Вплив зсуву вітру на безпеку польотів повітряних суден

Бабич Я.О.
науковий керівник: Полухін Анатолій Васильович
Навчально науковий інститут комп'ютерних інформаційних технологій
Національний авіаційний університет
Київ, Україна
yanababich@icloud.com

Бочелюк А.О.
науковий керівник: Полухін Анатолій Васильович
Навчально науковий інститут комп'ютерних інформаційних технологій
Національний авіаційний університет
Бориспіль, Україна
a.bogelyk1997@gmail.com

***Анотація* — робота присвячена розгляду проблеми безпеки польотів літаків на етапах заходу на посадку і посадки в умовах зсуву вітру. Проведено дослідження динаміки польоту літаків на етапах заходу на посадку в умовах зсуву вітру.**

Ключові слова — літак, зсув вітру, захід на посадку, безпека польотів.

I. Вступ

Зсув вітру є однією з основних причин авіаційних катастроф на малих висотах. Виходячи з цього, на Восьмій Аеронавігаційній конференції (Монреаль, 1974 р.) було прийнято рішення про підготовку матеріалу щодо метеорологічних засад та методів виявлення зон зсуву вітру і протидії цьому атмосферному явищу [4].

На основі досліджень та експериментів була розроблена Поправка 64 до Додатку 3 до Конвенції про міжнародну цивільну авіацію «Метеорологічне забезпечення міжнародної аеронавігації», яка містить положення щодо виявлення зон зсуву вітру на малих висотах з використанням спеціального бортового та наземного обладнання [2].

II. Постановка проблеми

Статистика свідчить, що близько 36% усіх катастроф на авіаційному транспорті відбувається на етапах заходу на посадку і посадки. З них на етап заходу на посадку припадає понад 14% катастроф. Якщо врахувати, що тривалість етапів заходу на посадку і посадки у середньому не перевищує 3-4% часу польоту літака, то стає зрозумілим, що рівень безпеки польоту на цих етапах у десятки разів менший, ніж рівень безпеки протягом усього польоту [5].

Остання резонансна трагічна подія з літаком Boeing 737-800 авіакомпанії FlyDubai 19 березня 2016 року, який під час руху по глісаді потрапив у зону зсуву вітру в аеропорту Ростов-на-Дону і зазнав катастрофи. Це все свідчить про те, що проблеми, пов'язані із зсувом вітру, досліджені ще недостатньо повно.

III. Основна частина

Зсув вітру, як атмосферне явище, становить собою векторну різність швидкостей вітру в двох точках повітряного простору, віднесену до відстані між ними. У залежності від просторової орієнтації точок, він поділяється на вертикальний та горизонтальний. Вертикальним зсувом вітру називається зміна горизонтальної складової швидкості та/або напрямку вітру при зміні висоти польоту. Горизонтальним – зміна горизонтальної складової швидкості та/або напрямку вітру при зміні відстані вздовж напрямку полоту в горизонтальній площині [4].

За рекомендацією ІСАО прийнята така класифікація інтенсивності зсуву вітру (вертикального – на 30 м висоти, горизонтального – на 600 м на відстані): слабкий зсув – 0-2 м/с, помірний зсув – 2-4 м/с, сильний зсув – 4-6 м/с, дуже сильний зсув – понад 6 м/с. Відповідно до встановлених ІСАО правил польотів, при інтенсивності зсуву вітру понад 5 м/с на 30 м висоти зліт та посадка повітряних суден забороняється [1].

Негативний вплив зсуву вітру на безпеку польоту літака проявляється в тому, що літак має велику інерцію, яка не дозволяє йому миттєво збільшити або зменшити земну (шляхову) швидкість при зміні швидкості вітру, в той час як істинна повітряна швидкість змінюється відповідно до її зміни. У зв'язку з цим змінюються аеродинамічні сили та моменти літака і він відхиляється від заданої траєкторії польоту.

На великих висотах польоту через наявність власної стійкості за швидкістю літак, маючи достатні запаси висоти, швидкості та часу, може за певний час самостійно, без втручання пілота, відновити вихідний режим польоту, порушений зміною істинної повітряної швидкості.

На етапі ж заходу на посадку, коли літак у посадковій конфігурації знаходиться на глісаді, значення його істинної повітряної швидкості та висоти польоту близькі до критичних, тому динаміка польоту літака на цих етапах є особливо чутливою до впливу такого небезпечного атмосферного явища, яким є зсув вітру. Адже у цьому випадку екіпаж обмежений у своїх діях запасами висоти та швидкості, а також прийомистістю двигунів при зміні їх сили тяги та дефіцитом часу для активної протидії негативному впливу зсуву вітру.

Хроніка падіння літака Boeing 737-800 авіакомпанії FlyDubai 19 березня 2016 року відображена на рис. 1. Екіпаж на висоті 220 м (за 4 км до злітно-посадкової смуги - ЗПС) прийняв рішення про «ухід на другий круг» та ініціював набір висоти з виводом двигунів на злітний режим. На висоті 900 метрів одночасно з віддачею штурвалу «від себе» стабілізатор літака був відхилений на 5 градусів на пікірування, внаслідок чого літак перейшов в енергійне зниження з реалізацією вертикального перевантаження до -1 одиниць. Подальші дії екіпажу вже були неспроможні запобігти зіткнення літака із землею. Це все свідчить про те, що проблеми, пов'язані із зсувом вітру, досліджені ще недостатньо повно.

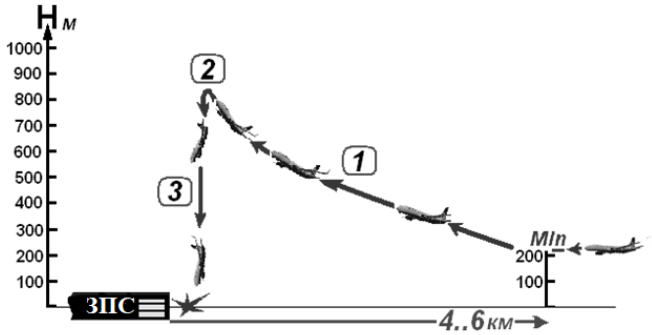

Рис. 1. Хроніка падіння Boeing 737-800.

Найбільш небезпечним для безпеки польотів є додатній зсув вітру (збільшення попутної або зменшення зустрічної швидкості вітру при зменшенні висоти), тому що викликає ухід літака під глісаду з великою ймовірністю зіткнення його з земною поверхнею до ЗПС. При від'ємному зсуві вітру (зменшення попутної або збільшення зустрічної швидкості вітру при зменшенні висоти) літак летить вище глісади і пілот має запас по висоті, швидкості та часу, щоб відвести літак на друге коло [6].

Керівні документи ІСАО передбачають, що ухід літака на друге коло повинен починатися не нижче висоти прийняття рішення з номінальним градієнтом набору висоти 2,5% та з додержанням запасу висоти над перешкодами не менше 50 м [3]. При цьому зазначені документи не передбачають урахування втрати висоти при «просадці» літака в умовах додатного зсуву вітру, яка збільшується із збільшенням вертикальної швидкості зниження при запізненні втручання пілота в управління літаком.

Звідси випливає, що своєчасне втручання пілота в управління літаком при попаданні в зону зсуву вітру в умовах відсутності запасу висоти та швидкості і дефіциту часу на виявлення цього метеорологічного явища, аналіз його особливостей і прийняття рішення щодо подальшого управління літаком відіграє значну роль у забезпеченні безпеки польоту.

Зменшення часового запізнення пілота літаком при зсуві вітру в реальних умовах забезпечується своєчасним його виявленням та правильними діями всього екіпажу, що досягається виконанням техніки пілотування літака під час тренувань на тренажері.

IV. Висновок

Зсув вітру є важкопрогнозованим атмосферним явищем, особливо небезпечним на таких етапах польоту літака, як зліт, захід на посадку і посадка, тому що значення його істинної повітряної швидкості та висоти польоту на цих етапах близькі до критичних, а екіпаж обмежений у своїх діях запасами висоти та швидкості, а також прийомистістю двигунів для зміни їх сили тяги та дефіцитом часу для активної протидії цьому небезпечному атмосферному явищу.

Окремі положення нормативних документів ІСАО, що стосуються правил виконання польотів в особливих умовах, зокрема, при уході на друге коло в умовах зсуву вітру, потребують уточнення та доповнення окремо для кожного класу літаків з урахуванням їх геометричних, масово-інерційних, аеродинамічних, пілотажних та інших характеристик.

Список використаних джерел

[1] Журавлев А И. Влияние сдвига ветра на взлет и посадку самолетов / А.И. Журавлев, О. К. Трунов // Методичсекие рекомендации для летного и деспетчерского состава гражданской авиации. М.: МГА, 1979. – 15 с.

[2] Метеорологичское обеспечение международной аєронавигации. При ложение 3 к Конвенции о международной гражданской авиации // Издание восемнадцатое (с поправками). Часть 1. Основніе SARPS. Часть 2. Добавления и дополнения. ИКАО: 2013. – 222с.

[3] Производство полетов воздушных судов // Издание пятое (с поправками). Doc 8168 OPS/611/ Том 1. Правила производства полетов. ИКАО: 2006. – 386 с.

[4] Руководство по сдвигу ветра на малых высотах // Издание первое (с поправками). Doc 9817 AN/449/ ИКАО: 2005. – 264 с.

[5] Statistics/ Causes of Fatal Accidents. Fatalities by Phase of Flight. – URL: http://www.planecrashinfp.com/cause.html.

[6] Бабич Я. О., Бочелюк А. О., Полухын А. В. Особливості заходу на посадку літака в режимах автоматичного та шутрвального управління в умовах вертикального зсуву вітру // Проблеми інформатизації та управління.- 2017. - №4(60) – с.5-11.

Складнощі впровадження концепції віртуалізації мережевих функцій

Бабич Я.О.
науковий керівник: Савченко Аліна Станіславівна
Кафедра комп'ютерних інформаційних технологій
Навчально-науковий інститут комп'ютерних інформаційних технологій,
Національний авіаційний університет,
Київ, Україна
yanababich@icloud.com

Бочелюк А.О.
науковий керівник: Савченко Аліна Станіславівна
Кафедра комп'ютерних інформаційних технологій
Навчально-науковий інститут комп'ютерних інформаційних технологій,
Національний авіаційний університет,
Бориспіль, Україна
a.bogelyk1997@gmail.com

***Анотація* — в роботі проаналізовано концепцію віртуалізації мережевих функцій та класи задач, які можна вирішити з її допомогою. В результаті аналізу виявлено області доцільного застосування та недоліки впровадження.**

Ключові слова — мережеві технології; NFV; хмарні обчислення.

I. ВСТУП

Концепція віртуалізації мережевих функцій NFV (Network Functions Virtualization) - це новий підхід для надання мережевих послуг, який дозволяє замінити звичне обладнання (маршрутизаторів, комутаторів та ін.) віртуальними аналогами. Це надає можливість підвищити ефективність роботи мережі і загальну гнучкість мережевої архітектури за рахунок переходу від апаратної реалізації до програмного впровадження мережевих послуг.

Цей підхід має широке застосування у мережах операторів зв'язку, його реалізація має колосальний вплив на безпеку архітектури і дозволяє використовувати нові сучасні методи в сфері інформаційних послуг.

II. АКТУАЛЬНІСТЬ ПРОБЛЕМИ

Проблема зростання мобільного трафіку даних та збільшення кількості сервісів набуває глобального рівня. Крім того, збільшуються об'єми та частота надходження службового трафіку, що передається мережею, а тому виникає необхідність у його ефективному управлінні з метою забезпечення потрібної якості обслуговування користувачів та оптимального використання ресурсів мережі оператора зв'язку.

На ринку телекомунікаційних послуг України оператори роблять перші практичні кроки до віртуалізації інфраструктури, активно тестують рішення на базі концепції NFV з подальшими планами щодо побудови «телеком хмари» (Telco Cloud). При цьому, чим ширше вони використовують віртуалізацію мережевих послуг, тим більше перешкод зустрічають на своєму шляху. В першу чергу виникають проблеми неготовності технологій до комерційної експлуатації, відсутність стандартів, нерозуміння економічної ефективності.

Тому актуальною задачею є аналіз недоліків технології NFV при впровадженні та доцільних областей використання.

III. ОСНОВНА ЧАСТИНА

Мережі операторів зв'язку складаються з великого різноманіття фізичного обладнання. Для запуску нової служби або послуги часто необхідна установка нового устаткування, що тягне за собою необхідність пошуку окремого джерела живлення, висококваліфікованих фахівців і т. д. Більш того, будь-яке фізичне обладнання може виходити з ладу і перестає підтримуватися виробником, що провокує новий цикл інтеграції обладнання на заміну. Однак, в даний час темпи розвитку технологій вкрай високі і життєвий цикл обладнання скорочується. Віртуалізація мережевих функцій націлена на трансформацію принципу побудови мереж за рахунок еволюції стандартів в технологіях віртуалізації.

Віртуалізація мережевих функцій — це концепція мережевої архітектури, що пропонує використовувати технології віртуалізації для відтворення цілих класів функцій мережевих вузлів у вигляді складових елементів, які можуть бути з'єднані разом або пов'язані в ланцюжок для створення телекомунікаційних послуг (сервісів).

NFV відрізняється від традиційних способів віртуалізації, що використовуються в інформаційних технологіях рівня підприємства. Функція мережі, що віртуалізується, може включати одну або кілька віртуальних машин, що використовують різноманітне програмне забезпечення та процеси, на верхівці галузевих стандартів сервери, комутатори і сховища великого обсягу, або навіть інфраструктури хмарних обчислень, замість окремих апаратних рішень для кожної конкретної мережевої функції.

NFV відкриває нові можливості і шляхи реалізації. Схематично цей підхід працює наступним чином [1].

• Спочатку пристрої, що мають різний функціонал, реалізуються у вигляді мережевих функцій, потім віртуалізуються, розбиттям на ряд віртуальних мережевих функцій. Потім вони виконуються на стандартизованих серверах, які для ефективного розгортання NFV потрібно віртуалізувати у фізичну мережеву інфраструктуру.

• Створюється зв'язок між віртуальними мережевими функціями. Для цього, за допомогою відкритих платформ, таких як OpenStack, формуються сервісні ланцюжки, які можна змінювати, адмініструвати та моніторити.

• Після цього у операторів зв'язку з'являється можливість інтегрувати цю оптимізовану віртуальну інфраструктуру в свою існуючу або оновлену систему оркестрації, тобто виділення віртуальних ресурсів тих чи інших послуг за запитом. На цьому етапі можлива інтеграція з системою OSS / BSS, яка забезпечує облік, білінг, динамічне надання і швидке створення нових послуг.

Переваги, які забезпечує NFV:

• Гнучкість. Використання віртуалізації мережевих функцій забезпечує швидкість і легкість розгортки мережі і її експлуатацію. Крім того, код застосовуваних мережевих функцій не вимагає централізованого розміщення і може перебувати, де завгодно. Це дозволяє переміщувати його територіально і розміщувати якомога ближче до місця застосування.

• Вартість. Гнучкість розгортання NFV веде до зниження витрат на управління послугами, які надаються та скорочує витрати, пов'язані з управлінням усією мережею.

• Масштабованість. Послуги, організовані на базі програмного забезпечення, а не на фізичному рівні, дозволяють протягом дня збільшувати або зменшувати обсяг використовуваних ресурсів одного і того ж апаратного забезпечення в залежності від навантаження, а також підвищувати навантаження на устаткування при виникненні надзвичайних ситуацій.

• Безпека. Оператори хочуть захистити дані користувачів, а віртуалізація мережевих функцій дозволяє підвищити рівень безпеки і збереження даних за рахунок поділу та ізоляції сегментів мережі.

• Швидке розгортання послуг в іншій мережі. Віртуалізація дозволяє операторам отримати можливість розгортати власні сервіси в будь-якій точці світу, йдучи назустріч потребам своїх абонентів.

