УДК 004.42

Р.В. Величанський, магістр, гр. ІПЗ-1
О.І. Грабар, к.т.н., доц., каф.ІПЗ
Житомирський державний технологічний університет

РОЗРОБКА ПРОГРАМНОГО КОМПЛЕКСУ ДЛЯ ПРОВЕДЕННЯ ВСЕУКРАЇНСЬКОГО КОНКУРСУ ЗМАГАННЯ ЛІСОРУБІВ

Development the mobile addition for prognostic of work-load of roads and taking into account of unforeseeable situations

На даний момент інформаційні технології набули широкого поширення у всіх сферах діяльності людини. Основною перевагою у застосуванні таких технологій є швидкість обробки великих об'ємів даних і можливість представлення результатів роботи обчислювального комплексу у зручному для людини вигляді. Так інформаційні технології не оминули увагою і різні види змагань та взяли на себе всю роботу по підрахунку і відображенню результатів. В даний час важко уявити собі змагання високого рівня без професійного інформаційно-технічного супроводу. Переважна більшість таких програмних комплексів базується на підрахунку результатів змагань у індивідуальних і командних видах спорту. У ігрових, таких як бадмінтон, сквош, теніс, настільний теніс, дартс та єдиноборствах, таких як бокс, боротьба, карате і ін. Але жоден з програмних продуктів, представлених на ринку не в змозі виконати поставлену задачу по проведенню конкурсу-змагання лісорубів через складність розрахунків та специфіку при виконанні кожної з вправ. Засновником проведення конкурсу-змагання лісорубів є International Association Logging Championships. Саме вона регламентує правила проведення змагань і проводить фінальні міжнародні змагання. Так вже багато років поспіль починаючи з 1998 року на базі ДП «Тетерівський лісгосп» проводиться Всеукраїнський конкурс-змагання лісорубів. Змагання проводяться за міжнародними правилами [http://www.ialc.ch/]. Консурс-змагання складається з 5 вправ: звалювання дерева, монтаж пильного ланцюга, розкряжування стовбура комбінованим різом, точне розкряжування і обрізування сучків.

За виконання кожної вправи нараховуються початкові бали, після чого до них додаються нараховані бали згідно встановлених показників і віднімаються штрафні бали, отримані за порушення правил проведення змагань. Розрахунок кількості нарахованих і штрафних балів проводиться за відповідними таблицями, наведеними у правилах «Правила чемпіонату світу серед лісорубів». Після завершення обробки всіх протоколів формуються результати змагань по окремим вправам і результати змагань у загального заліку.

Призери Всеукраїнського конкурсу-змагання отримують цінні призи від спонсорів, які обраховуються десятками тисяч гривень, тому вимоги до рівня проведення змагань, швидкості і якості оцінювання ставляться дуже високі. З переможців Всеукраїнських змагань в подальшому формується збірна команда України, яка представляє Україну на міжнародному конкурсі- змаганні лісорубів.

Упродовж багатьох років для проведення конкурсу-змагання лісорубів в Україні використовувалися підручні засоби, починаючи з програми EXCEL і закінчуючи власними розробками. Аналіз всіх цих розробок виявив значні недоліки. Так до них можна віднести обмежену кількість операторів, які обробляють протоколи змагань, складність приймання і вирішення спірних питань при оцінюванні конкурсів. Також відзначаються значні затримки часу на обробку протоколів і підрахунку проміжних і кінцевих результатів. З досвіду, затримка між оформленням останнього протоколу і видачою суддівською колегією кінцевих результатів може досягати годину і більше. Всі існуючі програмні комплекси неможливо швидко адаптувати при зміні правил проведення змагань, які час від часу змінюються.

Детальний аналіз попередніх напрацювань і використання сучасних технологій вказав шляхи для подальшої модернізації при розробці подібних систем. Використання клієнт-серверної архітектури при написанні програмного комплексу вирішує питання одночасної обробки протоколів змагань декількома операторами, а також дає змогу в режимі онлайн спостерігати за процесом оцінювання протоколів. Винесення всіх критеріїв оцінки в окрему базу даних дозволяє швидко адаптувати програмний комплекс при зміні правил без зміні коду основної програми. Додавши до програмного комплексу фотофіксацію протоколів можна буде вирішити питання оперативності реагування на помилково внесені.

Отже, запровадження передових інформаційних технологій і новітніх технічних засобів при проектуванні систем інформаційно-технічного супроводу проведення конкурсу змагання лісорубів може вивести організацію таких змагань на дуже високий рівень.

УДК 004.42

В.С. Генвальдт, магістр, група ПІ-47м
О.І. Грабар, к.т.н., доц. каф. ІПЗ
Житомирський державний технологічний університет

ПЕРЕВАГИ ВИКОРИСТАННЯ ТА ПЕРСПЕКТИВИ РОЗВИТКУ CRM-СИСТЕМ У КОМЕРЦІЙНІЙ ДІЯЛЬНОСТІ

Advantages of the using and prospect of development the CRM-systems in commercial activity

На сьогоднішній день найбільша частка діяльності людини припадає на комерційну діяльність. Дане поняття стало досить широким, оскільки охоплює не лише галузь торгівельної діяльності, а також сферу послуг. На початкових етапах ведення бізнесу може здатися, що застосування автоматизованих систем не дасть явних переваг, проте з ростом кількості клієнтів, а також за умови динамічного розвитку, стає очевидно, що контролювати велике підприємство, що динамічно зростає без застосування автоматизованих систем стає все важче та зростає ризик впустити щось важливе, що в свою чергу може призвести до значних фінансових або матеріальних збитків. Для вирішення даних проблем існує окремий клас програмного забезпечення – CRM-системи. Дані системи надають можливість отримати єдине формалізоване сховище даних та завдяки наявності модулів аналітики та аналізу даних, на основі яких можна побудувати загальну картину ведення справ та розвитку підприємства, а також сформулювати основні стратегії розвитку на майбутнє та оцінити перспективність обраних напрямків розвитку бізнесу.

Системи даного класу дозволяють проконтролювати повний цикл взаємовідносин з клієнтом, а саме зберігання електронних версій документів, зустрічей, листів або задач, які ставляться керівництвом перед працівниками підприємства. Деякі системи даного класу, що мають вузьку спеціалізацію надають можливість відслідкувати не лише взаємовідносини з клієнтом, а також виробничий цикл після якого клієнт отримує або товар або результат інтелектуальної власності. Несумнівним плюсом таких систем є автоматизація розрахунків, що стосуються фінансів, оскільки дана функціональність дозволяє зменшити фінансові ризики, які можуть виникнути через людський фактор.

Постачальники програмних продуктів на сьогоднішній день віддають перевагу організації систем за принципом SAAS (Software As Service), тобто відбувається перехід від традиційного програмного забезпечення до хмарних технологій, що в свою чергу підвищує їх ефективність та мобільність, а саме доступ з будь-якої точки світу не використовуючи ніяких спеціалізованих додатків окрім веб-браузера.

Також дані системи часто мають API що дозволяє користувачам інтегрувати систему з іншими програмними рішеннями та продуктами. До прикладу можна розглядати у якості таких систем системи інтернет банкінгу, системи IP-телефонії, системи SMS та E-mail розсилок або ынших програмних систем що виконують маркетингові функції. Також постачальники багатьох хмарних CRM-систем разом із своїм програмним продуктом надають мобільні додатки для підвищення комфорту, мобільності та ефективності використання системи, проте щоб підвищити ефективність роботи даних мобільних додатків слід зважати на той факт, що на сьогоднішній день існує декілька найбільш популярних мобільних платформ а саме Android, IOS, Windows Universal, а отже при розробці мобільних додатків слід робити їх також кроссплатформеними, що в свою чергу розширить користувацьку аудиторію.

Дана галузь програмного забезпечення є досить вигідною, оскільки за останні роки спостерігається значний ріст частки CRM-систем на світовому ринку, що в свою чергу може свідчити про те, що на програмні продукти даного класу є значний попит серед пересічних користувачів.

В останній час все більшої популярності набувають інтеграційні рішення, які розширюють функціональність CRM-систем та є необхідними для переважної більшості підприємств такі як:

– інтеграція з телефонією, що дає можливість фіксувати історію дзвінків клієнтам та зберігати записи розмов між менеджером та клієнтами;

– інтеграція з сервісом розсилок, що да можливість проводити промо-акції через поштові чи SMS сервіси;

– інтеграція з соціальними мережами, що є без сумніву найбільш ефективною зброєю залучення клієнтів, оскільки саме в соціальних мережах можна поширити інформацію на максимально велику аудиторію.

