно результати з метою встановлення придатності мови програмування орієнтованої на web вирішити ту чи іншу задачу. Для досягнення основної мети були виконані наступні завдання:

• аналіз інтернет технолій сьогоднішнього дня;

• аналіз математичних алгоритмів, які дозволяють єфективно вимірювати основні характеристики мов програмування;

• аналіз методів створення веб-додатків;

• аналіз можливих завдань в веб середовищі.

Проведене дослідження дає для більш розширеного дослідження мов програмування орієнтованих на web.

Студент Д.М. Барвіненко добре розібрався в специфіці аналізу мов програмування та проведенні їх тестуванні.

З огляду на вищевикладене, можна зробити висновок, що даний проект цілком відповідає вимогам, що пред'являються до кваліфікаційних робіт рівня магістр.

Ступінь опрацювання компонентів даного проекту, дозволяє оцінити роботу на «добре» і рекомендувати присвоїти студенту Барвіенку Д.М. кваліфікацію «інженер програміст».

Рецензент,