• Керованість мережі. Використання віртуальних мережевих функцій для моніторингу, балансування навантаження та управління мережевими ресурсами дозволяє динамічно реагувати на мінливі потреби клієнтів.

Але швидкий розвиток NFV гальмується за рахунок наступних недоліків [2].

Складнощі промислової реалізації NFV полягають у забезпеченні прозорості віртуальних сервісів для послуг споживачів, моніторингу, тестування, діагностування та відновлення, розділення трафіку управління від даних користувачів і надання необхідної якості обслуговування та виконання процесів.

Крім цього, відсутні єдині стандарти в технології NFV. Кожна компанія, що спеціалізується на управлінні NFV-мережами (MANO - NFV management and orchestration) створює сервіси, виходячи зі свого бачення. Різні підходи конкурують на ринку, а відсутність повністю готових до впровадження універсальних рішень гальмує розвиток NFV. Звідси ж випливає несумісність технологічних компонентів різних вендорів, позбавляючи систему гнучкості. Також величезна інстальована база змушує виробників обладнання підтримувати успадковані рішення.

Наступним стримуючим фактором для розвитку NFV є консерватизм операторів. Щоб отримувати вигоду від віртуалізації мережевих функцій, операторам потрібно бути готовими до трансформації. Коли компанія, запровадивши NFV, продовжує працювати за колишньою моделлю, часто NFV виявляється дорожче, ніж класична модель реалізації функції мережі зв'язку. У підсумку, формується думка, що запуск технології NFV - це дорогий процес. Також важливо розуміти, що для експлуатаційного обслуговування віртуалізованих мереж, співробітники повинні мати відповідну кваліфікацію.

IV. ВИСНОВКИ

Концепція NFV відносно молода, але за останні роки вона доволі активно розвинулась на світовому технологічному ринку, завдяки гнучкості, економічній ефективності, масштабованості і безпеці. Проте будь-яка нова технологія повинна пройти цикл зрілості, причому дозріти повинна не тільки сама концепція, але і ринок. Незважаючи на це, якщо технологія буде впроваджуватися то NFV і технології безпеки, а також інструменти безпеки для віртуалізованних середовищ будуть розвиватися.

Отже, аналіз технології віртуалізації ресурсів мережі показав доцільність її використання для вирішення актуальних телекомунікаційних проблем. Однак, такі фактори, як підвищена складність роботи і проблеми управління в порівнянні з перевагами даного підходу, повинні бути ретельно проаналізовані.

Хоча NFV обіцяє істотну економію коштів, гнучкість і простоту розгортання, потенційні проблеми в реалізації віртуалізованих мережевих елементів, які можуть підтримувати вимоги до продуктивності реального світу, і досі залишаються відкритим питанням, й в даний час NFV все ще перебуває на початкових етапах реалізації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] NFV виртуализация сетевых функций [Електронний ресурс]. Режим доступу:http://sci-article.ru/stat.php?i=1455156066.

[2] Виртуализация сетевых функций NFV для сетей операторов связи. Преимущества и практика внедрения NFV. Драйверы и препятствия развития. [Електронний ресурс]. Режим доступу: http://1234g.ru/novosti/nfv-v-setyakh-operatorov-svyazi.

Стійкість розв’язків задачі оцінювання архітектури програмних систем

Бочелюк А.О.
науковий керівник: Харченко Олександр Григорович
Навчально науковий інститут комп’ютерних інформаційних технологій
Національний авіаційний університет
Бориспіль, Україна
a.bogelyk1997@gmail.com

Бабич Я.О.
науковий керівник: Харченко Олександр Григорович
Навчально науковий інститут комп’ютерних інформаційних технологій
Національний авіаційний університет
Київ, Україна
yanababich@icloud.com

***Анотація* — в роботі проведено порівняльне дослідження класичного та модифікованого методів аналізу ієрархій в задачі оцінювання якості архітектури програмної системи. Для цього для різних значень неузгодженостей матриці парних порівнянь порівнювались рішення задачі оцінювання, отримані класичним і модифікованим методом.**

Ключові слова — архітектура програмної системи, метод аналізу ієрархій, оптимізація, оцінювання.

I. Вступ

У зв’язку з підвищенням складності програмних систем (ПС) зростають вимоги до їх архітектури. Складність вирішуваних системою задач робить неможливою розробку архітектури «з нуля», а використання існуючих рішень є неприйнятним у зв’язку із постійним підвищенням вимог до якості ПС та швидкими темпами удосконалення апаратно програмних платформ.

В [4] для рішення задачі оптимального вибору архітектури розподіленої програмної системи використано метод аналізу ієрархій (MAI). Але відомо, що застосування стандартного MAI, при значній кількості альтернатив (n≥9) приводить до суттєвих неузгодженостей між елементами матриці парних порівнянь, що породжує похибки при визначенні вагових множників альтернатив. Тому, в [2] запропонована модифікація MAI, в якій вагові множники визначаються з умови мінімізації неузгодженості матриці парних порівнянь, що приводить вихідну задачу до задачі математичного програмування. В [5] розглянуті питання застосування модифікованого MAI до задачі вибору оптимальної архітектури програмних систем.

В даній роботі проведено дослідження того, наскільки така модифікація може покращити рішення вихідної задачі.

II. Стійкість модифікованого MAI до похибок експертних даних

Розглядаються два рівні критеріїв якості:

- $K_i^1, i = \overline{1, m1}$, – критерії якості у використанні ПС у відповідності із стандартом ISO/IEC 25010;
- $K_i^2, i = \overline{1, m2}$, – критерії якості архітектури;
- $A_i, i = \overline{1, n}$, – альтернативні архітектурні рішення.

Необхідно вибрати таке архітектурне рішення яке б оптимізувало сукупність критеріїв $\{K_i^1\}$, $\{K_i^2\}$. Для розв’язання таких задач найчастіше використовується метод аналізу ієрархій Сааті [3].

При використанні MAI для рішення таких задач вагові множники альтернатив (критеріїв)$\{w_i\}$ на кожному рівні знаходяться з використанням матриць парних порівнянь $\Gamma(\gamma_{ij})$, які заповнюють експерти (тут γ_{ij} визначає перевагу *i*-тої альтернативи над *j*-ю).

Коефіцієнти матриць повинні бути узгодженими, тобто $\gamma_{ij} = w_i/w_j \ \ \forall\gamma_{ij} \in \Gamma$. Але при значній кількості альтернатив в силу дії на експертів різних факторів матриця $\Gamma\{\gamma_{ij}\}$ є неузгодженою і в неї ранг відмінний від одиниці, тобто матриця буде мати декілька власних значень. Для оцінювання узгодженості при незначних порушеннях пропонується використовувати індекс узгодженості [3]:

$$I_u = \frac{\lambda_{max} - n}{n - 1} \qquad (1)$$

де $\lambda_{max} = \sum_{i=1}^{n}(x_i \times \sum_{k=1}^{n}\gamma_{ik})$ - максимальне власне значення, n-порядок матриці Γ, x_i - компоненти власного вектору. Потім обчислюються значення I_u для значень γ_{ij} з похибками, які задаються як випадкові величини, і обчислюється відношення узгодженості:

$$I_o = \frac{I_u}{M(I_u)} \qquad (2)$$

тут - $M(I_u)$ випадкова узгодженість, яка повинна обчислюватись зі збереженням оберненої симетричності випадкової матриці Γ.

Індекс узгодженості містить інформацію про порушення чисельної (кардинальної) та транзитивної узгодженості. Якщо значення I_o не перевищує 10 відсотків то рівень узгодженості вважається задовільним [3].

Крім перерахованих вище будемо використовувати наступні показники ,введені в [2]:

- коефіцієнт узгодженості

$$K(w_i^*) = \frac{1}{n-1}\sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{1}{\gamma_{ij}} \left| \frac{w_i^*}{w_j^*} - \gamma_{ij} \right|, \qquad (3)$$

- а також міри узгодженості

$$M_1 = \sum_{i=1}^{n} K(\varpi_i^*), \qquad M_2 = max_i K(\omega_i^*) \qquad (4)$$

В роботі [2] для рішення задач прийняття ієрархічних рішень по неузгодженим матрицям парних порівнянь запропоновано вагові множники альтернатив шукати з умови мінімізації неузгодженості матриці Γ(γ).

В якості міри неузгодженості, яка є більш зручною з точки зору практичного застосування, можна взяти наступний вираз:

$$\left| \frac{w_i}{w_j} - \gamma_{ij} \right| \leq \delta_{max} \cdot \gamma_{ij} \qquad (5)$$

де δ_{max} – граничне значення міри неузгодженості.

Вагові множники $w_i, i = \overline{1,n}$ знаходяться з рішення задачі мінімізації міри (5), яка підстановкою $w_i - \gamma_{ij} w_j = y_{ij}^+ - y_{ij}^-;\ \ y_{ij}^+, y_{ij}^- \geq 0;\ \ i,j = \overline{1,n}$ зводиться до наступної задачі лінійного програмування:

$$\min_{\{w_i\}_{i=\overline{1,n}}} \sum_{i=1}^{n}\sum_{j=1}^{n} \left(y_{ij}^+ + y_{ij}^-\right) \qquad (6)$$

$$-\delta_{max} \cdot \gamma_{ij} \cdot w_{ij} \leq w_i - \gamma_{ij} \cdot w_{ij} \leq \delta_{max} \cdot \gamma_{ij} \cdot w_{ij}; \qquad (7)$$

де $1 \leq a \leq w_i, \quad i = \overline{1,n}$; a – задане число.

Були проведені дослідження, в яких для заданих відносних значень неузгодженості δ_{max} знаходились вагові множники $w_i^*, i = \overline{1,n}$ стандартним і модифікованим MAI. Після цього обчислювались коефіцієнти, та міри узгодженості (1),(2),(3),(4) для результатів, отриманих обома методами. Результати порівняння приведені в наступному розділі.

III. Експериментальне дослідження стійкості розв'язку.

Дослідження проводилось для десяти і п'ятнадцяти архітектурних рішень, які оцінювались відносно наступних критеріїв якості: модифікованість, масштабованість, експлуатаційні якості, вартість, затрати на розробку, переносимість, легкість установки.

По кожному з критеріїв формувалась матриця $\Gamma^s(\gamma_{ij}^s)$, $i,j = \overline{1,n}, s = \overline{1,7}$, де γ_{ij}^s показує, наскільки *i*-та альтернатива переважає *j*-ту по реалізації *s*-го критерію. При чому, матриці задавались ідеально узгодженими. Потім моделювались помилки експертів шляхом генерування випадкових величин K_{ij} в інтервалі $K_{ij} \in [0; \delta_{max}]$ з певним кроком $\Delta\delta$, і елементи матриці $\Gamma^s(\gamma_{ij}^s)$ визначались за формулою:

$$\gamma_{ij}^{s*} = \gamma_{ij}^s + K_{ij} \cdot \gamma_{ij}^s \qquad (6)$$

Для отриманих матриць $\Gamma^*(\gamma_{ij}^{s*})$ визначались набори вагових множників $\{w_i^s\}, i = \overline{1,n}, s = \overline{1,7}$ стандартним MAI, і як рішення задачі (3), (4). Після цього обчислювались міри узгодженості *M*1 і*M*2, які осереднювались по множині критеріїв якості.

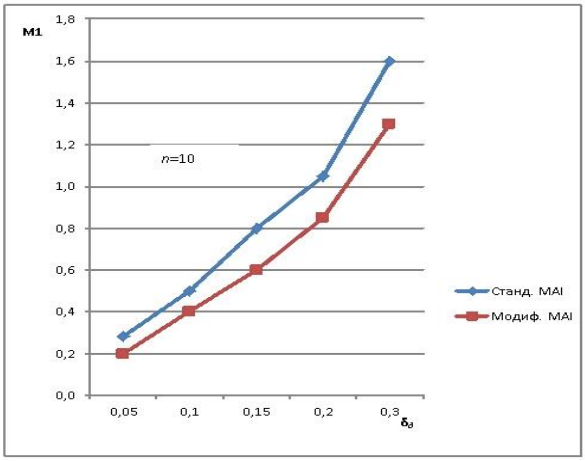

Рис.2. Залежність критерію*M*1 від величини похибки

На рисунку 2 зображена залежність критерію *M*1 від величини інтервалу, з якого вибирався K_{ij} для обох методів. Як видно з графіка, вже при похибках в матриці $\Gamma^*(\gamma_{ij}^{s*})$ в межах $\delta_{max} = 0.2$ модифікований MAI дає значно кращі результати за критерієм *M*1, ніж стандартний.

IV. Висновки

Результати проведених досліджень показали, що неузгодженість матриць парних порівнянь при використанні стандартного MAI суттєво впливає на коректність результатів. З метою дослідження стійкості методу аналізу ієрархій до похибок експертних даних проводилось тестове обчислення вагових множників з використанням стандартного методу аналізу ієрархій та модифікованого методу аналізу ієрархій. Дослідження модифікованого MAI показали, що навіть при великій ступені неузгодженості та великій кількості критеріїв отримані точні результати. Стандартний MAI при кількості альтернатив $n > 10$ та неузгодженості матриці парних порівнянь $\delta_{max} > 0.2$ дає суттєві відхилення значень вагових множників за двома критеріями *M*1 та *M*2.

Список використаних джерел

[1] HarchenkoAlexandr, BodnarchukIhor, HalayIryna, YatcyshynVasyl. Software Architecture Design on the Base of Method of Hierarchic Optimization // Proceeding of VIIIth International Conference on Perspective Technologies and Methods in MEMS Design. pp. 39–40, Polyana, 2012.

[2] Павлов А.А. Математические модели оптимизации для нахождения весов объектов в методе парных сравнений. Павлов А.А, Лищук Е.И., Кут В.И. // Системні дослідження та інформаційні технології. – К.: ІПСА, – 2007. №2, с. 13 – 21.

[3] Decision Making with the Analytic Network Process. Saaty T. Vargas L. – N.Y.: Springer, 2006. – 278 h.

[4] Al-Naemm. A quality driven systematic approach for architecting distributed software application, In Proceedings of the 27th International Conference on Software Engineering St.Louis,2005.

[5] Харченко О.Г. Проектувангня архітектури WEB-застосування на основі моделі якості. / О.Г. Харченко, І.О. Галай, І.О. Боднарчук, В.В. Яцишин // Інженерія програмного забезпеченняю № 4, 2010, с. 26 – 34.

AR-based heads-up для голографічного проектування

Валюх В.О
науковий керівник: Куклінський М.В.
Кафедра комп'ютерних інформаційних технологій
Навчально-науковий інститут комп'ютерних інформаційних технологій,
Національний авіаційний університет,
Київ, Україна
vladyslav.valiukh@ukr.net

Романенко Д.А
науковий керівник: Куклінський М.В.
Кафедра комп'ютерних інформаційних технологій
Навчально-науковий інститут комп'ютерних інформаційних технологій,
Національний авіаційний університет,
Київ, Україна
dima.romanenko.200036@gmail.com

Анотація **– робота присвячена розгляду проблеми використання голографічного проектування для відображення віртуального світу за допомогою інтегрування реальної інформації.**

Ключові слова – доповнена реальність, доповнена віртуальність, голографічне проектування, штучний інтелект, GPS-система.

I. ВСТУП

Доповнена реальність – це проекти, спрямовані на доповнення реальності будь-якими віртуальними елементами. Доповнена реальність – складова частина змішаної реальності, в яку також входить «доповнена віртуальність» (коли реальні об'єкти інтегруються у віртуальне середовище) [1]. На спектрі між віртуальною реальністю, яка створює занурення, комп'ютерне середовище та реальний світ являються розширеною реальністю, яка наближається до реального світу. Розширені реалії додає графіка, звуки, зворотний зв'язок та запах природного світу, якщо його можна передати.