На основі вище наведеної інформації можна зробити висновок, що дана галузь програмного забезпечення є одною із найперспективніших та є вигідною для потенційних постачальників програмного забезпечення. Вона демонструє значний ріст на загальному ринку та має значний попит серед представників бізнесу, а також тенденції до перетворення даних програмних продуктів у хмарні системи. Також можна зробити висновок, що дана галузь буде розвиватися паралельно з розвитком потреб комерційних підприємств, а отже, і надалі буде залишатися перспективною для розробників програмного забезпечення.

УДК 004.424

Головня С.А., студ., гр. ПІ-49
Марчук Г.В., старший викладач каф. ІПЗ
Житомирський державний технологічний університет

ПАТЕРНИ ПРОГРАМУВАННЯ

Фабричний метод (англ. Factory Method також відомий як Віртуальний конструктор (англ. Virtual Constructor)) - породжує шаблон проектування, надає підкласам інтерфейс для створення екземплярів деякого класу. У момент створення спадкоємці можуть визначити, який клас створювати. Іншими словами, даний шаблон делегує створення об’єктів спадкоємцям батьківського класу. Це дозволяє використовувати в коді програми не специфічні класи, а маніпулювати абстрактними об’єктами на більш високому рівні.

Шаблон Фабричний метод слід використовувати коли:

– класу не відомо заздалегідь, об’єкти яких саме класів йому потрібно створювати;

– клас спроектовано так, щоб об’єкти, які він створює, специфікувалися підкласами;

– клас делегує свої обов’язки одному з кількох допоміжних підкласів, та потрібно локалізувати знання про те, який саме підклас приймає ці обов'язки на себе.

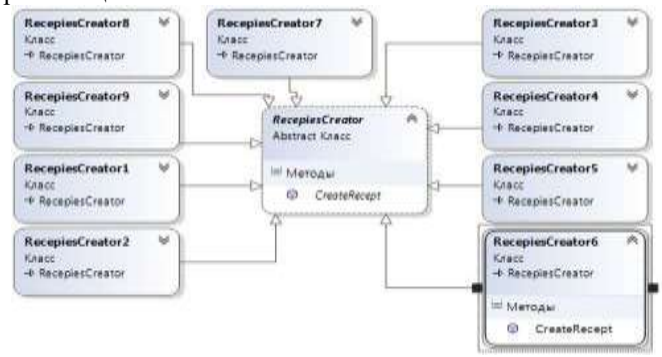

Рис. 1. Використання патерну у дипломному проекті

Творець «покладається» на свої підкласи у визначенні фабричного методу, який буде повертати екземпляр відповідного конкретного продукту.

Фабричні методи позбавляють проектувальника від необхідності вбудовувати в код класи, які залежать від додатку. Код має справу тільки з інтерфейсом класу RecepeisCreator, тому він може працювати з будь-якими певними користувальницькими класами конкретних продуктів.

Наслідки використання патерну:

– надає підкласам операції-зачіпки (hooks). Створення об’єктів всередині класу за допомогою фабричного методу завжди виявляється більш гнучким рішенням, ніж безпосереднє створення. Фабричний метод створює в підкласах операції-зачіпки для надання розширеної версії об’єкта;

– з’єднує паралельні ієрархії. Паралельні ієрархії виникають в разі, коли клас делегує частину своїх обов’язків іншому класу.

Розглянемо наступні питання, що виникають при використанні партерна фабричний метод:

– параметризовані фабричні методи. Це ще один варіант патерну, який дозволяє фабричному методу створювати різні види продуктів. Від фабричного методу передається параметр, який ідентифікує вид створюваного об’єкта. Всі об’єкти, що виходять за допомогою фабричного методу, поділяють загальний інтерфейс RecepeisCreator;

– два основні різновиди патерну. По-перше, це випадок, коли клас CreateRecept являється абстрактним і не містить реалізації оголошеного в ньому фабричного методу. Друга можливість: CreateRecept – конкретний клас, в якому за замовчуванням є реалізація фабричного методу. Рідко, але зустрічається і абстрактний клас, що має реалізацію за замовчуванням;

– мовно-залежні варіації і проблеми. У різних мовах виникають власні цікаві варіанти і деякі нюанси.

Результати застосування патерну Factory Method.

Переваги патерна Factory Method:

– Створює об'єкти різних типів, дозволяючи системі залишатися незалежною як від самого процесу створення, так і від типів створюваних об'єктів.

Недоліки патерна Factory Method:

– У разі класичного варіанту патерна навіть для породження єдиного об'єкта необхідно створювати відповідну фабрику

УДК-004.03

Дацюк Д.В., студ., гр. ПІК-14
Марчук Г.В., старший викладач каф. ІПЗ
Житомирський державний технологічний університет

СИСТЕМА ПРОДУКТИВНОЇ РОБОТИ

Актуальною проблемою сьогодення є відсутність достатньої кількості часу для виконання запланованих справ, що здебільшого провокує неповноцінна здатність людини втримати в своїй пам'яті і одночасно синхронізувати список завдань. Більшість приймають рішення використовувати звичні планери, де вміщують у виді певного рейтингу заплановане і відмічають вже виконані завдання деякими відмітками. Продуктивність методу суттєво падає останніми роками, бо система має дещо застарілий вигляд в теперішніх умовах автоматизації процесу. Людина зіштовхується з переліком перешкод: звичайний, друкований планер легко можна будь-де забути, що потягне за собою повну дезорієнтацію в майбутніх справах, також він не забезпечує ніяких нагадувань та синхронізації в ситуації зміни часу завдання чи появі непередбачуваних раніше справ у вже не актуальному списку, що може стати причиною їх втрати або плутанини. Цей метод є повністю застарілим и не бажаним для використання. Якщо звернути увагу на більш сучасний варіант використання планеру у вигляді нотатників на усіляких гаджетах, варто зазначити, що такий додаток вже має декілька переваг перед своїм попередником: він присутній майже скрізь з користувачем, що дає змогу швидко звертатися до нього в разі необхідності, проте декілька проблем все ще лишаються наявні.

Проаналізувавши ситуацію, було прийнято рішення створити сучасний вихід з ситуації недостатнього планування та розподілу часу - віртуальний органайзер, проте і в основну ідею покласти не тільки занесення своїх справ по мірі надходження, а й певну ідею.

Існує декілька способів планування подій: матриця Ейзенхауера, SMART та GTD. Матриця Ейзенхауера (рис.1) являє собою досить примітивний спосіб занесення своїх планів до матричного списку, розділяючи їх за певними характеристиками. У підсумку отримується розфасовані завдання на чотири категорії: «важливо та терміново», «не важливо, але терміново», «не важливо, але не терміново», «важливо, але не терміново». Спосіб є досить корисним для використання, проте потребує серйозних часових витрат для того, щоб систематизувати свої справи, розділивши їх по всіх напрямках. Таким чином, тайм-менеджмент стає практично ще одним завданням, яке можна було б занести в дану матрицю, на що людина точно не розраховує, починаючи використовувати методи планування. На додаток, спосіб матриці Ейзенхауера включає в собі ряд правил, яким потрібно слідувати, використовуючи його: необхідно слідкувати за тим, щоб квадрат з категорією «важливо і терміново» був максимально порожнім, що на думку автора зменшить напругу на людину та звільнить її від небажаних затримок в справах. Проте, весь метод, не дивлячись на досить добре вибудовану теорію, на практиці має досить утопічний вигляд за рахунок активності та мінливості життя людини.

Матриця не є гнучким методом планування, а її використання може бути необхідним лише на рівні дуже низького уміння планувати та розподіляти свій час, скоріше задля навчального посібника для засвоєння подібних навичок.

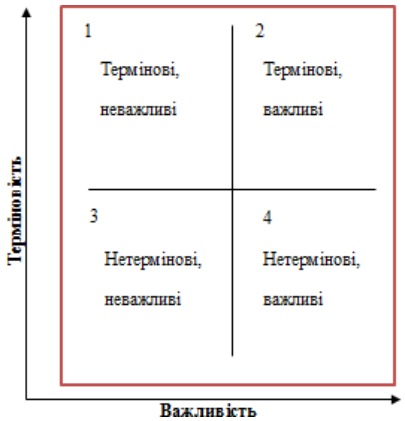

Рис. 1. Матриця Ейзенхауера

Наступний метод SMART дуже продуктивний метод для довгострокового, зваженого планування справ на основі генерування максимально чітких та конкретних цілей, які прийнято називати «розумними».