Як відеоігри, так і мобільні телефони стимулюють розвиток розширеної реальності. Кожний користувач від туриста до солдата, або до когось, хто шукає найближчу зупинку метро, відтепер може скористатися можливістю розміщення комп'ютерної графіки у своїй області зору.

II. ПОСТАНОВКА ПРОБЛЕМИ

На даний момент існує проблема зв'язку штучного інтелекту та водіїв автомобілів. Причина конфлікту проста – коли автомобіль сам виконує всі завдання водія, він, по суті, стає відповідальним за його життя та тих, хто поділяє з ним дорогу.

Автоматично пілотований автомобіль перш ніж фактично зробити переїзд з одного пункту в інший, повинен показати свій намір зробити це. І саме тут додаткова реальність (*Augmented Reality*, AR) отримує своє визнання серед споживачів.

III. ОСНОВНА ЧАСТИНА

WayRay, швейцарський провайдер програмного забезпечення для додаткової реальності, який намагається вирішити проблему довіри до пілотованих автомашин. Вони представляють AR-based head-up дисплей, який голографічно проектує наміри автомобіля повернути вліво або вправо, прискорити, гальмувати або зробити інший маневр прямо на його вітровому склі.

Цей розширений HUD також показує інформацію про навколишнє середовище, маршрут, точки інтересу, доступні функції, потенційні перешкоди та продуктивність автомобіля, щоб гарантувати водієві, що все гаразд.

WayRay робить розширену реальність критичною частиною інтерфейсу Human Machine, який порушує бар'єр між людьми та повноцінним металевим кокпітом автономної машини.

Також AR досліджується і для пасажирів. Компанія Toyota випустила робочі концепції своєї системи AR, яка дозволить пасажирам збільшувати об'єкти поза автомобілем, вибирати та ідентифікувати об'єкти, а також переглядати відстань об'єкта від автомобіля за допомогою сенсорного екрана [2], [3].

IV. ВИСНОВКИ

В роботі описано один з найперспективніших на даний час спосіб використання доповненої реальності та віртуальності, а саме голографічного проектування з використанням AR-based head-up дисплею.

Використання даної системи, наприклад, з GPS-системою автомобіля зможе додатково вирішити питання виділення смуги, на ділянках зі слабою видимістю [4].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] https://uk.wikipedia.org/wiki/Доповнена реальність
[2] https://www.intellias.com/ces-2018-tomorrow-trends-to-solve-today-problems/
[3] https://wayray.com/
[4] https://computer.howstuffworks.com/augmented-reality.htm

Поетапний підхід створення Web-сайтів

Верещака Б.П.
науковий керівник: Холявкіна Т.В.
Кафедра комп'ютерних інформаційних технологій,
Навчально-науковий інститут комп'ютерних інформаційних технологій,
Національний авіаційний університет,
Київ, Україна
bohdanprod@ukr.net

***Анотація* — робота присвячена проблемі розробки концептів, спеціалізованих на продаж web-сайтів.**

Ключові слова — web-сайт, web-сторінка, HTML, CSS, JavaScript.

I. ВСТУП

На сьогодні проблема продажу свого товару не є складною задачею для більшості компаній. Потенційним покупцям достатньо відкрити Інтернет, не виходячи з дому, і онлайн замовити все, що їм подобається. Для замовлення існує безліч web-сайтів, тобто сукупність електронних документів (файлів) приватної особи або організації в комп'ютерній мережі, об'єднаних під одною адресою (доменним ім'ям або IP-адресою).

Web-сторінка – це набір текстових файлів, розмічених мовою HTML. Ці файли, будучи завантаженими відвідувачем на його комп'ютер, розуміються і обробляються браузером і виводяться на засіб відображення користувача. Мова HTML дозволяє форматувати текст, розрізняти в ньому функціональні елементи, створювати гіпертекстові посилання і вставляти в сторінку зображення, звукозаписи і інші мультимедійні елементи. Відображення сторінки можна змінити додаванням в неї таблиці стилів CSS [1].

II. ПОСТАНОВКА ПРОБЛЕМИ

Сьогодні існує безліч сайтів різної тематики. Вони відрізняються між собою оформленням, специфікою пропонованого товару, способом замовлення та оплати продукту.

Хоча більшість подібних сайтів орієнтуються на спосіб оплати товару за допомогою Інтернет-гаманців за фактом замовлення, є сайти, які орієнтуються на оплату товару кур'єру виключно готівкою при отриманні товару. Метою роботи є опис етапів створення web-сайтів, головною перевагою яких буде легкість їх наповнення новим матеріалом, забезпечення зручності, безпеки і інтуїтивної зрозумілість інтерфейсу користувачеві.

III. ОСНОВНА ЧАСТИНА

Створення сайту можна поділити на такі етапи:

1. Попередній етап розробки сайту (на цьому етапі розв'язуються питання загального характеру. Обговорюється загальна концепція сайту, формулюються та фіксуються цілі створення сайту).
2. Етап проектування сайту (Визначення структури сайту: меню, посилання, розміщення модулів, побудова списку компонентів, що підключаються, тощо).
3. Етап розробки й тестування сайту.
4. Розміщення сайту.
5. Розвиток ресурсу.

Створення веб-сайту починається зі створення інформаційної моделі сайту. Будь-яку веб-сторінку можна оцінити за двома параметрами: зміст та зовнішній вигляд. Необхідно детально проаналізувати, скільки і якої інформації потрібно подати на веб-сторінці. Створюючи проект сайту, потрібно добре продумати його загальну структуру, зміст інформації та посилання [2].

В основі створення web-сайту є принцип створення звичайної сторінки. Спочатку використовується мова розмітки HTML – стандартна мова розмітки веб-сторінок в Інтернеті. Документ HTML обробляється браузером та відтворюється на у звичному для людини вигляді [3].

Потім підключаються таблиці стилів CSS які використовуються для опису зовнішнього вигляду сторінок, написаних мовами розмітки даних. Сторінки зменшуються в об'ємі та стають більш структурованими, також CSS дає можливість прискорити завантаження сторінок.

Щоб сайт був інтерактивний, необхідно використовувати JavaScript, для написання сценаріїв веб-сторінок і надання їм інтерактивності, а також змоги програмування на стороні сервера. В цілому JS – динамічна, об'єктно-орієнтована мова програмування [4]. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта взаємодіяти з користувачем, змінювати структуру та зовнішній вигляд веб-сторінки.

IV. ВИСНОВКИ

Створення сучасного сайту вимагає процесного та поетапного підходу. В статті, описані підходи, що до основ створення web-сайту, етапів розробки та елементарних розумінь про мову розмітки HTML, таблиці стилів CSS та JavaScript. Дотримуючись описаним рекомендаціям організації зможуть створити орієнтований під їх діяльність web-сайт, який буде задовольняти потребам більшості їх потенціальних покупців.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] https://2ip.ua/ua/blog/website
[2] http://urok-informatiku.ru/structura-web-saitiv/
[3] https://uk.wikipedia.org/wiki/HTML
[4] https://uk.wikipedia.org/wiki/JavaScript

Практична цінність централізованого сховища Google Drive

Горовая Н.М.
науковий керівник: Куклінський Максим Володимирович
Кафедра комп'ютерних інформаційних технологій,
Науково-навчальний інститут комп'ютерних інформаційних технологій,
Національний авіаційний університет,
Київ, Україна
natalya21753@gmail.com

***Анотація* — робота присвячена розгляду проблеми зберігання інформації та швидкого доступу до неї з будь-якого комп'ютера, а саме ми проаналізуємо вирішення цієї проблеми за допомогою централізоаного сховища Google Drive.**

Ключові слова — централізоване сховище, Google Drive (Диск), доступ до файлів, спільне середовище.

I. ВСТУП

Різні сервіси зберігання інформації вже досить давно користуються величезною популярністю у користувачів з усього світу. І це зовсім не дивно, адже такі сервіси надзвичайно зручні. Вони дозволяють не використовувати фізичну пам'ять, яка є на комп'ютері, телефоні, планшеті, флешці або будь-якому іншому носії інформації, а завантажувати на такі ось сховища. Їх, до речі, ще називають хмарними.

Така назва обумовлена тим, що користувач, маючи логін і пароль, може користуватися даними з сховищ на всіх своїх пристроях. Ця можливість сильно виручає всіх, у кого постійно не вистачає пам'яті, а розширити її неможливо.[1]

II. ПОСТАНОВКА ПРОБЛЕМИ

Сучасні інформаційні технології неухильно розвиваються щодня. Кожну годину у світі відбувається вдосконалення вже створених, та створення нових додатків та самого комп'ютерного оснащення. Таким чином виникає потреба у більшому просторі для зберігання інформації, а також у швидкому доступі до збереженої інформації.

Особливо ця проблема актуальна для підприємств, навчальних закладів, офісів тощо, оскільки в таких установах необхідно зберігати набагато більшу кількість файлів та надавати доступ до них одразу декільком користувачам.

III. ОСНОВНА ЧАСТИНА

Описану вище проблему було вирішено 24 квітня 2012 року компанією Google Incorporation, саме в цей день було презентовано нову розробку під назвою Google Disk. Google Drive – це централізоване сховище для безпечного та швидкого доступу до файлів, яке було розроблено з метою завантаження та зберігання інформації, для швидкого доступу до цих файлів з будь-якої точки світу та з будь-якого комп'ютера чи смартфону за допомогою облікового запису Google або посилання на ці файли.[2]

Це файлове сховище було написано на мові програмування Python, воно підтримується на наступних операційних системах: Windows, Android,OS X, iOS, Chrome OS, окрім того Google Drive підтримує 68 мовних інтерфейсів. Зі спільним Google Drive можна не турбуватися про захист результатів командної роботи, тому що вони зберігаються в безпечному спільному середовищі, яким легко керувати. Файли, додані на спільний Google Drive, належать усім членам команди, тому кожен користувач завжди в курсі останніх змін. Значною перевагою використання Google Drive є необмежений простір для зберігання файлів.

Також можна легко розширити можливості Google Drive за допомогою додатків інших розробників, наприклад таких як DocuSign для електронних підписів, CloudLock для додаткового захисту, LucidCharts для створення та редагування діаграм. Файли на Диску залишатимуться приватними, доки користувач не вирішить ними поділитися. Також можна надавати іншим користувачам дозвіл завантажувати, редагувати, коментувати чи переглядати файли на Диску. При цьому можна встановлювати кінцевий термін доступу до спільних файлів

IV. ВИСНОВКИ

В тезі розкрито загальну та практичну цінність сховища Google Drive для вирішення проблеми зберігання та швидкого доступу до файлів декількох користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] https://ua.tehpo.xyz/servisy-xraneniya-informacii/
[2] https://gsuite.google.com.ua/intl/uk/products/drive

Застосування плагінів Google Chrome

Гриб М.О.
науковий керівник: Холявкіна Т.В.
Кафедра комп'ютерних інформаційних технологій
Навчально-науковий інститут комп'ютерних інформаційних технологій,
Національний авіаційний університет,
Київ, Україна
marinka.grib@gmail.com

***Анотація* — робота присвячена розгляду застосування плагінів для розширення функціональності браузера. В роботі запропонований розгляд основних можливостей браузера Google Chrome та їх доповнення за допомогою плагінів.**

Ключові слова — веб-браузер, графічний інтерфейс, розширення, доповнення, плагін, веб-переглядач, панель інструментів, програмне забезпечення.

I. ВСТУП

Одним з перших способів організації доступу до інформаційних ресурсів мережі Інтернет, стали веб-браузери. Браузер, також веб-браузер – це програмне забезпечення для комп'ютера, або іншого електронного пристрою, як правило, під'єднаного до Інтернету. Плагін – призначений для розширення чи використання загальних можливостей браузера. Вони використовуються для забезпечення показу форматів даних, які не мають вбудованої підтримки браузером, для підлаштування можливостей під вимоги користувача, та навіть для налогодження деяких скриптових програм [1].

II. ПОСТАНОВКА ПРОБЛЕМИ

З розвитком інформаційних технологій та суттєвого збільшення інформації в ресурсах, особливої уваги потребують веб-переглядачі та їхні функціональні можливості. Тому, постає питання актуальності застосування плагінів в браузерах. Такий спосіб представлення робочого місця користувача Інтернету займає все вищі позиції, адже він дозволяє обрати які дані найбільш важливі для користувача і розмістити їх безпосередньо поверх відкритого вікна браузера.

III. ОСНОВНА ЧАСТИНА

Браузер надає графічний інтерфейс для інтерактивного пошуку, виявлення, перегляду та обробки даних у мережі. За допомогою посилань веб-переглядач дозволяє користувачеві швидко та просто отримувати інформацію, розміщену на багатьох гіпертекстових веб-сторінках.

Google Chrome — це найбільш розповсюджений, безкоштовний веб-переглядач, розроблений компанією Google на основі веб-переглядача з відкритим кодом та програмного забезпечення. Інтерфейси всіх браузерів оснащені багатьма інструментами, та обов'язково включають розширення [2].

Розширення браузера – комп'ютерна програма, яка певним чином розширює функціональні можливості веб-браузера, підвищує його безпеку та продуктивність. Залежно від браузера і версії, цей термін може відрізнятися від подібних термінів, таких як плагін або доповнення.

Плагін – це незалежний компільований програмний модуль, що динамічно підключається до основної програми, призначений для розширення або використання її можливостей. Оскільки розширення та доповнення для різних браузерів не є однаковим та тотожним, слід звертатися до відповідної бази даних залежно від браузера. Іконки плагінів відображаються на панелі інструментів. Панель інструментів – елемент графічного інтерфейсу користувача, на якій розташовані кнопки управління реалізацією функцій, які найбільш часто використовуються.

Основна програма(браузер) надає сервіси, які плагін може використовувати. До них відноситься надана плагіну можливість зареєструвати себе в основній програмі, а також протокол обміну даними з іншими плагінами. Плагіни є залежними від сервісів, що надаються основною програмою, і переважно окремо не використовуються [3]. На противагу їм, основна програма незалежно оперує плагінами, надаючи кінцевим користувачам можливість динамічно додавати й оновлювати плагіни без необхідності внесення в неї змін.

Вперше плагіни з'явилися в Chrome 4.0, а галерея була офіційно відкрита 25 січня 2010 року. На момент відкриття в ній було вже понад 1500 плагінів.

Плагіни Google Chrome дозволяють розширити можливості і функції браузера. Та, чи інша функція може бути корисна для певного кола користувачів, але не для всіх. Тому, плагіни дозволяють додавати в Google Chrome тільки потрібні можливості, уникаючи накопичення функцій, які не використовуються. Окрім того, фахівці вважають, що надмірно перенавантажувати браузер не варто. Відбирають лише найпотрібніші доповнення (для блокування реклами, конвертори одиниць вимірювання та інші).

Будь-який браузер має у своєму арсеналі ряд вбудованих функцій, та найчастіше користувачів не влаштовує його функціональність і переглядаючи інтернет-сторінки, виникає бажання завантажити відповідні плагіни.

Тому, область розширення функціоналу браузера, як от створення різноманітних плагінів, наприклад для порівняння курсів валют різних банківських установ чи плагіну для блокування реклами є актуальними серед розробників програмного забезпечення.

IV. ВИСНОВКИ

Проаналізувавши основні можливості та застосування плагінів браузера можна зробити висновок, що дані додатки суттєво розширюють функціональність та зручність користування веб-переглядачами, адже кожен користувач може обрати, яка саме інформація буде відображатись в нього на екрані, та доступ до якої буде здійснюватись безпосередньо натисканням кількох кнопок, та зміною відповідних налаштувань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Орлов. А. Потрібні програми для Інтернету. - Спб.: Пітер, 2006.
[2] Браузери: аналітичний огляд. / Комп'ютера. № 3, 2006.
[3] Топорков С. «Альтернативні браузери» М.: «ДМК Пресс» 2006р .- 320с.