SMART – це абревіатура, широко застосовна в сфері проектного управління та менеджменту для постановки завдань. Так позначається своєрідна методика грамотного встановлення цілей. Кожна буква визначає собою слово, що є характерним для неї.

- S – Specific, що перекладається, як «конкретний».
- M – Measurable, тобто «вимірний».
- A – Attainable, що перекладається, як «досяжний».
- R і T – Relevant і Time-bound («актуальне» і «обмежений за часом»).

Розробник ідеї впевнений, що правильне формулювання цілі та її повноцінне усвідомлення є гарантом успіху в її досягненні. Механізм системи планування «SMART» дуже продуктивний для розроблення цілих стратегій чи завдань, які потребують навіть здебільшого логічних прорахунків, а не економії часу: ідея не передбачає в собі намагання якнайшвидше виконати заплановане, нічого не забувши і ніде не запізнившись, вона скоріше націлена на максимально продуктивний результат. Таким чином для повсякденного, буденного планування «SMART» не виглядає доцільною.

Найбільш корисною та відповідною ідею, яку можна було б покласти в основу саме буденного органайзеру є система GTD, яка передбачає врахування не лише найважливіших справ, про які люди саме і боїться забути, а також і менш суттєвих, які ми ніколи не фіксуємо, проте намагаємось тримати в голові і при цьому втрачаємо продуктивність. GTD пропонує врахувати і занотувати всі плани, які вже згенеровані на поточний день. Таким чином, пам'ять буде повністю позбавлена необхідності контролювати потік справ, а мозок зможе сповна сконцентруватися за максимальній виробничій здатності, позбавившись стресу. Ще одним принципом GTD є регулярна перевірка плану і внесення найменших змін або доповнень, навіть якщо їх суттєвість є мінімальною. Необхідним моментом і доповненням звичайно є здатність органайзеру нагадати користувачу про нагальну справу сповіщенням, бажано, якщо не одним, а через декілька ресурсів. Дана маніпуляція максимально знизить ризик напруження мозку і пам'яті через побоювання втратити інформацію. Таким чином можна окреслити основний massage системи GTD – «Мозок створений для генерації ідей, але не їх збереження.».

Однозначно, що додаток-органайзер, що має в основі систему GTD є корисним і актуальним засобом для сьогоденної розробки і зможе доцільно використовуватися в бізнес-індустрії, сфері навчання та особистого ведення справ з мінімізацією зусиль на планування і максимізацією зусиль на виконання завдань.

Процес розробки безпосередньо спирається на план, який включає в себе такі складові як аналіз сервісів організації планування, виявлення їх недоліків та переваг та знаходження методів з мінімізації недоліків схожих сервісів та покращення або модифікація їх переваг.

Проектування та розробка інтерфейсної частини, що має високий вплив на майбутнє використання додатку так, як сервіс є різновидністю віджету, який має бути приємним та привабливим для користувача. Використовуватимуться обов'язково всі канони гармонійного сприйняття користувачем віджету такі, як поєднання кольорів, об'єктів та шрифтів. Інтерфейс обов'язково створюється інтуїтивним і зрозумілим, щоб користувач ні в якому разі не мав потреби витрачати час на навчання перед його застосуванням.

Після зробленого аналізу виявлена потреба у створенні наступних функцій, які будуть доступні при користування сервісом – органайзером:

– Створення записів – планів разом з датою їх виконання та обов'язковою можливістю вказати повноцінний опис, наслідуючи ідею відсутності в подальшому потреби від користувача розуміння та усвідомлення «що це за справа» та «для чого» вона.

– Створення нагадування, причому не тільки через даний додаток, а ще й можливість синхронізації з Google календарем, який в свою чергу також буде нагадувати про нагальну справу користувачу. Ця опція буде відразу запропоноване користувачу для впровадження її в користування додатком. Результатом є подвійне страхування від проблем, що зв'язані з забуванням завдань або їх невчасного виконання.

Обов'язковим є момент донесення ідеї GTD до користувача, який полягатиме в прямій її демонстрації при початку користування програмою у вигляді текстових правил та настанов. Також, наслідуючи принципи GTD, справи буде рекомендовано інтерфейсом вносити зважаючи на їх пріоритетність, тобто починаючи з найбільш серйозних та важливих і до найменш суттєвих. Таке рішення було прийняте для того, щоб кожен користувач, зосередившись заніс у список справ весь їх перелік, а не лише обрані, що на його думку є головними. Звичайно, додатково всі заплановані завдання будуть синхронізуватися за датою і зводитися до зручного для перегляду та редагування їх виду.

При розробці програмного продукту використовуються такі технології:

– Vue.js – за допомогою цього фреймворку розробляється клієнтська частина додатку, це додасть реактивності і швидкості роботи додатку.

– Node.js + Mongo.db – серверна частина додатку

– C# xamarin – мобільний додаток, який допоможе охопити більшу аудиторію користувачів.

Варто також зазначити, що додаток має бути доступним і на персональних комп'ютерах, і на мобільних гаджетах, що підтримують операційну систему Android, а також IOS, щоб планування було комфортним та доступним на будь-якому телефоні чи планшеті.

УДК 004.424

Докічук М.О., студ., гр. ПІК-14
Левківський В.Л., старший викладач каф. ІПЗ
Житомирський державний технологічний університет

АВТОМАТИЗОВАНА СИСТЕМА СЛУЖБИ ДОСТАВКИ

На сьогоднішній день автоматизовані системи служб доставки актуальні як ніколи, це пов’язано з розвитком інформаційних технологій та бізнесом в цілому. Так як шаленими темпами відбувається розвиток різних комерційних підприємств на кшталт інтернет-магазинів, домашньої розробки (handmade).

Основним призначенням кур'єрської служби є швидка і надійна доставка документів, друкованої продукції або інших товарів конкретному отримувачу. Належне збереження вантажу і швидкість його доставки тут грає ключову роль. Для вирішення даної проблеми розробляється автоматизована система служби доставки, яка дозволяє досягти максимальної ефективності та швидкості в послугах даної сфери діяльності. Автоматизована система надає користувачам зручно та легко взаємодіяти з посилками незалежно від їх статусу - отримувача чи відправника, а саме:

- відслідковування місця знаходження вантажу;
- спілкування з службою підтримки в разі виникнення питань;
- ознайомлення з детальною інформацією на сайті, такою як: новини, контакти, місце знаходження, час роботи служби доставки та ін.

Вся система представляє собою веб-сайт розроблений за допомогою фреймворку laravel на базі мови програмування PHP. Також використовуються такі технології як: HTML5, CSS3, PHP, MySQL, JavaScript, JQuery, Twitter Bootstrap, owl carousel, fancybox, та інші.

Система базується на основі шаблону проектування MVC (ModelViewController).