Система сповіщення задимленості приміщення на базі одноплатного комп'ютера Raspberry Pi B

Загурська Ю.Г.
науковий керівник: Холявкіна Т.В.
Кафедра комп'ютерних інформаційних технологій
Навчально-науковий інститут комп'ютерних інформаційних технологій,
Національний авіаційний університет,
Київ, Україна
mailto:zagurska1@gmail.com

***Анотація* — робота присвячена створенню системи сповіщення задимленості житлового будинку чи офісу. В роботі запропоновано спосіб опрацювання даних від датчиків задимлення та температури на базі одноплатного комп'ютера Raspberry PI B та представлення отриманих даних у вигляді відправки сповіщень про задимленість по електронній пошті.**

Ключові слова — архітектура процесора, опитування датчиків задимленості MQ-2, модуль датчика температури, налаштування контактів GPIO, серверна частина, SMTP-сервер.

I. ВСТУП

На сьогоднішній день безпека житлових та комерційних приміщень є безперечно актуальною в наш час. Поступ науково-технічного прогресу наявний у всіх галузях та сферах життя людини. Розвиток різноманітних технологій, а саме новий підхід до створення програмного забезпечення, технологій проектування обчислювальних машин, застосування сучасних та вдосконалених алгоритмів логіки роботи процесорів та компонентів спричинив появу численної кількості різних за застосуванням та відносно доступних у ціновій категорії датчиків, плат та комп'ютерів.

Існує численна кількість зареєстрованих випадків загорянь у житлових приміщеннях, що не мали видимих причин та ознак. До причин виникнення вогню в більшості випадків відносять незадовільний стан електротехнічних пристроїв, стару та несправну проводку, несправність приладів опалювальних систем або ж необережне поводження із вогнем [1].

Питання запобіганню займання у житлових приміщеннях та офісах є надзвичайно важливим та необхідним, оскільки швидкість поширення вогню є моментальною, а наслідки пагубними. Задимленість внаслідок горіння різноманітних матеріалів спричинює задуху, погану видимість від чого людина може втратити свідомість та контроль над ситуацією, а що найгірше отримати опіки, що в сукупності призведе до смерті.

На сучасному ринку існує численна кількість сучасних систем пожежної безпеки як офісних приміщень, магазинів, приватних підприємств так і житлових квартир та будинків. Принцип забезпечення моніторингу та контролю над об'єктом безпосередньо залежить від типу обладнання, інтегрованості системи із сучасними системами зв'язку та засобів здійснення контролю над ними.

В залежності від типу взаємодії із потенційною загрозою комплекс засобів, направлених на захист може бути пасивним та активним. Пасивна система має на меті сповістити власника помешкання чи охоронну службу про наявність злочинця [2]. За способом передачі інформації охоронні системи прийнято класифікувати наступним чином:

- дротові (аналогові, адресні);
- бездротові (із наявним зворотнім зв'язком та без).

II. ПОСТАНОВКА ПРОБЛЕМИ

На тлі зростання попиту на охоронні системи та тенденцій розвитку новітніх технологій набуває актуальності мобільна система безпеки, що матиме ряд переваг, а саме моментальний зворотній зв'язок, можливість дистанційного відслідковування, моніторингу ситуації та сповіщення про надзвичайні ситуації.

Однак найбільше значення у даній роботі має саме технічна реалізація системи, що матиме налагоджену систему сповіщення про стан периметру приміщення внаслідок опитування датчиків задимленості та температури.

III. ОСНОВНА ЧАСТИНА

Вирішення поставленої задачі вимагає послідовного виконання всіх частин роботи. Програмування одноплатного комп'ютера *Raspberry Pi* полягає в першу чергу у конфігурації програмного забезпечення та написанні коду для приєднання датчиків диму та температури [5].

Програмна частина апаратної конфігурації здійснюється за допомогою бібліотек та можливостей високорівневої мови програмування *Python*.

Аналоговий резисторний датчик диму *MQ-2* (рис.1) визначає наявність диму у повітрі та сповіщає про це збільшенням напруги на виході [3]. Чим більше диму у повітрі тим більша напруга на виході. У свою чергу датчик диму *MQ-2* містить вбудований потенціометр, який дозволяє коректувати чутливість аналізатора.

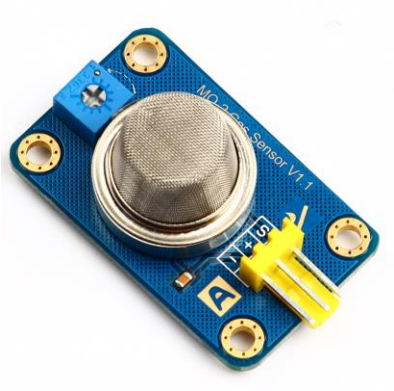

Рис.1 Аналоговий датчик диму MQ-2.

Модульні частини розробленого пристрою: датчик температури та диму "прослуховуються" за допомогою низькорівневого інтерфейсу вводу та виводу прямого управління [4] *GPIO (General Purpose Input Output)*, що власне і є ланкою на шляху назначених команд від процесора до будь-якого приєднаного модуля (рис.2).

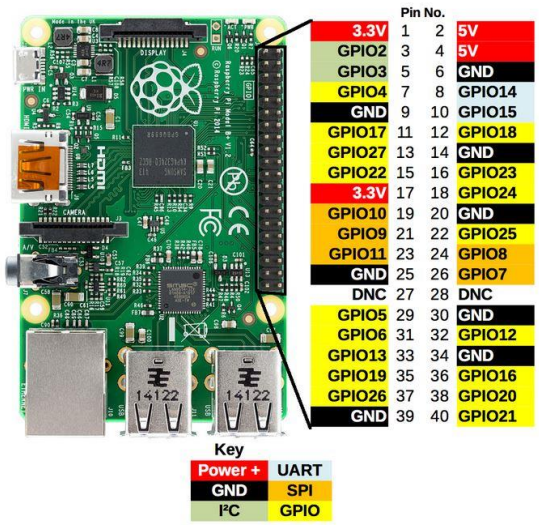

Рис.2 Конфігурація портів GPIO.

Для підключення датчика до одноплатного комп'ютера *Raspberry Pi* було застосовано аналогово-цифровий перетворювач (АЦП), в даному випадку мікросхема *MCP3002*, що безпосередньо під'єднується через порти *GPIO*.

Програмна конфігурація мікросхеми полягає у налагодженні роботи датчика із бібліотекою мови *Python bootbook_mcp3002*, котра підключається для управління мікросхемою і містить необхідний програмний код, що у свою чергу спрощує обробку аналогових даних, які поступають з датчика [6].

Конфігурація датчика температури відбувається аналогічним чином.

Для реалізації відправки повідомлень задіються можливості мови *Python*. Принцип "бібліотечності" *Python* включає можливість підключення спеціальної бібліотеки, що забезпечує підтримку *SMTP*-технологій обміну повідомленнями. Тобто немає необхідності у задіяні низькорівневого програмування сокетів [3].

Процедура полягає в наступному: при відправці клієнту-отримувачу поштового сповіщення поштовий клієнт сторони-відправника зв'язується зі своїм *SMTP*–сервером, який перенаправляє повідомлення на сервер клієнта-отримувача, а при перевірці своєї поштової скриньки буде отримана копія зі свого *IMAP*-сервера.

Поштові повідомлення складаються із простого тексту: рядка заголовку, пустого рядка і тіла повідомлення. Тіло повідомлення зазвичай представляється у *MEME*-кодуванні, оскільки чимало символів можуть бути відсутні у стандартному *ASCII*-кодуванні, так само і інші вкладення у електронних повідомленнях.

SMTP-сервер відповідає тільки за відправку повідомлень , отже у коді програмі необхідно вказати дані адрес поштових скриньок.

Генеруючи поштові повідомлення варто врахувати частоту їх відправки, навіть у випадку спрацювання системи, тобто наявності задимлення чи вогню, важливо не перенавантажувати сервер великою кількістю повідомлень, які відправляються із місця подій. Тому необхідно встановити спеціальну затримку відправки повідомлень – інтервал часу, протягом якого програма буде простоювати перед відправкою чергового повідомлення.

IV. ВИСНОВКИ

Описаний в статті метод створення системи сповіщення задимленості житлового помешкання дозволяє створити охоронну систему контролю, моніторингу та захисту будинку із високою швидкодією та достатньою надійністю.

Особливість роботи з одноплатним комп'ютером Raspberry PI B полягає у наявності гнучкого підходу для вибору та конфігурації модульних пристроїв.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Статистика українського науково - дослідницького інституту цивільного захисту за наказом ДСНС України.
[2] Магауенов Р.Г. , Системы охранной сигнализации. Основы теории и принципы построения , 2004. - 367 с.
[3] Tero Karvinen, Kimmo Karvinen, Ville Valtokari «Make:sensors», 2014-377 с.
[4] Виктор Петин, "Микрокомпьютеры Raspberry Pi.Практическое руководство, 2015 – 240 с.
[5] Simon Monk, "Programming the Raspberry Pi: Getting started with Python",2012 - 312 с .
[6] Ruth Suehle, Tom Callaway, "Raspberry Pi Hacks : Tips and tools or making things with inenexpensive linux computer", 2013.

Методи захисту інформаційних систем з використанням стеганографії

Іванова О.А.
науковий керівник: Савченко Аліна Станіславівна
Кафедра комп'ютерних інформаційних технологій,
Навчально-науковий інститут комп'ютерних інформаційних технологій,
Національний авіаційний університет,
Київ, Україна
ivanova7elena@gmail.com

***Анотація* — робота присвячена розгляду проблеми застосування стеганографічних методів захисту інформації в управляючих системах. Запропоновано бібліотеку «stegoDevLib» із застосуванням стеганографії, яка може динамічно приєднуватися до проектованих та існуючих інформаційних систем .**

Ключові слова — стеганографія, захист інформації, шифрування, LSB-стеганографія, метод приховування інформації з використанням таблиць квантування.

I. Вступ

Сучасний рівень розвитку обчислювальної техніки і швидкість удосконалення інформаційних технологій передбачають посилення вимог щодо збереження конфіденційної інформації. Найбільш поширеним засобом захисту інформації в інформаційному просторі є застосування криптографічних методів. Але, на сьогодні, їх використання не завжди є ефективним для забезпечення гарантованого збереження даних. В таких випадках актуальними стають методи стеганографії [1].

Стеганографія – це наука про приховане передавання інформації. Вона являє собою сукупність методів і їх реалізацій, які дозволяють зробити не явним самий факт передачі або зберігання даних.

Принцип стеганографії полягає в тому, що певний масив даних буде замаскований в іншому масиві, який заданий явно [2].

В дійсний час стеганографія активно використовується для розв'язку наступних базових задач: захисту конфіденційної інформації від несанкціонованого доступу; камуфляжу програмного забезпечення; підтвердження достовірності надісланої інформації; захисту авторського та виключного права на деякі види інтелектуальної власності; недекларованого зберігання інформації; створення прихованих джерел інформаційного витоку.

II. Постановка проблеми

З огляду на те, що стеганографія є досить поширеним методом шифрування даних, стає актуальним питання розробки доступного програмного інструменту для роботи з даною технологією.

Існує спеціальне програмне забезпечення, яке дозволяє для приховування корисних даних у різних типах контейнерів (зображення, аудіо, відео, текстових файлах) використовувати стенографічні методи [3]. Недоліком вказаних програмних продуктів є їх орієнтованість на потреби пересічного користувача. Такі програми не надають можливості для вирішення всіх питань щодо суцільного захисту конфіденційної інформації.

Саме тому, актуальною стає задача створення універсальної бібліотеки на базі стеганографічних методів для забезпечення захисту інформаційних систем. Бібліотека може стати елементом, що легко інтегрується, як в уже існуючі інформаційні системи, так і на стадії їх розробки.

III. Основна частина

Для розв'язку поставленої задачі була створена бібліотека «stegoDevLib». Бібліотека містить методи для перекодування даних, з наступним записом у обраний контейнер та відновлення за запитом користувача. Крім того, інструментарій запропонованої бібліотеки дозволяє вести тонке налаштування алгоритмів запису – щільність та порядок розміщення даних в контейнері, зміщення міток початку та кінця запису тощо. Використання бібліотеки «stegoDevLib» дозволить розробнику інформаційних систем у зручному форматі застосувати методи стеганографії та вести тонке налаштування алгоритмів для пакування інформації у зображенні.

В наданій реалізації використані методи LSB-стеганографії і метод приховування інформації з використанням таблиць квантування. Це дозволяє забезпечити одночасно достатній ступінь приховування інформації та прийнятну пропускну здатність. Перед здійсненням запису відбувається стиснення інформації, що підвищує корисну місткість контейнера. Крім того, надається можливість додаткового шифрування інформації методом RSA.

IV. Висновки

Призначення стеганографії у задачах захисту інформаційних управляючих систем полягає у доповнені криптографічних методів задля підвищення рівня їх безпеки. Запропонована бібліотека «stegoDevLib» дозволяє у зручному форматі використовувати методи стеганографії для захисту інформації при проектуванні та експлуатації інформаційних систем.

Список використаних джерел

[1] Хорошко В.О., Азаров О.Д., Шелест М.Є., Яремчук Ю.Є. Основи комп'ютерної стеганографії: Навч. посіб. для студентів і аспірантів. — Вінниця: ВДТУ, 2003.

[2] Конахович Г.Ф., Пузыренко А.Ю. Компьютерная стеганография. Теория и практика. К.: "МК-Пресс", 2006. — 288 с

[3] Інформаційний ресурс https://cryptoworld.su/

Інтерфейси взаємодії людини і комп’ютера

Ковальов С.С.
науковий керівник: Куклінський М.В.
Кафедра комп’ютерних інформаційних технологій,
Науково-навчальний інститут комп’ютерних інформаційних технологій,
Національний авіаційний університет,
Київ, Україна
stas.kovalov@ukr.net

***Анотація* – робота присвячена розгляду проблеми застарілої комунікації між людиною та комп’ютером (клавіатура та миша), а саме аналізу вирішення цієї проблеми за допомогою новітніх інтерфейсів взаємодії.**

Ключові слова – інтерфейс, інтерфейси взаємодії, інтерфейс SILK, мовний інтерфейс, біометричний інтерфейс, семантичний інтерфейс.

I. ВСТУП

В останні роки успіхи використання комп'ютерних систем в зростаючій мірі стали визначатися роллю, яку відіграють інтерфейси взаємодії користувача з програмним додатком. У загальному випадку поняття інтерфейсу досить широке поняття, і може бути застосовано до різних областей комп'ютерних технологій. Предметом вивчення в даній статті є інтерфейси в їх взаємозв'язку з програмно-апаратними інтерфейсами рівня додатків.

Інтерес до цієї області проявився з боку не тільки фахівців в області обчислювальної техніки, а й ергономістів, психологів, соціологів і розробників графічних систем, що свідчить про багатоплановий характер цієї проблеми.

Як часто показувала практика, для успішного вирішення багатьох прикладних задач в комп'ютерних системах необхідно комплексно розглядати проблеми, пов'язані з розробкою і (або) вибором інтерфейсів [1].

II. ПОСТАНОВКА ПРОБЛЕМИ

На даний час звичайна клавіатура і миша вже не задовольняють користувачів за ступенем зручності у використанні, видом вводу і швидкості надходження даних в комп'ютер.

III. ОСНОВНА ЧАСТИНА

Інтерфейс – в широкому сенсі слова, це спосіб (стандарт) взаємодії між об'єктами. Інтерфейс в технічному сенсі слова задає параметри, процедури і характеристики взаємодії об'єктів. Інтерфейс – це, перш за все, набір правил, і як будь-які правила, їх можна узагальнити, зібрати в «кодекс», згрупувати за спільною ознакою. Існує також поняття «вид інтерфейсу» – як об'єднання за схожістю способів взаємодії людини і комп'ютерів [2].

Для вирішення вищеописаної проблеми були розроблені наступні інтерфейси взаємодії: мовний, біометричний, семантичний, SILK.

Мовна технологія передбачає подачу команд голосом шляхом проголошення спеціальних стандартних слів, які повинні вимовлятися чітко, в одному темпі з обов'язковими паузами між словами. Це найпростіша реалізація SILK-інтерфейсу.

Біометрична технологія («Мімічний інтерфейс») використовує для управління комп'ютером вираз обличчя, напрямок погляду, розмір зіниці та інші параметри. Для ідентифікації користувача використовується його унікальна інформація, яка зчитується з цифрової камери, а потім за допомогою програми розпізнавання образів з цього зображення виділяються команди.

Семантичний (суспільний) інтерфейс передбачає відсутність команд при спілкуванні з комп'ютером. Запит формується на природній мові, у вигляді пов'язаного тексту і образів. По суті – це моделювання спілкування людини з комп'ютером. В даний час використовується для військових цілей.

Інтерфейс SILK (speech, image, language, knowledge) найбільш наближений до звичайної людської форми спілкування. В рамках цього інтерфейсу йде звичайна розмова людини і комп'ютера. При цьому комп'ютер знаходить для себе команди, аналізуючи людську мову і знаходячи в ній ключові фрази. Результати виконання команд він також перетворює в зрозумілу людині форму. Цей вид інтерфейсу вимагає великих витрат апаратури, тому знаходиться в стадії розробки і вдосконалення і використовується поки тільки у військових цілях.

SILK- інтерфейс для спілкування людини з машиною використовує мовну технологію; біометричну технологію (мімічний інтерфейс); семантичний інтерфейс.

IV. ВИСНОВКИ

В статті розкрито загальну та практичну цінність інтерфейсів взаємодії. Розглянуті інтерфейси, які в майбутньому будуть в широкому застосуванні і повністю витиснуть собою теперішній інтерфейс взаємодії, що сприятиме підвищенню швидкості вирішення поставлених завдань, комфорту взаємодії людини з комп'ютером, якості виконаної роботи. Будуть з'являтися все більш і більш нові розробки, покликані наблизити рівень взаємодії людини та машини до рівня природного спілкування між людьми.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] https://works.doklad.ru/view/Jtpon-JBkGM.html
[2] https://ua-referat.com/Користувальницький_інтерфейс

Особливості мови програмування ν (ню)

Кушнір Ю.О.
науковий керівник: Куклінський М.В.
Кафедра комп'ютерних інформаційних технологій,
Науково-навчальний інститут комп'ютерних інформаційних технологій,
Національний авіаційний університет,
Київ, Україна
juliadesh1505@gmail.com

***Анотація* – робота присвячена розгляду особливостей мови програмування ν (ню), а також визначенню в чому її переваги і недоліки.**

Ключові слова – мова програмування, алгоритм, синтаксис програми.

I. ВСТУП

Мова програмування ν (лат. ню) – об'єктна мова програмування. Синтаксис мови в дечому є С-подібним, типізація статична. Трансляція генерує з вихідного коду «проміжне представлення» – сукупність описів структур даних, функцій і відношень між ними [1].

Проміжне представлення в подальшому транслюється в llvm інструкції, що робить мову придатною для будь-якої платформи, для якої реалізований відповідний back-end.

На даний момент мова перебуває у стані активної розробки.

II. ОСОБЛИВОСТІ МОВИ

Основні особливості мови ν:

- відділення імен від сутностей;
- гнучкість механізмів іменування;
- каскадний препроцесинг;
- контекстно-залежні правила синтаксичного розбору.

Основні цілі мови:

- універсальність;
- швидкодія;
- боротьба з архаїзмами і необґрунтованими обмеженнями;
- можливість розвитку синтаксичних конструкцій через користувацькі бібліотеки (а не ядро мови).

На меті розробників є створення універсального і гнучкого ядра, навколо якого користувачі могли б сформувати зручний для себе інтерфейс.

У своєму використанні мова ν опирається на активне застосування новітніх підходів до розробки код-стандартів і середовищ розробки. Ідея в тому, щоб не обмежувати розробника, а надати йому якомога більше, аби потім з цієї множини кожен міг виокремити для себе необхідні засоби.

Ню ν не прив'язаний до будь-яких обмежень платформи, будь то 4 байта в цілому числі, 8 біт в байті, тощо. Саме це визначає кросплатформеність мови.

Препроцесор мови реалізується таким же чином (на тій же мові), що і вихідний код. Це дозволяє генерувати код програми використовуючи будь-які конструкції мови, прямо на етапі компіляції.

Відкинувши несуттєві деталі, мову можна розділити на три основні компоненти: імена, схеми і алгоритми. Перші – дозволяють розробнику розрізняти сутності під час написання програми, другі – представляють собою контейнери для даних, треті – алгоритми над цими даними.

III. ОСНОВНА ЧАСТИНА

Імена у мові ν призначені для зручної ідентифікації об'єктів користувачами. Вони відділені від решти механізмів мови, даючи можливість оперувати іменем в незалежності від того, якій сутності воно належить (схема, алгоритм, блок, тощо).

Схеми є шаблонами майбутніх змінних, аналогами класів, типів і структур з відомих нам мов програмування. Схеми до певної міри визначають структуру і поведінку об'єктів які будуть створені «за їх виглядом і подобою». Також вони описують загальну структуру програми, умовно розбиваючи її на області, які вирішують окремі задачі.

Алгоритми визначають поведінку об'єктів, операції над даними. Кожен алгоритм описує необхідні синтаксичні умови для своєї реалізації, дозволяючи розширити множину варіантів свого застосування у коді програми. Алгоритми є аналогами функцій, властивостей і процедур, але з більшою претензією до синтаксису. Оголошуються алгоритми всередині схем і декларують поведінку для усієї множини її екземплярів. Мова ν – певною мірою експериментом, спробою дослідити можливі переваги відкритого доступу до формування синтаксису.

IV. ВИСНОВКИ

Ню ν – контекстно-залежна синтаксично-орієнтована компільована мова з сильною статичною типізацією. Можна було б сказати, що це мова програмування, але теоретично застосування ν програмуванням не обмежується. Мова націлена на створення предметно-орієнтованих мов (Domain-Specific Language – DSL) на його основі, автоматизація розробки, генерації коду і його аналізу. Можливість гнучких налаштувань синтаксису дає змогу уникнути багатьох проблем, формуючи предметно-орієнтовану мову поверх базового функціоналу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] https://gamedev.ru/flame/forum/?id=230576

Мова програмування Kotlin як майбутнє операційної системи Android

Марчук В.С.
науковий керівник: Куклінський М.В.
Кафедра комп'ютерних інформаційних технологій
Навчально-науковий інститут комп'ютерних інформаційних технологій,
Національний авіаційний університет,
Київ, Україна
fcliverpool9832@gmail.com

***Анотація* – робота присвячена розгляду проблеми розробки операційної системи Android та її додатків. В роботі запропонований огляд мови програмування Kotlin.**

Ключові слова — мова програмування, додатки, Kotlin, Android, Java.

I. Вступ

Можна з певністю сказати, що основним мобільним пристроєм на сьогодні являється смартфон. Вони зараз є майже в кожної сучасної людини, особливо в тої, що безпосередньо пов'язана з IT-індустрією.

На сьогодні смартфони, здебільшого працюють на двох операційних системах від компаній-гігантів – Android OS від Google та IOS від Apple. На відміну від IOS, Android являється відкритим програмним забезпеченням, що дозволяє йому швидко прогресувати. Мета даної роботи – огляд досить нової мови програмування Kotlin, як засіб написанні коду на Android.

Android – операційна система і платформа для мобільних телефонів та планшетних комп'ютерів, створена компанією Google на базі ядра Linux. Підтримується альянсом Open Handset Alliance (OHA). Базовим елементом цієї операційної системи є реалізація Dalvik віртуальної машини Java, і все програмне забезпечення і застосування спираються на цю реалізацію.

II. Постановка проблеми

При програмуванні мовою Java спеціалісти нерідко помічають, як важко їм писати цією мовою, особливо при створенні великих проектів. Особливо інколи не вистачає логічної конструктивності в мові. Саме мова Kotlin створена для вирішення цієї проблеми.

III. Основна частина

Kotlin (Ко́тлін) – статично типізована мова програмування, яка працює поверх віртуальної машини Java і розробляється компанією JetBrains [1]. Також компілюється в JavaScript.

Мову названо на честь острова Котлін у Фінській затоці, на якому розміщена частина Кронштадту.

Автори ставили перед собою мету створити лаконічнішу та безпечнішу мову, ніж Java, і простішу, ніж Scala. Наслідками спрощення, в порівнянні зі Scala, стали також швидша компіляція та краща підтримка IDE [2].

Мова розробляється з 2010 року, публічно представлена в липні 2011 року. Код мови було відкрито в лютому 2012 року. З 17 травня 2017 року входить в список офіційно підтримуваних мов для розробки додатків для платформи Android [3].

На сервісі Github вже більше 200 репозиторіїв з кодом на Kotlin, а якщо говорити про мобільні додатки, то Kotlin використовувався при розробці месенджера Telegram [4].

Для написання коду потрібна IDE – інтегроване середовище розробки (наприклад, IntelliJ IDEA, або Eclipse), плагін Kotlin, gradle і Android SDK [5].

До найголовніших особливостей мови Kotlin можна віднести:

- компактність коду: швидко писати, легко читати, менше наведених помилок;
- вбудований захист від помилок, перш за все – nullability, тобто вимога до програміста явно вказувати, що якась змінна може приймати значення null;
- легкість в освоєнні (початківець-розробник на Java освоїть Kotlin без проблем) [6];
- у мові є підтримка first-class функцій. Це означає, що функція – це вбудований в мову тип для якого є спеціальний синтаксис. Функції можна створювати за місцем, передавати в параметри інших функцій, зберігати на них посилання;
- extension methods – полягає в можливості визначити метод для типу окремо від його (типу) оголошення. Така функція, звичайно, не буде віртуальною і ніяк не змінює класу, до якого ми додаємо метод, однак дозволяє додати як утилітарну функціональність для вже існуючого коду, так і розвантажити інтерфейс від цих же утилітарних методів;
- у змінних не вказані типи – компілятор Kotlin виводить їх, якщо це можливо і не заважає зрозумілості інтерфейсу.

Взагалі, мова зроблена таким чином, щоб максимально позбавити людину за клавіатурою від набрання зайвих знаків: короткий, але зрозумілий синтаксис з мінімумом ключових слів, відсутність необхідності точок з комою для розділення виразів, висновок типів, де це доречно, відсутність ключового слова new для створення класу.

IV. Висновки

Через високу сумісність з Java і можливість замінювати старий код поступово, в майбутньому Kotlin міг би стати гарною заміною Java в великих проектах і зручним інструментом для створення невеликих проектів з перспективою їх розвитку. Простота мови і її гнучкість дає розробнику більше можливостей для написання швидкого, але якісного коду.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] https://uk.wikipedia.org/wiki/Kotlin
[2] https://habrahabr.ru/post/150104/
[3] https://habrahabr.ru/post/277479/
[4] https://habrahabr.ru/post/322116/
[5] https://habrahabr.ru/company/JetBrains/blog/231525/
[6] https://dou.ua/lenta/articles/java-vs-kotlin/

Принцип роботи хмарних сховищ

Мороз Б.П.
науковий керівник: Куклінський М.В.
Кафедра комп'ютерних інформаційних технологій
Навчально-науковий інститут комп'ютерних інформаційних технологій,
Національний авіаційний університет,
Київ, Україна
bogdanmoroz2004@ukr.net

Пухальський Б.М.
науковий керівник: Куклінський М.В.
Кафедра комп'ютерних інформаційних технологій
Навчально-науковий інститут комп'ютерних інформаційних технологій,
Національний авіаційний університет,
Київ, Україна
bohdanpukhalsky@ukr.net

***Анотація* – дана робота розглядає принципи роботи хмарних сховищ, історію їх розвитку та технології які вони використовують.**

Ключові слова -- хмарне сховище, прикладний програмний інтерфейс, веб-служба.

I. ВСТУП

Армія консервативних прихильників класичного зберігання файлів на домашньому вінчестері поступово перетворюється у невелику роту, адже віртуальні контейнери вже давно довели, що вони більш зручні та мають безліч переваг. Серед таких переваг можна виділити: гнучкість (надають доступ з будь-якого пристрою), надійність (всі розробники подібних сервісів роблять бекапи) та наявність додаткових інструментів для синхронізації.

II. ПОСТАНОВКА ПРОБЛЕМИ

На сьогодні не кожний знає що таке хмарне сховище, його структуру та як воно працює. Багато хто впевнений що краще та безпечніше зберігати інформацію на домашньому комп'ютері, але чому тоді передові компанії світового масштабу переходять на цей вид збереження інформації? Ця доповідь спрямована на те, щоб розповісти що таке хмарне сховище, як воно виникло та як ним користуватися.

III. ОСНОВНА ЧАСТИНА

Хмарне сховище – являє собою цифрову модель зберігання, де дані зберігаються в логічних пулах, фізичне зберігання охоплюється кількома серверами (часто у різних локаціях), а фізичне середовище, як правило, належить хостинговим компаніям, які керують цим середовищем [1]. Постачальники хмарних систем відповідають за зберігання наявної інформації та доступ до неї, а також за роботу фізичного середовища, яке, як правило, належить до хостингових сервісів зберігання об'єктів.

Послуги хмарного сховища можуть бути доступні через прикладний програмний інтерфейс веб-сервісу (API) або додатки, які використовують API [2].

Хмарні обчислення винайшов Джозеф Ліклайдер у 1960-х роках під час його роботи над ARPANET, щоб дати людям вільний доступ до даних у будь-якій точці планети та будь-коли.

У 1983 році, CompuServe запропонував своїм споживачам для використання 128 k дискового простору, як сховище для зберігання будь-яких файлів [2].

І вже у 1994 році AT&T запустила PersonaLink – Інтернет-платформу для особистого та ділового спілкування на підприємництві. Даний сервіс був одним із перших повністю орієнтований під Web.

Широке використання багатьма компаніями хмарних сховищ викликані, насамперед, їх перевагами:

- компанії платять тільки за фактичний обсяг інформації, яка зберігається, і як правило, помісячно;
- компанії, які використовують хмарне зберігання, можуть скоротити споживання енергії на 70%, що робить їх бізнес "зеленішим";
- питання захисту та зберігання даних є відповідальністю постачальника послуг, а не компанії;
- завдання обслуговування даних у сховищі, а також питання розширення обсягу під її зберігання також є відповідальністю постачальника послуг;
- хмарне сховище може надавати користувачам миттєвий доступ до широкого спектру ресурсів і додатків, розміщених в інфраструктурі іншої організації через інтерфейс веб-служби;
- хмарне сховище може бути використане для копіювання образу віртуальної машини із хмари на локальне місце, або імпортування образу віртуальної машини з локального місця в бібліотеку хмари. Крім того, хмарне сховище може використовуватися, для переміщення образів віртуальних машин між обліковими записами користувачів або між центрами обробки даних;
- хмарне сховище може бути використане, для убезпечення від стихійних лих через наявність резервного копіювання. Зазвичай постачальники послуг виділяють 2 або 3 різні сервери резервного копіювання, які територіально розділені.

Серед найпопулярніших на сьогодні хмарних сховищ можна виділити: Dropbox; Microsoft OneDrive; Box; OziBox; Syncplicity; Яндекс.Диск; Google Drive; MediaFire; OpenDrive; Mega [3].

Однозначної відповіді який кращий немає. Проте все залежить від задач, які ставляться перед сервісом. Наприклад, Google Drive може похвалитися великою кількістю дуже зручних додаткових інструментів, Mega ж за замовченням надає найбільший безкоштовний об'єм і при цьому використовує алгоритм «нульового доступу».