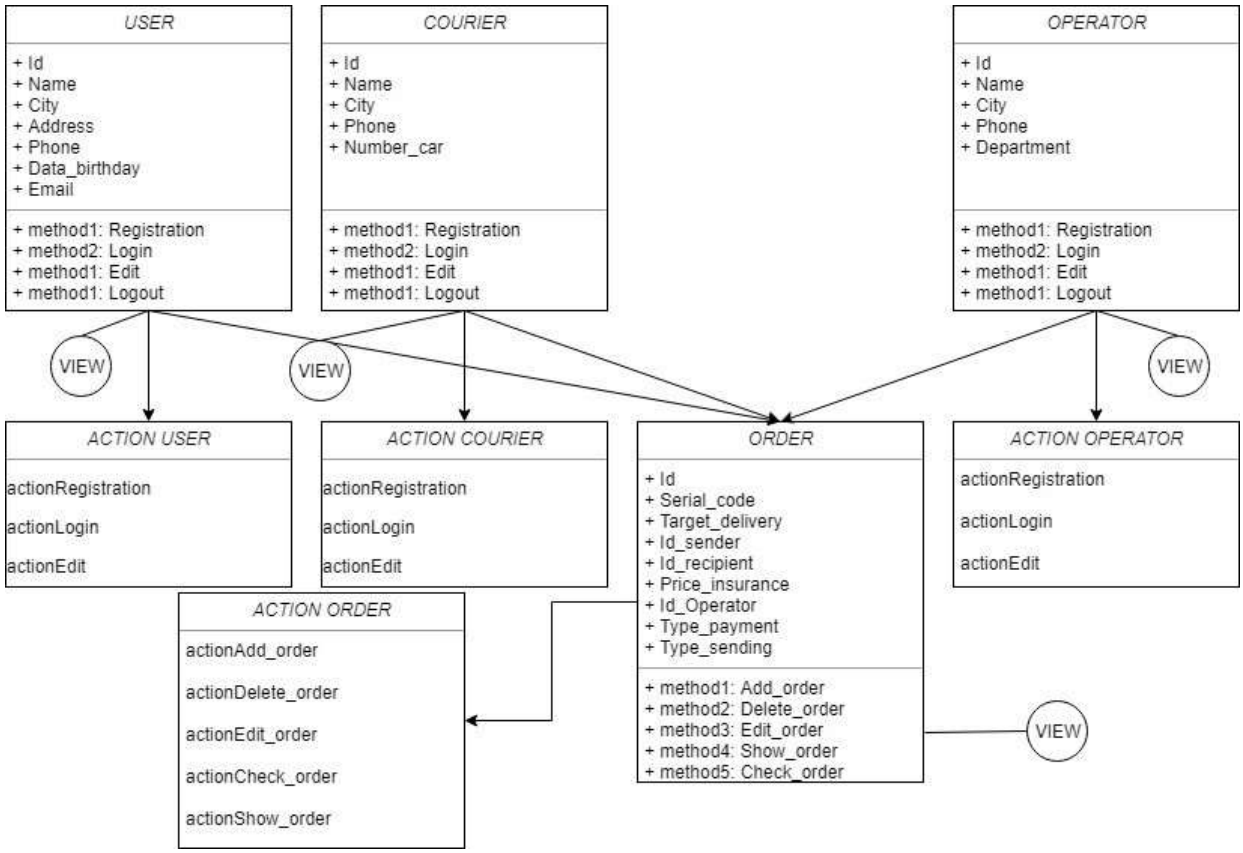

Рис. 1. MVC паттерн автоматизованої системи служби доставки товарів

На рисунку 1 зображено чотири контролера які призначені для різних типів користувачів, а саме для різних виконавчих функцій. В кожному з чотирьох контролерів описані їх методи за якими вони будуть працювати та поля для ідентифікації.

Також описані чотири дії для кожного контролера з інструкціями для нього, що буде виконуватись та яким чином відбувається взаємодія з контролером. В кожного контролера є своє графічне відображення користувачу для зручної роботи з системою в цілому та окремим функціоналом.

Таким чином, запропонована автоматизована система вносить зручність в користуванні системами доставки та контролюванні власних вантажів, таких як: цінні папери, грошові кошти, малогабаритні вантажі, та інші різноманітні відправлення.

УДК 004.42

В.М. Євдоченко, магістр, гр. ІПЗ-1
Грабар О.І., к.т.н., доц. каф. ІПЗ
Житомирський державний технологічний університет

ВПРОВАДЖЕННЯ СИСТЕМ ВІРТУАЛІЗАЦІЇ В ЯКОСТІ ВИКОРИСТАННЯ ЇХ ЯК СЕРВЕРНИХ ОС В ТЕРИТОРІАЛЬНОМУ ОРГАНІ ДФС

Implementation of virtualization systems in quality of using their series as to the territorial body of SFS

Усвідомлене розуміння, що ефективне функціонування існуючої в Україні системи оподаткування може бути досягнуте тільки на основі впровадження сучасних інформаційних технологій на всіх рівнях ієрархічної структури служби – є основою роботи протягом всього періоду існування підрозділу інформаційних технологій.

Велика кількість платників податків, постійний ріст вимог до комп'ютерного обладнання, тестування нових та допрацьованих версій старих програмних продуктів змушує використовувати сучасні технології віртуалізації в роботі серверного обладнання.

Мережеві служби, наприклад, DNS, DHCP, SMB у більшості випадків реалізуються на основі Windows Server 2000/2003/2008 – теж у більшості випадків без ліцензії.

Тому для підвищення надійності та масштабованості інформаційної системи органу ДФС, а також відмови від використання неліцензійного програмного забезпечення рекомендовано сервери переводити не вільну операційну систему Linux, на базі якої можна створити і сервер баз даних, і файловий сервер (використовується протокол SMB), і використовувати мережеві служби, наприклад DNS, DHCP.

При виборі технології впровадження віртуальної системи необхідно враховувати можливості апаратного забезпечення та з метою дотримання законодавства щодо використання комп'ютерних програм, визначення потреби в їх легалізації.

Перспективним є використання серверу віртуалізації Proxmox VE та реалізація серверу DHCP й файлового серверу за допомогою протоколу SMB на базі Linux, що є абсолютно безкоштовним та більш надійним ніж ОС Windows з реалізованими на її основі мережевими службами.

Проте, щоб перейти з Windows, де більшість маніпуляцій проводиться за допомогою графічного інтерфейсу та миші, на Linux - необхідно мати деякий досвід роботи з командним рядком Linux. Тобто одна з причин, з яких підрозділи інформаційних технологій не бажають переходити на Linux – це недостатній рівень знань щодо UNIX-подібних операційних систем. Для подолання цієї проблеми можна запропонувати дистанційні курси по адмініструванню Linux. Ще одна перешкода для впровадження систем віртуалізації – відсутність сучасного обладнання, яке дає ефект від використання подібних рішень. Це питання вирішується тільки ні рівні керівництва органу, тобто якщо керівник буде зацікавлений в безперебійній та надійній роботі інформаційної системи, то парк комп'ютерного обладнання буде постійно оновлюватися.

Ця технологія дозволяє суттєво економити грошові кошти на обслуговуванні, управлінні і адмініструванні інфраструктури серверів. Це відбувається завдяки розміщенню декількох віртуальних серверів на одному фізичному сервері. Це вкрай актуально в умовах недостатнього фінансування технічної бази в органах державної податкової служби.

При необхідності тестування нової версії програмного забезпечення, стару версію можна підтримувати на віртуальній машині, поки не будуть повністю вирішені можливі недоліки, які в свою чергу можуть привести до непрацездатності програмного засобу.

На одній хостовій системі може бути запущено одночасно кілька віртуальних машин, об'єднаних у віртуальну мережу. Ця особливість надає безмежні можливості по створенню моделей віртуальної мережі між декількома системами на одному фізичному комп'ютері. Особливо це необхідно, коли потрібно змоделювати якусь розподілену систему, що складається з декількох машин. Також можна створити декілька ізольованих користувальницьких оточень, запустити їх і перемикатися між ними в міру необхідності виконання тих чи інших завдань.

Віртуальність підвищує мобільність. Папка з віртуальною машиною може бути переміщена на інший комп'ютер, і відразу встановлена. Не потрібно створювати ніяких образів для міграції, і, до того ж, віртуальна машина відв'язані від конкретної апаратури. При використанні віртуальних машин істотно підвищується керованість щодо створення резервних копій, створення знімків віртуальних машин («снапшотів») і відновлень після збоїв. Все це зменшує витрати ресурсів, часу, коштів та підвищує ефективність роботи.

УДК 004.942

Р.І. Заблодський, магістр, гр. ПІ-47м
С.М. Кравченко, старший викладач каф. ІПЗ
Житомирський державний технологічний університет

РОЗПІЗНАВАННЯ ЖЕСТІВ ЗА ДОПОМОГОЮ МЕТОДУ ВІОЛИ-ДЖОНСА

В процесі розвитку інформаційних технологій створюються нові методи та алгоритми людино- машинної взаємодії. Одним з таких напрямів розвитку являється розпізнавання мови жестів людини, що розширює взаємодію машини та людей, з обмеженими можливостями. «Захоплення» жестів руки може бути здійсненим за допомогою різноманітних приладів, але найбільш поширеним способом отримання даних положення руки та здійснених жестів являється відеокамера.

В основному, процес розпізнавання жестів відбувається за наступною схемою:

1) Попереднє налаштування;
2) Знаходження руки на відеопослідовності;
3) Визначення параметрів жесту;
4) Розпізнавання жесту.