IV. ВИСНОВКИ

У роботі розглянуто тему хмарних сховищ, наведено приклади найпопулярніших з їх перевагами. Хмарні сховища можуть полегшити опрацювання інформації на комп'ютерах, її обмін та використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Маркелов А.А. OpenStack. Практическое знакомство с облачной операционной системой / Андрей Маркелов, 2-е изд., перераб. и доп. – М.: «ДМК Пресс», 2016. – 160с.
[2] Риз Дж. Облачные вычисления / Джордж Риз, пер. с англ. – СПб.: БХВ-Петербург, 2011. – 288 с.
[3] Хмарні сховища «Вікіпедія» [Електронний ресурс] Режим доступу: https://uk.wikipedia.org/wiki/Хмарні_сховища

Сучасні Web-фреймворки

Ніс Д.В.
науковий керівник: Куклінський М.В.
Кафедра комп'ютерних інформаційних технологій
Навчально-науковий інститут комп'ютерних інформаційних технологій,
Національний авіаційний університет,
Київ, Україна
duniyannis@gmail.com

***Анотація* — робота присвячена сучасним web-фреймворкам. Аналізу back-end та front-end фреймворків, їх переваги та недоліки використання.**

Ключові слова — фреймворк, програмне забезпечення, WEB-технологія, back-end, front-end.

I. ВСТУП

Розробка веб-сайтів є перспективним напрямком в розвитку інформаційних технологій, адже глобальна мережа Інтернет охоплює з кожним роком все більше і більше користувачів. В Інтернеті знаходяться мільйони веб-сайтів різного спрямування. Створення та підтримка яких є важкою працею. Розвиток Інтернету нерозривно пов'язаний з проектуванням сайтів. Масова поява сайтів спровокувала проблему їх якості.

Популярність створення веб-ресурсів сприяла розробці різних систем і програм, які спрощують процес написання сайту. Також вони допомагають підвищити ефективність роботи, а також дозволяють розробнику сфокусуватися над основною логікою програми.

Такі технології, як PHP, Java, Microsoft.Net, MySQL, Oracle, Microsoft SQL Server і розроблені на їх основі фреймворки – це каркаси системи або підсистеми, що можуть включати допоміжні програми, мови сценаріїв і все, що полегшує розробку та об'єднання різних компонентів [1].

Фреймворки за останній час набрали популярність, і стали базовою платформою для розробки WEB-додатків. Іншими словами можна сказати, що вони забезпечують основну структуру програми. Використання фреймворків, дозволяє економити велику кількість часу, зменшити навантаження на процес розробки, позбавляючи від проблеми повторюваного коду, і швидко створювати додатки.

Проаналізувавши інформацію з мережі Інтернет, можна побачити що існує сотні фреймворків для створення web-додатків. Тому досить складно зробити вибір фреймворку, так як кожен з них має велику кількість привабливих функцій та доповнень. А неправильний вибір фреймворку може стати основною причиною невдачі проекту.

Порівнюючи фреймворки слід обрати такі критерії порівняння, які будуть найкраще демонструвати переваги та недоліки кожного. Також потрібно оцінити які фреймворки є найбільш простими та дозволяють створити веб-сайт швидше за все, а які надають сайту найбільше можливостей й гнучкості [2]. Результатом роботи стане список критеріїв порівняння, які будуть демонструвати які технології краще використовувати в яких ситуаціях, а також опис основних можливостей.

II. ПОСТАНОВКА ПРОБЛЕМИ

Для того щоб скористатися всіма можливостями фреймворка потрібний чималий багаж знань в розробці додатків. PHP–фреймворки можуть допомогти усунути дуже розповсюджену помилку при програмуванні додатків, а саме повторення коду, а також систематизувати процес розробки. Фреймворк є потужним інструментом для такої мови програмування як PHP, який допомагає організувати її код.

Кожна людина має різні вподобання і потреби. Для одного розробника використання фреймворків може допомогти у прискоренні процесу програмування, а для іншого це може здатися марною тратою часу. У більшості випадків це залежить від рівня професіоналізму, але, загалом, фреймворки призначені, щоб заощадити час і абстрагуватися від рутинних завдань. В основному, фреймворки застосовуються для розробки проектів складніше, ніж дво- або тристорінковий сайт з текстовими сторінками.

III. ОСНОВНА ЧАСТИНА

Фреймворк – це програмне забезпечення, що полегшує розробку і об'єднання різних компонентів великого програмного проекту. Ця платформа підходить для створення сайтів, бізнес-додатків і веб-сервісів. На відміну від бібліотек, які об'єднують набір підпрограм близької функціональності, фреймворк містить у собі велику кількість різних за призначенням бібліотек.

Широкого вжитку також набуло слово «каркас», а деякі автори використовують його в якості основного, взагалі не беручи до уваги англомовний аналог. Можна також говорити про каркасний підхід як про підхід до побудови програм, де будь-яка конфігурація програми будується з двох частин: перша (постійна частина) – каркас, в якому конфігурації не пов'язані між собою і містяться гнізда, в яких розміщується друга частина (змінна) – змінні модулі або точки розширення [3]. Для того щоб відповісти на питання, що має входити до фреймворка, щоб останній розглядався як WEB-технологія, необхідно розглянути сучасні види фреймворків:

- фреймворки програмної системи;
- фреймворки додатків;

- фреймворки концептуальної моделі.

Фреймворк програмної системи – це каркас системи або підсистеми, що може включати допоміжні програми, мови сценаріїв – все, що полегшує розробку й об'єднання різних компонентів. Фреймворк відрізняється від поняття бібліотеки тим, що бібліотека може бути використана в програмному продукті просто як набір підпрограм з близькою функціональністю, не впливаючи на архітектуру програмного продукту і не накладаючи на неї ніяких обмежень. У той час як фреймворк диктує правила побудови архітектури додатку, задаючи на початковому етапі розробки поведінку за замовчуванням, каркас, який потрібно буде розширювати і змінювати відповідно до зазначених вимог. До цього виду фреймворків відносяться і фреймворки для WEB.

Фреймворк додатку має стандартну структуру. Зі зростанням необхідності у графічних інтерфейсах користувача з'явилася і необхідність у фреймворках додатків. З їх допомогою простіше розробляти засоби для автоматичного створення графічних інтерфейсів. Для створення фреймворку додатків використовують об'єктно-орієнтоване програмування. Перший подібний фреймворк написала компанія Apple для Macintosh. Спочатку він був створений за допомогою Паскаль, а з часом його було переформатовано у C++.

Фреймворк концептуальної моделі – це абстрактне поняття даної структури для визначення способів вирішення конкретної проблеми [4].

Також необхідно з’ясувати, що ж саме являють собою WEB фреймворки. WEB фреймворки – це каркас, призначений для створення динамічних веб-сайтів, мережевих додатків, сервісів або ресурсів. Він спрощує розробку і позбавляє від необхідності написання рутинного коду. Багато фреймворків спрощують доступ до баз даних, полегшують розробку інтерфейсу, а також зменшують дублювання коду.

Виділяють п'ять типів веб-фреймворків: Request-based, Component-based, Hybrid, Meta і RIA-based.

Request-based – фреймворки, які безпосередньо обробляють вхідні запити. Збереження стану відбувається за рахунок серверних сесій. Приклади: Django, Ruby on Rails, Struts, Grails.

Component-based – фреймворки, які абстрагують обробку запитів всередині стандартних компонентів і самостійно стежать за станом. Дані каркаси мають багато спільного зі стандартними програмними графічними інтерфейсами. Приклади: JSF, Tapestry, Wicket.

Hybrid-based – фреймворки, які комбінують Request-based та Component-based фреймворки, беручи під свій контроль всі дані і логічний потік в заснованої на запиті моделі. Розробники мають повний контроль над URL, формами, параметрами, cookies і pathinfos. Однак замість того, щоб відобразити дії і контролери безпосередньо до запиту, гібридні фреймворки забезпечують об'єктну модель компонентів, яка поводиться тотожно в багатьох різних ситуаціях, таких як окремі сторінки, перервані запити, подібні порталу фрагменти сторінок та інтегровані віджети. Компоненти можуть розподілятися окремо і ефективно інтегруватися в інші проекти. Приклади: RIFE.

Meta-based – фреймворки, що мають ряд базових інтерфейсів для загального обслуговування і основу, яка легко розширюється з метою інтегрування компонентів і служб. Приклад: Keel.

RIA-based – фреймворки для розробки Rich Internet Applications (RIA) – фреймворки, що служать для розробки повноцінних додатків, які запускаються всередині браузера. Приклад: Flex.

Найбільш поширеними є Request-based і Component-based веб-фреймворки. Зібравши і проаналізувавши інформацію, було виділено такі характерні компоненти web фреймворків:

- шаблонізатор – відповідає за незалежність верстки від програмного коду;
- роутер – розпізнає URL, за яким відбулося звернення до сервера;
- модуль доступу до бази даних;
- модуль кешування – прискорює завантаження сторінок;
- модуль безпеки – аутентифікація і авторизація користувачів;
- файли конфігурації.

Веб-фреймворки також можуть керувати сесіями, вести логи, спрощувати використання технології Ajax та ін.

IV. ВИСНОВКИ

Системи створення web-додатків або фреймворки активно використовуються розробниками при створенні web-додатків із різним функціоналом і рівнем складності. Оглянувши основні характеристики і можливості сучасних фреймворків можна вибрати оптимальний варіант для конкретних web-додатків з урахуванням поставлених завдань.

Вибір та використання фреймворків відіграє важливу роль при проектуванні, реалізації та супроводі як простих web-додатків, так і складних програмних комплексів. Вибираючи фреймворк, необхідно звертати увагу на те, наскільки він легкий в освоєнні і розумінні. Це має дуже важливе значення для малодосвідченого програміста.

Також завжди необхідно переконуватися, що база даних і web-сервер сумісні з архітектурою обраного фреймворку. Крім цього поширеною помилкою ще є неправильна установка фреймворку. Тому під час установки необхідно чітко слідувати інструкції, щоб уникнути помилок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] https://uk.wikipedia.org/wiki/Програмний_каркас

[2] https://web-creator.ru/articles/about_frameworks

[3] www.dbhelp.ru/what-is-framework/page/

[4] https://habrahabr.ru/post/253297/

[5] https://artjoker.ua/ru/big-brain/glossary/framework/

Інтеграція прикладних програм за допомогою віддалених викликів

Озімай Д.О.
науковий керівник: Холявкіна Т.В.
Кафедра комп'ютерних інформаційних технологій
Навчально-науковий інститут комп'ютерних інформаційних технологій,
Національний авіаційний університет,
Київ, Україна
naphanyaoz@gmail.com

***Анотація* — робота присвячена розгляду проблеми обробки однакових даних різними інформаційними системами. В роботі запропонований розгляд інтеграції прикладних програм за допомогою віддалених викликів.**

Ключові слова — загальна БД, інформаційна система, інтеграція прикладних програм, віддалені виклики, API-інтерфейси.

I. ВСТУП

Сьогодні жваво обговорюється комп'ютерним суспільством питання інтеграції програм. Загальні цілі інтеграції програм можливо сформулювати наступним чином: зменшення вартості експлуатації сукупності програм підприємства; збільшення швидкості виконання типових завдань або гарантувати його термін виконання; підвищення якості виконання завдань за рахунок формалізації процесів та мінімізації людського фактору, як основного джерела помилок. Успішна інтеграція корпоративних систем дозволяє досягти і додаткових цілей – забезпечити автоматизований контроль проходження основних бізнес-процесів на підприємстві, інформаційну безпеку при реалізації бізнес-процесів тощо.

II. ПОСТАНОВКА ПРОБЛЕМИ

Не існує інформаційних систем, які самостійно мають змогу задовольнити потреби сучасних підприємств. Середні і великі організації зазвичай експлуатують не менше десятка багатокористувацьких систем, а іноді і тисячі. У цих системах доволі таки часто оброблюються однакові данні – починаючи із довідників та класифікаторів. Звичайні ситуації, коли в межах одного бізнес-процесу задіяні різні інформаційні системи. Зараз існує багато WEB-програм, які покращують нам життя, але не завжди зручно, а іноді і неможливо перемикатись з однієї на іншу, щоб переносити інформацію між ними.

III. ОСНОВНА ЧАСТИНА

Для взаємодії програм звичайно використовуються такі методи, як обмін файлами, загальна БД, віддалений виклик і асинхронний обмін повідомленнями.

Розглянемо переваги та недоліки одного із методів, а саме інтеграція віддалених викликів.

Стандарти на віддалений виклик процедур виникли два десятка років тому, дозволяючи програмному коду, який виконується на одному комп'ютері, викликати код на іншому. Стандарти з'явились, розвивались і згасли: RPC, CORBA, DCOM, RMI ... останнім в цьому ряду став протокол SOAP, основа сучасних WEB-сервісів. У підході до інтеграції з використанням віддалених викликів за ці роки нічого принципово не змінилось – якщо програмі А щось потрібно від програми Б, то А одним із перерахованих способів викликає функцію програми Б [1].

Перевагою віддаленого виклика процедур є його синхронність, завдяки якій виклик виконується негайно. Іноді ця перевага обертається недоліком. Якщо виклик не може бути виконаний негайно (наприклад, тому що до мережі немає доступу), то він не спрацює. Якщо б виклик був асинхронний, то він би повторювався поки процедура віддаленої програми не була успішно викликана [2].

Тому основний недолік віддаленого виклика – потреба у працездатності усіх задіяних програм під час взаємодії. Уявіть собі систему ведення довідників, зміни з якої кожну ніч розповсюджуються в десятки корпоративних систем. Ймовірність того, що, скажемо, у дві години ночі усі корпоративні системи знаходяться у робочому стані повної працездатності, маленька [3].

IV. ВИСНОВКИ

Проаналізувавши метод інтеграції за допомогою віддалених викликів можна зробити висновок, що цей метод актуальний тільки тоді, коли взаємодія програм ініціюється користувачем, який сам контролює результат. Для автоматичної взаємодії без участі людини даний підхід майже не можливий.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Морозова О.А. Интеграция корпоративных информационных систем: М80 учебное пособие. — М.: Финансовый университет, 2014. — 140 с.

[2] Шаблоны интеграции корпоративных приложений. : Пер. с англ. – М. : ООО «И.Д. Вильямс», 2007.– 672 с.: ил. – Парал. Тит. Англ.

[3] Грегор Хоп, Бобби Вульф Шаблоны интеграции корпоративных приложений.: Пер. с англ. .– М: Издательский дом "Вильямс", 2016.– 672 с.

Веб-додаток. Розробка фронтенду.

Омельянюк М.О.
науковий керівник: Холявкіна Т.В.
Кафедра комп'ютерних інформаційних технологій
Навчально-науковий інститут комп'ютерних інформаційних технологій,
Національний авіаційний університет,
Київ, Україна
nikitaom12344321@gmail.com

***Анотація* — в роботі розкрито важливість і необхідність зрозумілого і якісного зробленого фронтенду для web-додатку. В роботі розглянуто і запропоновано мови розмітки і програмування для розробки фронтенду.**

Ключові слова — web-додаток, фронтенд, бекенд, web-сторінка, сервер, клієнт.

I. ВСТУП

Сучасний світ тісно пов'язаний з IT. Люди все більше часу проводять в Інтернеті і вже звикли до того що основні банківські операції, покупки у магазині, або ж оплату будь-яких послуг можна робити через Інтернет, не виходячи з дому. І для виконання цих дій існують безліч web-додатків.

Web-додаток – це клієнт-серверний додаток, у якому клієнт взаємодіє з сервером за допомогою браузера, а за сервер відповідає web-сервер. Логіка веб-додатку розподілена між сервером і клієнтом, зберігання даних здійснюється, переважно, на сервері, обмін інформацією відбувається по мережі. Однією з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-додатки є міжплатформенними службами.