Реалізація комп'ютерного зору може бути здійснена з використанням бібліотеки OpenCV (Open Source Computer Vision Library). Бібліотека надає засоби для обробки і аналізу вмісту зображень, у тому числі розпізнавання об'єктів на фотографіях (наприклад, осіб і фігур людей, тексту тощо), відстежування руху об'єктів, перетворення зображень, застосування методів машинного навчання і виявлення загальних елементів на різних зображеннях.

Для розпізнавання жестів на відеопослідовності часто застосовується метод Віоли-Джонса. Навчання класифікаторів йде дуже повільно, але результати пошуку образів дуже швидкі, саме тому даний метод часто застосовується для пошуку образів в реальному часі. Віола-Джонс є одним з кращих по співвідношенню показників ефективності розпізнавання / швидкості роботи. Також цей детектор має вкрай низьку ймовірністю помилкового виявлення образу. Алгоритм добре працює і розпізнає образи під кутом, приблизно до 30 градусів. Але при куті нахилу більше 30 градусів відсоток виявлення різко падає.

Алгоритм має чотири етапи:

1) Вибір ознаки Хаара;
2) Створення інтегрального зображення;
3) Машинне навчання з використанням алгоритму AdaBoost (adaptive boosting);
4) Каскадні класифікатори.

Ознаки, використовувані алгоритмом, використовують підсумовування пікселів з прямокутних регіонів. Ознака Хаара складається з суміжних прямокутних областей (рис.1). Вони позиціонуються на зображенні, далі підсумовується інтенсивність пікселів в областях, після чого обчислюється різниця між сумами. Ця різниця і буде значенням певної ознаки, визначеного розміру, певним чином позиціонованого на зображенні. Такий вид ознак називається 2-прямокутним. Віола і Джонс так само визначили 3-прямокутні і 4-прямокутні ознаки.

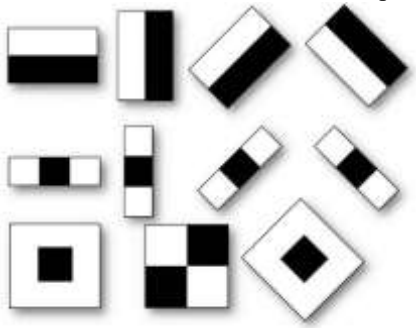

Рис. 1. Ознаки Хаара

Кожна ознака може показати наявність (або відсутність) будь-якої конкретної характеристики зображення, такий як межа або зміна текстур. Наприклад, 2-прямокутна ознака може показати, де знаходиться межа між темним і світлим регіонами.

Отже, алгоритм AdaBoost має багато переваг: хороша узагальнююча здатність. В реальних задачах вдається будувати композиції, що перевершують за якістю базові алгоритми. Узагальнююча здатність може поліпшуватися у міру збільшення числа базових алгоритмів; простота реалізації; власні накладні витрати бустингу невеликі. Час побудови композиції практично повністю визначається часом навчання базових алгоритмів; можливість ідентифікувати об'єкти, які є шумовими викидами.

УДК 004.424

Каліберда С.С., студ., гр. ПІК-14
Марчук Г.В., старший викладач каф. ІПЗ
Житомирський державний технологічний університет

РЕАКТИВНЕ ПРОГРАМУВАННЯ (REACTIVE PROGRAMMING)

Реактивне програмування - це програмування з асинхронними потоками (streams) даних. Шини подій або типові події натискань на кнопки - це все реальні приклади асинхронних потоків подій. По суті - реактивність означає те, що одні події можуть виконатися тільки після інших подій. Розробникам надається можливість створювати потоки чого-небудь, а не тільки події натискань. Потоки легкі і використовуються усюди: змінні, властивості, кеш, структури даних і багато іншого. Наприклад, стрічка Facebook може бути потоком даних нарівні з низкою подій користувальницького інтерфейсу. Тобто можна слухати потік і реагувати на події в ньому.

Більш того, програміст отримуєте чудовий набір інструментів і функцій для поєднання, які дозволяють створювати і фільтрувати кожен з цих потоків. Ось де функціональна парадигма дає про себе знати. Потоки можуть бути використані як вхідні параметри один одного. Навіть множинний потік може бути використаний як вхідний аргумент іншого потоку. Програміст можете об'єднувати кілька потоків або можете фільтрувати один потік, щоб потім отримати інший, який містить тільки актуальні дані, або можете об'єднувати дані з одного потоку з даними іншого, щоб отримати ще один.

Якщо потоки - це центральна ідея реактивності, то більш детально розглянемо їх на прикладі користувальницького кліка мишею, що зображено на рис.1.

Рис. 1. Діаграма потоку подій

Потік – це послідовність подій, упорядкована за часом. Він може викидати три типи даних: значення (певного типу), помилку або сигнал завершення. Сигнал завершення поширюється, коли поточне вікно або вікно, що містить кнопку, закривається.

Ми отримуємо події, що згенеровані асинхронно, завжди. Згідно з ідеологією реактивного програмування існують три види функцій: ті, які повинні виконуватися, коли деякі конкретні дані будуть відправлені, функції обробки помилок і інші функції з сигналами про завершення роботи програми. Іноді останні два пункти можна опустити і зосередитися на визначенні функцій для обробки значень. Слухати (listening) потік означає підписатися (subscribing) на нього. Тобто функції, які ми визначили це спостерігачі (observers), а потік є суб'єктом який спостерігають. Такий підхід називається Observer Design Pattern.

Альтернативним і досить популярним способом представити вищезгадану діаграму, що зображена на рис. 1 є ASCII графіка:

```
--a---b-c---d---X---|->
a, b, c, d are emitted values          // значення що генеруються X is an error      // помилка
| is the 'completed' signal            // сигнал завершення
---> is the timeline                   // тимчасова ось
```

Розберемо приклад. Згенеруємо нові потоки повідомлень кліка, трансформовані з оригінального потоку. Для початку зробимо потік лічильників, який визначає, скільки разів кнопка була натиснута. У більшості реактивних бібліотек у кожного потоку є безліч вбудованих функцій, таких як map, filter, scan і т.п. Коли ви викликаєте одну з цих функцій, наприклад clickStream.map (f), вона повертає новий потік, заснований на батьківському. Функції не модифікують батьківський потік. Це називається незмінюваність і є невід'ємною частиною реактивного підходу, дозволяючи нам викликати ланцюжок функцій, наприклад clickStream.map (f) .scan (g):

```
clickStream: ---c----c--c----c------c--> vvvvv map(c becomes 1) vvvv
---1----1--1----1------1-->
vvvvvvvvv scan(+) vvvvvvvvv counterStream: ---1----2--3----4------5-->
```

Функція map(f) створює новий потік, в якому за допомогою функції f замінюватися кожна нова подія. В нашому випадку прив’язується одиниця до кожного натискання на кнопку. Функція scan(g) агрегує всі попередні значення в потоці, повертаючи значення x = g (accumulated, current), де g в даному випадку - це просто функція складання. Після цього counterStream посилає загальну кількість натискань.

Давайте припустимо, що ми хочемо реалізувати потік подій «подвійний клік». Щоб зробити цю задачу ще цікавіше новий потік повинен приймати множинні натискання за подвійні. Кращий спосіб навчитися розуміти і проектувати потоки- це зображати діаграми (рис. 2).

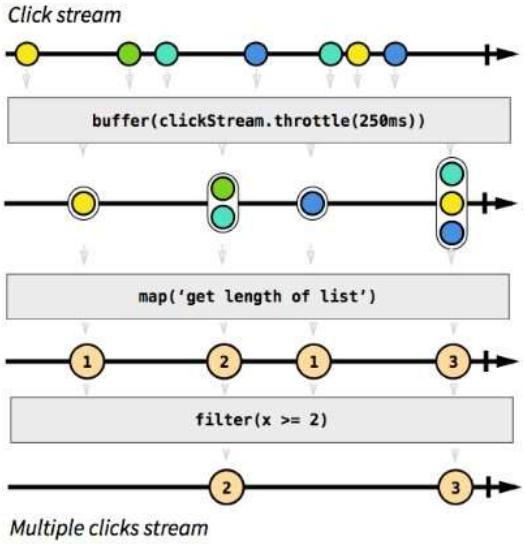

Рис. 2. Діаграма потоку подій

Сірі прямокутники – це функції, які трансформують один потік в інший. Спочатку ми збираємо кліки в списки. Якщо минуло 250 мілісекунд без єдиного натискання кнопки - ми застосовуємо функцію map () на кожному зі списків, щоб обчислити його довжину. В кінці ми фільтруємо списки з довжиною 1, використовуючи функцію filter (x> = 2). Ось так, в три дії, можна отримати результат - потік подій множинних кліків.

Цей приклад показує всю простоту, з якою реалізовується досить складна на перший погляд задача, якщо ми використовуємо реактивний підхід.

Основні принципи функціонального реактивного програмування:

- Динамічні/еволюціонуючи значення (тобто значення "за часом") є класами. Розробник можете визначити їх і поєднувати їх, передати їх у функції. Загалом це можна назвати "поведінкою".
- Поведінка складається з декількох примітивів, таких як постійний (статичний) стан та час (наприклад, годинник), а потім послідовне та паралельне поєднання. n поведінок поєднуються, застосовуючи n-арію функцію (на статичні значення), "точка-точка", тобто постійно з часом.
- Після виконання деяких операцій над даними створюється новий потік, а висхідний не змінюється, це називається незмінність (Immutable). Це дозволяє одні і ті ж дані обробити декільком обробникам.
- Для обліку дискретних явищ є інший тип (сім'я) "подій", кожний з яких має потік (кінцевий або нескінченний) подій. Кожна подія має пов'язаний час і значення.
- Щоб придумати композиційний словник, з якого можна побудувати всі способи поведінки та події, потрібно експериментувати з деякими прикладами. Постійно робити декомпозицію загальних частин на прості частини.
- Реактивне програмування засноване на двох шаблонах проектування: Спостерігач (Observer) та Ітератор.
- Сучасні додатки представляють не просто набір послідовних команд, які послідовно виконуються, а додатки які повинні реагувати на якісь зміни (натискання на клавішу, зміна стану моделі, запис до бази даних).

Для чого потрібно реактивне програмування?

Реактивний підхід підвищує рівень абстракції вашого коду і ви можете сконцентруватися на взаємозв'язку подій, які визначають бізнес-логіку, замість того, щоб постійно підтримувати код з великою кількістю деталей реалізації. Код в реактивному програмуванні, ймовірно, буде коротшим.

Перевага більш помітно в сучасних веб-і мобільних додатках, які працюють з великою кількістю різноманітних UI-подій.

Реактивне програмування дуже добре підходить для обробки великої кількості різноманітних подій.

УДК 004.9

Н.Г. Корчмар, студ., гр. ПІ-49
Науковий керівник – к.т.н., доц. каф.ІПЗ Сугоняк І.І.
Житомирський державний технологічний університет

ПРОБЛЕМА РЕФАКТОРІНГУ (REFACTORING) В ПРОЦЕСІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Проблема якісного написання коду постає перед розробниками досить часто, адже саме код, що пишеться з урахуванням патернів та особливих технік, свідчить про компетентність розробника.

Для виявлення проблеми застосовують аналіз коду, що часто застосовується до таких гнучких практик розробки як попарне програмування, CI, TDD.

Задля покращення якості написання коду спеціалісти сфери розробки програмного забезпечення (ПЗ) залучають до своєї роботи рефакторінг коду на всіх етапах розробки продукту. Передбачуваним результатом проведеного рефакторінгу є реалізація більш швидкої в роботі та покращеної версії ПЗ.

Процес рефакторінгу (від англ. *refactoring*) − перепроектування коду, процес зміни внутрішньої структури програми, але без зміни зовнішньої поведінки.

Обгрунтування скористатись рефакторінгом:

- Код повторюється;
- код містить глобальні змінні;
- метод занадто великий;
- метод приймає занадто багато параметрів;
- метод має невдале ім’я;
- цикл занадто великий або занадто глибоко вкладений в інші цикли;
- при зміненні програми потрібно паралельно змінювати декілька класів;
- підклас використовує тільки малу долю методів батьківських класів;
- окремі частини класу видозмінюються незалежно від інших;
- клас має погану зв’язаність; (якщо клас має багато ніяк не пов’язаних аспектів – розбийте його на декілька класів так, щоб отримати зв’язний набір аспектів);
- складний код пояснюється за допомогою коментарів.

Використання методик «чистого» чи «досконалого» коду (clean code) допомагають проектувати з мінімальною складністю і максимальною продуктивністю. Найбільшої популярності зазнала праця Роберта Мартіна «Чистий код. Створення, аналіз і рефакторінг», яка є скарбницею цінних правил та порад.

Приступаючи до рефакторінгу побудуйте надійний набір тестів для обробки частини коду. Тести потрібні для того, щоб, навіть послідовно виконуючи рефакторінг, виключити появу помилок.

Імплементуйте рефакторінг при:

- Створенні нових методів та класів;
- виправленні дефектів;
- наявності модулів, які містили помилки;
- наявності складних модулів.

Рефакторінг – ефективний спосіб підвищення якості коду. Але не забувайте і про наступне:

- Зберігайте початковий код.
- Намагайтесь обмежити об’єм окремих видів рефакторінгу (деякі більш масштабними, ніж інші. Тому виконуйте всі види рефакторінгу по одному разу, компілюючи та тестуючи програму перед наступним видом рефакторінгу).
- Створіть список дій, які намагаєтесь застосувати.
- Створіть та підтримуйте список видів рефакторінгу, які маєте застосувати пізніше.
- Часто створюйте контрольні точки.
- Виконуйте регресивне тестування.
- Створюйте додаткові блочні тести для перевірки нового коду. Застарілі – видаляйте.
- Виконуйте огляд змін.

Рефакторінг під час розробки – найліпша можливість покращення програми та внесення в неї всіх змін, які вам захочеться внести пізніше. Користайтеся цією можливістю.

Сьогодні існує широкий вибір інструментів для проведення перевірки коду, серед них Crucible (Atlassian), Gerrit (Shawn Pearce), Upsource (JetBrains), GitLab, Review Board, Collaborator (SmartBear Software), Codacy (Codacy), Malevich. Дані продукти мають широку спільноту користувачів, завдяки якій створена значна кількість плагінів та розширень, що дозволяють вирішити майже будь-яку проблему.

При вмілому імплементуванні вищезазначених практик процес налагоджується. Це значно полегшує подальшу роботу, зменшує кількість помилок та прискорює процес їхнього усунення, що ефективно економить ресурси.

УДК 004.42

В.І.Лажевський, магістр, гр. ПІ-48м
О.І. Грабар, к.т.н., доц. каф.ІПЗ
Житомирський державний технологічний університет

СИСТЕМА ВІЗУАЛЬНОГО РОЗПІЗНАВАННЯ СОРТІВ РОСЛИН

The main objective of the thesis is to create a web service for plant recognition by image.

Основна мета проекту полягає у створенні веб-сервісу для розпізнавання рослин за зображенням. Високоякісне розпізнавання зображень рослин – складне завдання для комп'ютера через різноманітний вигляд і складну структуру рослин. Ми розглядаємо стан сучасної задачі розпізнавання рослин, від ідентифікації рослин, конкретних органів рослини до загального розпізнавання рослин "в дикій природі".

З точки зору машинного навчання, розпізнавання рослин є дрібнозернистим класифікаційним завданням з високою мінливістю між класами та часто незначними відмінностями між класами, які часто пов'язані з таксономічною ієрархічною класифікацією. Інтерес до методів візуальної класифікації рослин виріс в останній час через зростання кількості пристроїв, які оснащені камерами. Саме ж розпізнавання рослин було поставлено, майже без винятків, як розпізнавання фотографій, що зображують виключно певний організм рослини, такі як квітка, кора, фрукти, лист або їх комбінація. Розпізнавання листів стало найпопулярнішим підходом до розпізнавання рослин, і в літературі було зазначено широкий спектр можливостей. Визнання листя зазвичай стосується лише визнання широких листів, голки розглядаються окремо. Для опису листів було запропоновано декілька прийомів, які часто грунтуються на об'єднаних рисах різного характеру (особливості форми, особливості кольорів тощо).

Одним з хороших алгоритмів для розпізнавання листів є SIFT (Scale-invariant feature transform), методів-геометричних особливостей, моментних інваріантів, моментів зерніке та полярних перетворень Фур'є-останнє найкраще виконується на неопублікованому наборі даних.