Web-додаток складається з клієнтської і серверної частин, тим самим реалізуючи технологію «клієнт-сервер». Клієнтська частина реалізує інтерфейс користувача, формує запити до сервера і обробляє відповіді від нього. Серверна частина отримує запит від клієнта, виконує обчислення, після цього формує web-сторінку і відправляє її клієнту через мережу з використанням протоколу HTTP (Рис.1). Саме сукупність web-сторінок інтерфейсу користувача і є фронтендом web-додатку [1].

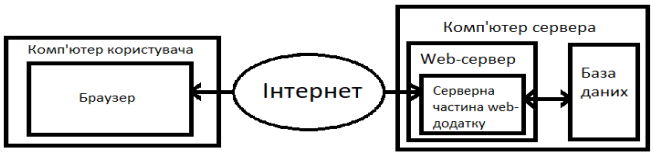

Рис.1. Організація взаємодії інтерфейсу користувача із серверною частиною web-додатку.

II. ПОСТАНОВКА ПРОБЛЕМИ

З кожним роком веб-додатки стають все більшими та складнішими. На теперішній час майже усі веб-додатки мають можливість показувати нові дані без оновлення усієї сторінки браузера. Швидкість роботи браузера зменшується через те що окрім виведення розмітки сторінки і підключення стилів, у сучасних web-додатках є обробка інформації і зміна відображення елементів в залежності від дій користувача. Тому для прискорення роботи браузера при розробці клієнтської частини використовують такі технології як Angular 2 та ReactJS, які дозволяють збільшити швидкість візуалізації елементів інтерфейсу та контенту браузером, а також надають зручні інструменти для розміщення інформації на web-сторінці.

III. ОСНОВНА ЧАСТИНА

В основі розробки фронтенду для web-додатків лежить той самий принцип що і при розробці звичайних статичних web-сторінок. Спочатку розробляється каркас сторінки, за допомогою мови розмітки HTML, а для надання зовнішнього вигляду сторінок використовується каскадні таблиці стилів – CSS [2]. У разі якщо розробник хоче додати який-небудь функціонал або сценарій на свою web-сторінку він може зробити це за допомогою динамічної, об'єктно-орієнтованої мови програмування – JavaScript [3]. Програма, написана на JavaScript і яка виконується на web-сторінці, може керувати вбудованими в сторінку компонентами, тим самим реалізуючи призначений для користувача інтерфейс з багатими можливостями.

Як правило в теперішній час для створення візуально зрозумілої і зручної web-сторінки (фронтенду web-додатку) треба використовувати усі вище перераховані мови розмітки і програмування, і це як мінімум, адже навідміну від web-сторінок звичайного сайту, web-додаток завжди має якийсь функціонал, при якому виконується постійне оновлення певних елементів на сторінці і обробка інформації, які можуть відбуватися настільки часто і в такій великій кількості що постійне повне перезавантаження web-сторінки може суттєво знизити швидкодію браузера. Тому в такому випадку ще може використовуватись такий підхід для побудови web-сторінок, як AJAX. AJAX – це підхід до побудови користувацьких інтерфейсів web-застосунків, за яких web-сторінка, не перезавантажуючись, у фоновому режимі надсилає запити на сервер і сама звідти довантажує потрібні користувачу дані, за рахунок чого ми використовуємо набагато менше Інтернет трафіку і збільшуємо швидкодію браузеру.

IV. ВИСНОВКИ

В роботі запропоновано принцип розробки фронтенду для web-додатку. Цей принцип є універсальним але не усюди є можливість його використання через те що для деяких більш складних web-додатків може знадобитися інші технології які можуть краще оптимізувати взаємозв'язок фронтенду і бекенду web-додатку та збільшити його функціонал на стороні клієнта.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] https://ru.wikipedia.org/wiki/Веб-приложение.
[2] Фрейн – HTML5 CSS3 – разработка сайтов для любых браузеров
[3] Профессиональное программирование – Прохоренок Н.А. – HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера (4-е издание)

Вдосконалення Web-додатку засобами платформи RIA

Пухальський Б.М.
науковий керівник: Куклінський М.В.
Кафедра комп'ютерних інформаційних технологій
Навчально-науковий інститут комп'ютерних інформаційних технологій,
Національний авіаційний університет,
Київ, Україна
bohdanpukhalsky@ukr.net

Мороз Б.П.
науковий керівник: Куклінський М.В.
Кафедра комп'ютерних інформаційних технологій
Навчально-науковий інститут комп'ютерних інформаційних технологій,
Національний авіаційний університет,
Київ, Україна
bogdanmoroz2004@ukr.net

***Анотація* – робота присвячена розгляду архітектури RIA, її переваги та необхідність створення RIA додатків. Описується порівняння класичних HTML додатків та RIA, та приклади платформ для створення «багатих» додатків.**

Ключові слова – RIA, Web-додаток, HTML, HTTP, клієнт-сервер

I. ВСТУП

Web-розробка в наш час є однією з найбільш затребуваних галузей IT. Не секрет, що на сьогодні майже всі компанії потребують популяризації свого імені та своєї діяльності. Найкращий спосіб зробити це – сайт або будь-який інший Web-додаток.

Вимоги до сайту мають бути наступними:

- багатофункціональність (що зазвичай виходить за межі можливостей барузера);
- швидкість роботи, в тому числі вивантаження та надсилання даних із сервера, динамічне обновлення інформації;
- швидкість розробки додатку;
- автономність Web-додатку.

Всі ці проблеми можна вирішити за допомогою створення RIA (Reach Internet Application), що дослівно перекладається як багатий Інтернет-додаток [1]. Незважаючи на те, що дана технологія була розроблена ще наприкінці 90-х років, популярності вона набула лише 2-3 роки тому. Архітектура подібної програми кардинально відрізняється від архітектури класичного Web-додатку та забезпечує дотримання вищезгаданих вимог.

II. ПОСТАНОВКА ПРОБЛЕМИ

Звичайні сайти-сервіси працюють по принципу переходу на іншу сторінку по гіперпосиланням і відправлення форми на сервер засобами веб-браузера. Іншими словами, робота таких програм сконцентрована навколо клієнт-серверної архітектури з тонким клієнтом. HTML, являючись мовою розмітки документів що відображається браузером, ідеально для цього підходить. Послідовність дій користувачів представляє собою постійне відправлення запитів на сервер. При такому підході виникає ряд проблем:

- збереження користувацьких даних між сеансами роботи програми та їх синхронізація з сервером;
- проблема відправлення і отримання даних з сервера тільки по мірі необхідності, а не на кожну дію користувача;
- проблема запуску додатку при відсутності підключення до мережі.

ТАБЛИЦЯ 1.ТЕХНОЛОГІЇ СТВОРЕННЯ WEB-ДОДАТКІВ

Платформа	GWT	Flex/AIR	JavaFX	Java-аплети	Silverlight
Розробник	Google Inc.	Adobe Inc.	Sun Microsystems	Sun Microsystems	Microsoft
Мова створення додатків	Java	Actionscript	JavaFX Script	Java	C++, C#, Visual Basic,
Підтримка браузером клієнта	100% із увімкнутим Javascript	~97-98% браузерів	~70% браузерів	~70% браузерів	~50% браузерів

III. ОСНОВНА ЧАСТИНА

RIA – це Web-додатки, що мають характеристики графічних прикладних програм для ПК. RIA збудовані на потужних інструментах для розробки, можуть працювати швидше та бути більш привабливими. Вони надають користувачам кращий візуальний інтерфейс та більшу інтерактивність ніж традиційні додатки що використовують лише HTML та HTTP [2].

Всі RIA мають схожу особливість – наявність проміжної частини, яка передається по мережі клієнту та відповідає на взаємодію з сервером і відображення інтерфейсу користувача, що перевершує HTML-аналоги. На рис.1 можна побачити принципи організації взаємодії з користувачами в HTML та RIA додатках.

Поступовий розвиток даного підходу привів до реалізації подібних технологій на практиці. Найвідоміші з них наведено у таблиці 1 [3].

IV. ВИСНОВКИ

У роботі розглянуті особливості RIA-додатків та принцип їх дії. Також розглянуті найпопулярніші платформи для розробки подібних програм та їх порівняння.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1]. Rich Internet Applications – інтернет джерело, тип доступу: https://www.computerworld.com/article/2551058/networking/rich-internet-applications.html

[2]. Clarke J., Connors J., Bruno E., JavaFX: Developing Rich Internet Applications 1st Edition - Boston: Addison-Wesley, 2009, – 359p.

[3]. Tartakovsky A., Rasputnis V., Fain Rich Y. Internet Applications with Adobe Flex and Java: Secrets of the Masters – Woodcliff Lake,: SYS-CON Media 2007, – 707p.

Проблеми використання Memcached при організації DDoS-атак

Романенко Д.А
науковий керівник: Куклінський М.В.
Кафедра комп'ютерних інформаційних технологій
Навчально-науковий інститут комп'ютерних інформаційних технологій,
Національний авіаційний університет,
Київ, Україна
Dima.romanenko.200036@gmail.com

Валюх В.О
науковий керівник: Куклінський М.В.
Кафедра комп'ютерних інформаційних технологій
Навчально-науковий інститут комп'ютерних інформаційних технологій,
Національний авіаційний університет,
Київ, Україна
vladyslav.valiukh@ukr.net

Анотація **– робота присвячена розгляду проблеми незахищеності від DDoS-атак і конфіденційності мережі. В роботі показано приклад використання інформації про мережу для виконання DDoS-атак,спосіб знаходження інформації про мережу та спосіб вирішення проблеми захисту від DDos-атаки.**

Ключові слова – DDos-атака, Memcached, Shodan, Github.

I. ВСТУП

Сучасний розвиток Інтернет-мереж набирає обертів. Разом з цим збільшується необхідність в забезпеченні безпеки і стабільності роботи мережі.

Найтиповішим прикладом атаки на мережу є DDoS. DDoS – тип мережевої атаки заснований на обмеженості ресурсів служби, на яку проводиться атака. Мережевий ресурс виходить з ладу в результаті безлічі запитів відправлених до нього з різних точок.

Одна із таких атак нещодавно була проведена на GitHub – найбільший веб-сервіс для хостингу IT-проектів і їх спільної розробки.

На GitHub обрушилася найсильніша DDoS-атака за всю історію спостережень, пікова потужність якої сягала 1,35 Тб/сек. Фахівці пояснюють, що виною всьому не черговий ботнет, а ампліфікація DDoS за допомогою вразливих Memcached-серверів.

II. ПОСТАНОВКА ПРОБЛЕМИ

Memcached – це програмне забезпечення, що реалізує сервіс кешування даних в оперативній пам'яті на основі хеш-таблиці. Фактично, Memcached допомагає адміністраторам зняти навантаження з бази даних, поліпшивши продуктивність веб-додатків і полегшивши завдання, пов'язані з масштабуванням. Memcached використовують тисячі сайтів (близько 93 000 серверів, за даними Shodan), включаючи таких гігантів, як Facebook, Flickr, Twitter, Reddit, YouTube, GitHub.

Раніше фахівці не раз попереджали про початок хвилі атак, посилених за допомогою Memcached, і писали, що в даному випадку запит, об'ємом всього 15 байт, може перетворюватися у відповідь розміром 134 Кб (тобто запит на 203 байт обернеться відповіддю на 100 мегабайт) [1]. При цьому дослідники попереджали, що для ампліфікації (посилення) атак вже використовуються сервери таких великих хостинг-провайдерів, як OVH, Digital Ocean, Sakura.

Тепер прогнози фахівців почали втілюватися в життя: жертвою рекордної за потужністю DDoS-атаки став GitHub [4]. Інженери компанії Akamai, першими зафіксували те, що трапилося, пишуть, що даний інцидент не можна порівнювати навіть з гучними атаками ботнетів Mirai, що відбулися в 2016 році і до недавнього часу вважалися найбільш руйнівними.

Представники GitHub повідомили, що атака виходила від тисячі різних автономних систем і десятків тисяч унікальних ендпоінтов. У пікові моменти потужність атаки досягала 126,9 млн пакетів в секунду.

Тепер експерти попереджають, що найпотужніша атака на GitHub – це, швидше за все, тільки початок. Справа в тому, що раніше представники Qrator Labs і Cloudflare вже повідомляли про відображення Memcached-атак потужністю 260-480 Гб/сек [2]. Тепер сталася атака потужністю 1,35 Тб/сек. Але ще в 2017 році дослідники компанії 0Kee Team, одними з перших виявили даний вектор ампліфікації, писали, що посилені за допомогою Memcached атаки можуть досягати і 2 Тб/сек [3].

III. ОСНОВНА ЧАСТИНА

Відразу три proof-of-concept утиліти для організації DDoS-атак з ампліфікацією допомогою Memcached були опубліковані в мережі.

Перша утиліта – це написаний на Python скрипт Memcacrashed.py, який сканує Shodan в пошуках уразливих Memcached-серверів. Скрипт дозволяє користувачеві відразу ж використовувати отримані IP-адреси для посилення DDoS-атаки. Автором даного рішення є ІБ-фахівець, провідний блог Spuz.me.

Друга утиліта була опублікована на Pastebin ще на початку поточного тижня, її автор невідомий. Інструмент написаний на C і комплектується списком з 17000 IP-адрес, що належать вразливим Memcached-серверів. Скрипт запускає DDoS-атаку, використовуючи адреси зі списку для її ампліфікації.

Третя представляє з себе експлоїт.

Загалом той факт, що тепер зловмисникам не потрібно купувати машини, розробляти або скачувати різні програми і софти, а для потужної DDoS атаки необхідно лише просканувати сайти на наявність незахищених портів, змушує задуматися над самою концепцією безпеки, оскільки на даний момент цей спосіб атаки є найбюджетнішим і найпотужнішим відносно інших.

Щодо вирішення проблеми фахівці настійно рекомендують блокувати або обмежити UDP для порту 11211, на якому працює Memcached. Так як Memcached «слухає» INADDR_ANY, і підтримка UDP включена за замовчуванням, адміністраторам рекомендується відключити UDP зовсім, якщо це можливо, а краще захистити сервер файрволом або розмістити в закритій мережі. Докладні інструкції можна знайти, наприклад, в блозі компанії Cloudflare [5].

IV. ВИСНОВКИ

В статті описано один з найперспективніших на даний момент спосіб DDoS-атаки, а саме DDoS-атаки з використанням Memcached. Також були розглянуті причини, засоби отримання інформації про мережу, і методи захисту від цієї атаки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] https://xakep.ru/2018/03/02/github-memcached-ddos/

[2] https://blog.cloudflare.com/memcrashed-major-amplification-attacks-from-port-11211/

[3] https://habrahabr.ru/company/pentestit/blog/252233/

[4] https://ru.wikipedia.org/wiki/GitHub

[5] https://lolzteam.net/threads/390545/

Адаптивний web-сайт вантажоперевезення

Середа В.В.
науковий керівник: Холявкіна Т.В.
Кафедра комп'ютерних інформаційних технологій
Навчально-науковий інститут комп'ютерних інформаційних технологій,
Національний авіаційний університет,
Київ, Україна
vlad.sereda.1997@gmail.com

***Анотація* — робота присвячена розгляду проблеми адаптації web-сайтів для якісного відображення контенту на дисплеях пристроїв з різною роздільною здатністю. В роботі розглянуто мови розмітки і програмування для правильної побудови web документа, а також медіа-запити для правильного відображення елементів фронтенду на різних пристроях.**

Ключові слова — web-сайт, адаптивність, медіа-запити.