Pl@ntNet – це система розпізнавання рослин на основі вмісту. Це спільна інформаційна система, що забезпечує програму для обміну фотографіями та пошуку для ідентифікації установок. Вона була розроблена вченими чотирьох французьких дослідницьких організацій (Cirad, INRA, INRIA та IRD) та мережі Tela Botanica. База даних дерева дерева ідентифікується шляхом поєднання інформації з зображень середовища проживання, квітка, фруктів, листя та кору. Точні алгоритми, що використовуються в веб-службі визначення Pl@ntNet та їх точність не публічно задокументовані.

Текстурна інформація є важливою ознакою для розпізнавання багатьох органів рослин. Текстурний аналіз є загальною проблемою з великою кількістю існуючих методів. Саму текстуру важко визначити. Є різні визначення візуальної текстури, але вони часто не мають формальності та повноти. Поняття текстури, здається, залежить від трьох інгредієнтів: (1) місцевий «порядок» повторюється над регіоном, який є великим у порівнянні з розміром замовлення, (2) порядок полягає у невипадковому розташуванні елементарних частин, і (3) частини є приблизно однорідними об'єктами, що мають приблизно однакові розміри всюди в текстурованому регіоні. Кілька недавніх підходів до розпізнавання текстури відрізняються відмінними результатами в стандартних наборах даних, багато з яких працюють лише з інтенсивністю зображення та ігнорують доступну інформацію про кольори.

Для того, щоб описати текстуру незалежно від розміру візерунка та орієнтації на зображенні, необхідний опис, інваріантний для обертання та масштабу. Для практичного застосування нам також потрібне ефективне обчислення. Нижче приведемо один з методів для розпізнавання текстур.

Завершено локальне подвійне зображення та гістограма Фур'є. Перше опис описується на основі локальних двійкових шаблонів (LBP). Загальний оператор LBP локально обчислює ознаки відмінностей між центральним пікселем і його *P* сусідів по колу радіуса *R*. З функцією зображення $f(x, y)$ та точками координат точки (x_p, y_p):

$$\mathrm{LBP}_{P,R}(x, y) = \sum^{P-1} s\,(f(x, y) - f(x_p, y_p))\,2^P,\ s(z) = \begin{cases} 1: if\ z \le 0 \\ 0: otherwise \end{cases} \quad (1)$$

Для досягнення інваріантності обертання ми приймаємо так звані Фур'є-функції гістограми LBP (LBP-HF). LBP-HF описують гістограму рівномірних візерунків, використовуючи коефіцієнти дискретного перетворення Фур'є (DFT). Уніфіковані LBP - це шаблони з максимум 2 просторовими переходами (побітові 0-1 зміни). На відміну від простих інваріантів обертання з використанням LBP^{ri}, який об'єднує всі однорідні візерунки з таким самим числом 1s в один контейнер, функції LBP-HF зберігають інформацію про відносне обертання шаблонів.

Позначення рівномірної картини $U^{n,r}$,рде *n* – це число «орбіти», що відповідає кількості бітів «1», а *r* – обертання шаблону, то DFT для заданого *n* виражається як:

$$H(n, u) = \sum_{r=0}^{p-1} h_1\left(U_p^{n,r}\right) e^{-\frac{i2\pi ur}{p}}, \quad (2)$$

Де значення гістограми $h_I\left(U_p^{n,r}\right)$ позначає кількість входжень даного ріномірного малюнка на зображенні. Функції LBP-HF дорівнюють абсолютному значенню величин DFT, і тому на них не впливає фазовий зсув, викликаний поворотом.

$$LBP - HF(n,u) = |H(n,u)| == \sqrt{H(n,u)\overline{H(n,u)}}. \quad (3)$$

Так h_I реальні, $H(n,y) = H(n,P-y)$ для $y = (1, ..., P-1)$, і тому тільки $\left[\frac{P}{2}\right] + 1$ від величини DFT використовуються для кожного набору рівномірних візерунків з n "1" біт при $0 < n < P$. Три резус-представлення додасться до трьох інших контейнерів, а саме: два для "1-рівномірних" візерунків (з усіма контейнерами одного значення) та одна для всіх неоднорідних візерунків.

Функції Фур'є гістограми LBP можна узагальнити на будь-який набір рівномірних візерунків. У першому випадку використовується опис LBP-HF-SM, де для побудови дескриптора розраховуються функції Фур'є гістограми як знаків, так і величини-LBP. Величина-LBP перевіряє, чи величина різниці сусіднього пікселя (x_p, y_p) відносно центрального пікселя (x, y) перевищує порогове значення t_p:

$$LBP - M_{P,R}(x,y) = \sum_{p=0}^{P-1} s\left(\left|f(x,y) - f(x_p,y_p)\right| - t_p\right)2^p. \quad (4)$$

Ми прийняли загальну практику вибору порогового значення (для сусідів на p біт) як середнє значення всіх абсолютних різниць m у всьому зображенні:

$$t_p = \sum_{i=1}^{m} \frac{|f(x,y) - f(x_{ip},y_{ip})|}{m}. \quad (5)$$

Гістограма LBP-HF-SM створюється шляхом об'єднання гістограм LBP-HF-S та LBP-HF-M (обчислено з рівномірного знаку-LBP та величини-LBP).

Для додавання інваріантів обертання. Функції LBP-HF, які використовуються в запропонованому першому описі, зазвичай будуються з DFT величини різнорідних рівномірних візерунків. Можна використовувати всі LBP, а не тільки підмножину уніфікованих візерунків. Зверніть увагу, що в цьому випадку деякі орбіти мають меншу кількість моделей, оскільки деякі неоднорідні шаблони демонструють симетрії, як це показано на рис. 1.

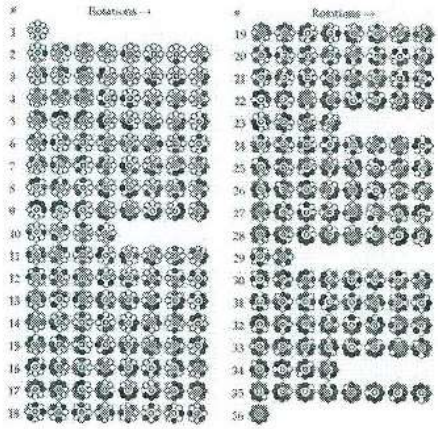

Рис.1. Повний набір локальних шаблонів двійкового розподілу на 36 орбіт для функцій Фур'є гістограми.

$$LBP - HF^{+} = \sqrt{H(n,1)\overline{H(n+1,1)}} \quad (6)$$

$^{+}$ позначає метод, що використовує повний набір моделей для функцій LBP-HF та додавання додаткових функцій LBP- HF $^{+}$.

У глибинних навчальних завданнях є поширеною практикою навчання декількох мереж на різних (але не обов'язково взаємовиключних) підмножинах навчальних даних. Ансамбль таких мереж, зазвичай поєднуються за допомогою простого механізму голосування (наприклад, сума або максимальна кількість класів), прагне перевершити індивідуальні мережі. Визначення видів рослин з фотографій з використанням текстурного визнання за допомогою сучасних методів дає досягти доволі хороших результатів, при цьому зберігаючи обчислювальні вимоги невеликими, що робить його придатним для обробки в реальному часі.

УДК 004.42

С.В. Лип'янець, магістр, гр. ІПЗ-1
Єфремов Ю.М., к.т.н., доц.
Житомирський державний технологічний університет

ОСНОВНІ ТЕНДЕНЦІЇ РОЗРОБКИ ЯКІСНОГО ВЕБ-ДИЗАЙНУ ЯК ВАЖЛИВОЇ СКЛАДОВОЇ ПРИ СТВОРЕННІ СУЧАСНОГО САЙТУ

Main trends for developing a quality web design as an important component in creating a modern site

Розвиток інтернет – технологій призвів до появи нової гілки в Інтернеті. Сайти почали реєструватися під тисячами доменів, з'явилися цілі портали, де люди з усього світу можуть спілкуватися, отримати відповіді на будь-які питання, здійснювати різноманітні операції. В наш час важко уявити організацію, успішний бізнес, що не мають виходу в мережу і своєї web-сторінки. Цей процес стрімко розвивається, використання Інтернету набагато збільшує інформативні можливості людини.