I. ВСТУП

В сучасному світі, людині тяжко уявити своє існування без Інтернету. Мережа Інтернет вже давно перейшла до другої фази свого розвитку – умовно цю фазу можна назвати Інтернет 2.0. Якщо в першій фазі можна було говорити про просте відображення Інтернет сторінок для передачі інформації, то друга фаза цілком присвячена сервісам, які б могли полегшити існування людини в сучасних умовах. Людина постійно занята справами (робота, навчання і т. д.), часу на те, щоб йти до магазину за кожною дрібницею стає все менше. Не дивно, що сервіси у вигляді Інтернет сторінок стають все більш популярними.

Web-сайт – це сукупність документів пов'язаної гіпертекстової інформації, котрі були розміщені в Інтернеті і доступні для відкриття на будь-якому пристрої з активним підключенням до Інтернету.

II. ПОСТАНОВКА ПРОБЛЕМИ

Коли розвиток світової мережі Інтернет тільки починався, інтернет-сайти відкривалися на персональних комп'ютерах, розмір екрана в яких був переважно однаковий.

З часом, технології стали стрімко розвиватися, що призвело до того, що навіть мобільні телефони отримали доступ до Інтернету і тим самим з'явилася проблема сумісності веб-сайтів, котрі спочатку були розроблені для відображення на великих дисплеях персональних комп'ютерів.

Їх було практично нереально відкривати на мобільних телефонах, оскільки текст відображався занадто обширно. Потрібно було якесь рішення для цієї проблеми.

Ними стали медіа запити (більш відомі як Media Query) [2]. Вони були додані разом з третім поколінням CSS. За допомогою них можна задати певне значення роздільної здатності того чи іншого дисплея для якого буде призначатися web-сайт. В залежності від кількості вказаних пікселей, буде відкриватися різні таблиці стилів.

III. ОСНОВНА ЧАСТИНА

В основі створення адаптивного web-сайту знаходиться той самий принцип, що і при створенні звичайної сторінки. Для початку використовується мова розмітки HTML. За допомогою неї елементи сторінки розміщуються у відповідності до певного, заздалегідь визначеного дизайну. За створення дизайну, як правило, відповідають дизайнери. Вони створюють візуальне представлення сторінки в десктопній та адаптиваній версії сайту у вигляді шаблону (частіше за все PSD макет).

Далі підключаються таблиці стилів CSS. За допомогою них, вказується стилізація тих чи інших елементів розмітки. Покращується їх зовнішній вигляд та вказуються правила їх відображення в тих чи інших web-браузерах, оскільки рушії в них можуть бути різні. (Особливо це стосується застарілих версій web-браузеру Internet Explorer від компанії Microsoft).

Для звичайної сторінки, цих складових буде достатньо, але деякі функції (особливо ті, що стосуються надання послуг, на кшталт вантажоперевезення), потребують додаткового функціоналу від web-сайту.

На допомогу приходить мова програмування JavaScript. Вона допомагає наповнити сторінку «життям», додаючи до неї обробників подій. Наприклад, навівши курсор на певну частину сайту, користувач може не тільки побачити вікно з підказкою, але також повноцінно взаємодіяти зі сторінкою.

А для того, щоб сайт мав гарний, а головне повноцінно діючий вид на екранах сучасних мобільних пристроїв (смартфонів, планшетів, ультрабуків і т.д.), використовуються медіа запити.

За допомогою правила @media, задаються данні пристрою, на якому буде відкриватися web-сайт. Найбільш популярним рішенням в наш час вважається Mobile-First створення сторінок.

Так називається процес, коли спочатку сторінка розробляється для портативних девайсів.

IV. ВИСНОВКИ

В статті описано метод адаптації web-сайту для роботи на пристроях з різною роздільною здатністю. Запропоновано метод, за допомогою якого це можна реалізувати.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] http://www.webtec.com.ua/uk/articles/index/view/2011-05-05/web-site
[2] http://htmlbook.ru/css/value/media

Застосування моделей теорії систем масового обслуговування АСУ для удосконалення вантажопотоків з використанням клієнтоорієнтованих технологій

Цейко Б.О.
науковий керівник: Кириченко Ганна Іванівна
Кафедра «Управління процесами перевезень»,
Факультет «Управління залізничним транспортом»,
Державний університет інфраструктури та технологій,
Київ, Україна
boris16@bigmir.net

***Анотація* — робота присвячена розгляду проблеми застосування моделей теорії систем масового обслуговування АСУ для удосконалення вантажопотоків з використанням клієнтоорієнтованих технологій.**

Ключові слова — системи масового обслуговування (СМО), автоматизовані системи управління (АСУ), клієнтоорієнтовані технології перевезень.

I. Вступ

Повсякчас кожній людині доводиться чекати на обслуговування в черзі (біля каси, терміналу обслуговування тощо). Аналогічні ситуації виникають, коли треба скористатися телефонним зв'язком або виконати свою програму на комп'ютері. Будь-яке виробництво теж можна уявити як послідовність систем обслуговування. До типових систем обслуговування належать також ремонтні і медичні служби, транспортні системи, термінали, аеропорти, вокзали і т.д. [1].

II. Постановка проблеми

Дослідженням питань удосконалення організації вантажопотоків на залізничному транспорті займалося багато науковців і вчених, але питання удосконалення вантажопотоків на залізниці з використанням клієнтоорієнтованих технологій повністю не вирішені і наразі є актуальними.

III. Основна частина

Особливого значення СМО набули у процесах інформатики. Це передусім комп'ютерні системи, мережі передавання інформації, операційні системи, бази і банки даних.

Концепція інтелектуальної транспортної системи управління процесами доставки вантажу, а також аспекти функціонування та роботи Інтелектуальної системи управління процесом доставки вантажу приведені у працях Г.І. Кириченко[2,3].

Формування автоматизованої технології просування групових поїздів оперативного призначення висвітлює у дисертаційному дослідженні Киман А.М.[4].

Питання формування методів управління розподілом пропускної спроможності залізничної інфраструктури в умовах недискримінаційного доступу розкриває у дисертаційному дослідженні Прохорченко А.В.[5].

Систему масового обслуговування загалом можна уявити як сукупність послідовно пов'язаних між собою вхідних потоків вимог на обслуговування (потоків замовлень), черг, каналів обслуговування і потоків обслужених замовлень. Будь-який пристрій, який безпосередньо обслуговує замовлення, називають каналом обслуговування[6].

Моделювання найпотужніший універсальний метод дослідження та оцінювання ефективності різноманітних систем, поведінка яких залежить від дії випадкових чинників.

Аналітичне моделювання полягає у побудові та дослідженні математичних моделей. У його основу покладено ідентичність форми рівнянь та однозначність співвідношень між змінними в рівняннях, які описують оригінал та модель.

Більшість СМО є клієнтоорієнтованими, оскільки створюються для задоволення потреб споживачів у різних сферах та галузях людської діяльності.

Клієнти залізниці – вантажовласники, вантажовідправники, вантажоодержувачі такаж с споживачами послуг, які їм надає залізниця, але питання клієнтоорієнтованості при вантажоперевезеннях повністю не вирішені і наразі є актуальні.

З цією метоє пропонується створення такої моделі обслуговування споживачів послуг, яка буде клієнтоцентричною, і ці завдання може вирішити застосування моделей теорії систем масового обслуговування АСУ.

Недоліками більшості аналітичних моделей, побудованих на основі понять теорії масового

обслуговування, є використання в них значних спрощень: зображення потоку замовлень як пуассонівського або найпростішого, припущення про показниковий розподіл часу обслуговування, неможливість обслуговування замовлень одночасно кількома каналами обслуговування тощо. Система масового обслуговування (СМО) – система, яка виконує обслуговування вимог, що надходять до неї . Обслуговування вимог у СМО проводиться обслуговуючими приладами. Класична СМО містить від одного до нескінченного числа приладів. В залежності від наявності можливості очікування вступниками вимогами початку обслуговування СМО поділяються на:

1. Системи з втратами, в яких вимоги, що не знайшли в момент надходження жодного вільного приладу, втрачаються;
2. Системи з очікуванням, в яких є накопичувач нескінченної ємності для буферизації надійшли вимог, при цьому очікують вимоги утворюють чергу;
3. Системи з накопичувачем кінцевої ємності (чеканням і обмеженнями), в яких довжина черги не може перевищувати ємності накопичувача; при цьому вимога, що надходить в переповнену СМО (відсутні вільні місця для очікування), втрачається.

Вибір вимоги з черги на обслуговування здійснюється за допомогою так званої дисципліни обслуговування. Їх прикладами є FCFS / FIFO (що прийшов першим обслуговується першим), LCFS / LIFO (що прийшов останнім обслуговується першим), RANDOM (випадковий вибір). У системах з очікуванням накопичувач в загальному випадку може мати складну структуру.

Виділяють такі основні поняття СМО:

- Вимога (заявка) – запит на обслуговування.
- Вхідний потік вимог – сукупність вимог, що надходять у СМО.
- Час обслуговування – період часу, протягом якого обслуговується вимогу.

Математична модель СМО – це сукупність математичних виразів, що описують вхідний потік вимог, процес обслуговування та їх взаємозв’язок.

Дослідженням питань удосконалення організації вантажопотоків на залізниці займалися багато науковців, але питання удосконалення організації вантажопотоків на залізниці з використанням клієнтоорієнтованих технологій повністю не вирішені.

IV. Висновки

Застосування моделей теорії систем масового обслуговування АСУ для удосконалення вантажопотоків дозволить запоровадити використання клієнтоорієнтованих технологій для вантажоперевезень на залізничному транспорті, що надасть користувачам цих послуг нові можливості при перевезенні вантажу.

Системи обслуговування відіграють значну роль у повсякденному житті. Досвід моделювання різних типів дискретних систем свідчить про те, що приблизно 80% цих моделей ґрунтуються на СМО.

Використання СМО при моделюванні систем, які включають клієнтоорієнтовані технології зумовлене науково – технічним прогресом, а також удосконаленням вже існуючих систем.

Список використаних джерел

[1] Кутковецький В.Я. Ймовірнісні процеси і математична статистика в автоматизованих системах: Навчальний посібник. - Миколаїв: Вид-во МДГУ, 2002. – 150С.

[2] Концепція інтелектуальної транспортної системи управління процесами доставки вантажу/ Г. І. Кириченко //Залізничний транспорт України. — 2013. — Вип. 1. — С. 37—40.

[3] Інтелектуальна система управління процесом доставки вантажу / Г. І. Кириченко //Інформаційно – керуючі системи на залізничному транспорті. — 2015. — Вип. 5(114). — С. 3—6.

[4] Киман А.М. Формування автоматизованої технології просування групових поїздів оперативного призначення [Текст]: дис… канд. техн. наук *: 05. 22. 01* /Киман Андрій Миколайович. — Харків, 2017. — 178с.

[5] Прохорченко А.В. Формування методів управління розподілом пропускної спроможності залізничної інфраструктури в умова недискримінаційного доступу [Текст]: дис… докт. техн. наук *: 05. 22. 01* /Прохорченко Андрій Володимирович. — Харків, 2015. — 412с.

[6] Жерновий Ю.В. Імітаційне моделювання систем масового обслуговування: Практикум. - Львів: Видавничий центр ЛНУ імені Івана Франка, 2007.- 307с.

Веб-додаток. Розробка бекенду.

Цуран А.В.
науковий керівник: Холявкіна Т.В.
Кафедра комп'ютерних інформаційних технологій
Навчально-науковий інститут комп'ютерних інформаційних технологій,
Національний авіаційний університет,
Київ, Україна
avtsuran@gmail.com

***Анотація* — робота присвячена розгляду розробки веб додатків. В роботі запропоноване основне рішення при проектуванні та розробки веб-додатку на платформі Java.**

Ключові слова — веб-сервер, веб-додаток, веб-сторінка, інтерфейс, фреймворк, шаблон, патерн, модуль.

I. ВСТУП

Веб-додаток — розподілений додаток, в якому клієнтом виступає браузер, а сервером — веб-сервер. Браузер може бути реалізацією так званих тонких клієнтів — логіка додатку зосереджується на сервері, а функція браузера полягає переважно у відображенні інформації, завантаженої мережею з сервера, і передачі назад даних користувача. Однією з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-додатки є міжплатформовими сервісами.

II. ПОСТАНОВКА ПРОБЛЕМИ

Сьогодні веб-додатки набирають популярність. найбільш відвідуваними сайтами стають не чисто інформаційні, гіпертекстові сайти, а ті, які надають будь-якої сервіс, будь-яким чином взаємодіють з користувачем. Але навіть і звичайні інформаційні сайти часто використовують системи управління контентом для зручності управління інформацією, так що і їх теж можна зарахувати до веб-додатків.

III. ОСНОВНА ЧАСТИНА

Мова Java спочатку позиціонувалась виробником, як мова для Інтернету. Перш за все існує кілька конкурентних серверів веб додатків, які не дивлячись на відмінності дотримуються деяких стандартів, встановлених Sun, а значить більшість додатків без будь-яких значних модифікацій можуть бути перенесені з сервера на сервер [1].

Крім того існує кілька рівнів складності та з різними підходами фреймворків для розробки веб-додатків. Це фреймворки для структурування додатків на основі патерну MVC, бібліотеки для побудови шаблонів веб-сторінок, бібліотеки для відображення реляційної таблиці в об'єкти і навпаки.

Підходи до побудови веб-додатків розвивалися еволюційно. Самим простим і ймовірно найпершим підходом було писати веб-додатки точно також, як звичайні консольні додатки. Тобто вся логіка і весь висновок програми знаходиться безпосередньо в одному файлі і ніяк не розділені.

Це було пов'язано з певними незручностями. Найпомітніше з них полягало в тому, що програма захаращувати величезною кількістю викликів процедури виведення тексту сторінки. Тому висновок шаблонних даних і безпосередньо програму вирішено було умовно розділити за допомогою спеціальних синтаксичних конструкцій (як в PHP або JSP).

І як тільки позбулися самого помітного недоліку вихідного підходу, увагу було звернуто на інший – проблема поділу логіки та інтерфейсу. Для цього в рамках використання попереднього підходу накладають ще одне обмеження: в тексті шаблону повинні бути присутніми тільки інструкції, що впливають на надання інформації, а вся логіка йде в окремі класи, описують модель додатки.

Найбільш зручним для побудови веб-додатків виявився патерн об'єктно-орієнтованого програмування під назвою MVC (Model, View, Controller). Він також використовується при побудові звичайних десктопних додатків.

Суть його в тому, що додаток організовується у вигляді трьох різних модулів: один для моделі програми, що включає уявлення даних і логіку системи, другий для представлення даних і введення даних від користувача і третій для контролера, який обробляє запити і передає управління різних модулів [2].

Переваги такого підходу до проектування системи полягають в наступному. MVC розділяє різні аспекти проектування (зберігання даних, поведінка системи, подання та управління), дає можливість повторного використання коду, централізує управління виконанням програми та спрощує внесення змін до системи.

Існують варіації MVC так звані Model 1 і Model 2, також відомі як Page Centric і Servlet Centric відповідно. Їх відмінність в тому, що в першому код контролерів міститься в JSP-файлах, а в другому він міститься в сервлетах, які передають управління JSP-файлів в кінці свого виконання. Перевага останнього способу в тому, що він дозволяє вибрати користувачеві сторінку на основі даних, що містяться в запиті. Крім того він покращує розділення різних аспектів програми та збільшує рівень абстракції класів.

IV. ВИСНОВКИ

У даній роботі було проведено дослідження засобів розробки веб-додатків і історії їх розвитку. Веб–додаток не вимагає установки на клієнтський комп'ютер будь-якого програмного забезпечення крім браузера.

Проаналізувавши основні можливості патерну MVC можна зробити висновок, що цей патерн є найбільш зручним для побудови веб-додатків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Budi Kurniawan. Java for the Web with Servlets, JSP, and EJB: A Developer's Guide to J2EE Solutions. New Riders Publishing, 2012.

[2] World Wide Web — Wikipedia, the free encyclopedia. (http://en.wikipedia.org/wiki/Www, 31.10.2014)