Разом із тим на перший план виходить питання раціональної розробки та підтримки окремо взятого веб-ресурсу, адже зацікавлення потенційного користувача у певній інформації є однією із пріоритетних цілей власників таких сайтів. Як показує практика, користувачеві достатньо лише відкрити одну із сторінок, щоб скласти ту чи іншу думку про сайт, на який він потрапив. Ну і само собою, якщо на людину це не справило приємного враження, швидше за все вона піде в пошуках іншого ресурсу.

Саме тому при розробці сайту особливу увагу потрібно приділити дизайну. Він в першу чергу покликаний вирішувати вищевказану проблему. Основною метою розробника є змусити людину затримати погляд і залишитися на сторінці сайту для подальшого дослідження, а згодом й пошуку конкретної інформації.

На сьогоднішній день існує велика кількість різних тенденцій та ідей, яких потрібно дотримуватися при розробці власного веб-ресурсу.

Однією із основних таких тенденцій є можливість розміщення всієї важливої інформації на одній сторінці. Користувач зможе легко оглянути її та прогорнути. Переважна більшість людей зацікавлена в максимально спрощеному і скороченому варіанті поданої інформації, а у випадку з використанням мобільних телефонів та планшетів це буде зробити зручніше, не застосовуючи при цьому кліки мишкою та переходячи між сторінками сайту.

Яскравий і насичений колір вивів дизайн на новий рівень. Якщо раніше розробники дотримувалися спокійних кольорових гам, то тепер все частіше і частіше зустрічається розмаїття відтінків та колірних палітр. Досить простим інструментом, який дає прекрасний результат є градієнт. Фон сторінки, виконаний в градієнті, створює враження свіжості і унікальності.

Важливим і водночас незамінним елементом будь-якого сайту є типографія. Сьогодні - це великі сміливі шрифти, які дозволяють задати тон і настрій ресурсу. Вони безпосередньо впливають на сприйняття інформації, і можуть, як привернути увагу користувача до сайту, так і змусити його закрити вкладку. Застосування унікальних шрифтів здатне підвищити загальний вигляд сторінки, навіть якщо її дизайн буде досить скромним.

Правильно налаштована анімація допоможе спростити процес взаємодії користувачів із сайтом та полегшить сприйняття інформації, а наявність інтерактивних елементів робить веб- ресурс більш зручним та привабливим. Ще більше уваги привертає використання відео, живих фотографій і фону, що надає сайту певної динаміки.

А чого лише вартує підключення так званих сінемаграфів, новинки у сфері веб-дизайну? Це сучасні статичні ілюстрації з частковою анімацією. Сінемаграфи швидко набувають популярності серед веб-розробників та додають пікантності зовнішньому вигляду веб-сторінки. Вони можуть повністю розкрити суть самого питання чи пропозиції, яку несе в собі проект.

Сучасні тенденції розробки сайтів такі, що всі ресурси прагнуть якомога більше спростити свою структуру і зробити її одноманітною. А тому доцільним є впровадження особливого контенту, який буде персоналізований, мати нестандартне розташування елементів та виділятися з поміж інших. Правильне ж його застосування дозволить повноцінно реалізувати дизайн та привернути увагу користувачів.

УДК 004.9

Н.В. Лазорко, студ., гр. ПІ-49
Науковий керівник – старший викладач кафедри ІПЗ, Скачков В.О.
Житомирський державний технологічний університет

ПЕРЕХІД ВІД МОВИ JAVA ДО KOTLIN. ПЕРЕВАГИ ТА НЕДОЛІКИ

Мова Java на сьогоднішній день є старою, але й досі однією з найпопулярніших мов програмування, до якої всі звикли. Вона використовується в багатьох напрямках програмування, мобільна розробка не є тому виключенням. Але вже в 2011 році компанією JetBrains було представлено нову мову програмування Kotlin, яка, на мою думку, стала головним конкурентом Java.

Kotlin – статично типізована мова програмування, що працює поверх JVM. Вона підтримує ООП та процедурне програмування. Автори ставили перед собою ціль створити лаконічнішу та типо-безпечнішу мову, ніж Java, і простішу, ніж Scala. Наслідками спрощення, порівняно з Scala стали також швидша компіляція та краща підтримка IDE.

Чи можливий остаточний перехід та відмова від Java? Розглянемо спершу переваги мови Kotlin:

- Сумісність з мовою Java на 100 %;
- null-safety (по замовчуванню типи не можутьбути null, що вирішує проблему NullPointerException);
- підтримка конвертації Java на мову Kotlin (починаючи з Android Studio 3.0);
- компактність та краща читабельність коду;
- поява класів даних (Data Classes), в яких містяться різноманітні шаблони: equals(), hashCode(), toString(), get, set та ін.;
- «розумне» приведення типів, вбудоване в компілятор;
- підтримка функціонального програмування (використання функцій вищого рівня, анонімні функції, лямбда-вирази);
- Extension Functions (можливість розширювати вже існуючі класи, додаючи в них власні методи);
- Type-Safe Builders (для роботи з HTML, XML, Anko);
- ви хочете вивчити щось нове і цікаве, але при цьому бажано, щоб витрачені зусилля були того варті.

Щодо недоліків слід зазначити:

- Молода мова програмування, що тільки починає розвиватися та вдосконалюватися;
- наявність помилок при конвертації з мови Java;
- недоліки в середовищі розробки IDE (Exceptions, Debugging);
- видалення тернарного оператора (?:).

Не дивлячись на те, що мова ще потребує багатьох допрацювань та виправлень, все ж вона пропонує розробникам настільки багато нових можливостей. Ми можемо покладатися на використовувані інструменти, і відчувати впевненість при роботі з ними. Компілятор не робить нічого зайвого або ризикованого. Він робить тільки те, що в Java нам потрібно робити вручну, економлячи нам час і ресурси. В якійсь мірі це приносить користь і кінцевим користувачам, тому що завдяки більш суворої типобезпеки ми залишимо менше багів в додатках.

Крім того, компілятор Kotlin постійно поліпшується, так код, що генерується стає все ефективніше.

Ще одним великим плюсом щодо мови Kotlin є його схожість до мови Swift, яка використовується для розробки мобільних додатків під платформу IOS. Тобто програміст може без великих зусиль спробувати себе в програмуванні на обох платформах та вибрати, яка йому до вподоби. Йому не потрібно освоювати дві різні мови та йти лише в одному напрямку.

Щодо швидкості роботи можна сказати наступне: перша збірка Kotlin-коду займає приблизно на 15–20 % більше часу, ніж аналогічний процес на Java. Однак інкрементна збірка Kotlin навіть трохи швидше, ніж у Java. Таким чином, мови приблизно рівні за швидкістю компіляції.

Як висновок, можна сказати, що Kotlin – це продовження Java, новий його етап у розвитку. Ідея створення зародилася з метою отримання компактного коду, який буде легко сприйматися та економити час. Kotlin є повністю сумісним із мовою Java, тому проекти легко можна конвертувати, а програмісти, які багато часу використовували тільки Java, зможуть і надалі мати всі бібліотеки та фреймворки, до яких звикли, бо в основному, для них зміниться тільки синтаксис. Всі ресурси, вихідні файли, допоміжні документи та файли наявні у відкритому доступі. Тому можна легко почати вивчати та освоювати цю нову мову, бо в майбутньому вона може значно знадобитися та широко використовуватися як одна з офіційних мов програмування для Android.

УДК 004.42

Ю.О. Магеррамова, магістр, гр., ПІ-48м
О.І. Грабар, к.т.н., доц., каф.ІПЗ
Житомирський державний технологічний університет

СИСТЕМИ АВТОМАТИЗАЦІЇ ДЛЯ РОЗШИРЕННЯ МОЖЛИВОСТЕЙ УПРАВЛІННЯ РЕСТОРАННИМ БІЗНЕСОМ

У сучасному світі у зв'язку зі стрімким збільшенням кількості ресторанів значно посилюється конкуренція, що призводить до необхідності пошуку нових шляхів покращення розвитку ресторанного бізнесу. Одним з варіантів вирішення даної проблеми є автоматизація ресторанів.

Автоматизація ресторану – це процес впровадження програмно-апаратних комплексів автоматизації бізнес-процесів на підприємстві громадського харчування. Система автоматизації ресторану - це багатофункціональна система управління закладом. Переваги використання комплексу програмного забезпечення для автоматизації ресторанів заключаються в наступному: система дозволяє завжди мати доступ до поточної інформації про роботу ресторану; система автоматизації ресторану допомагає виключити трудомісткі операції з обліку, планувати банкети та корпоративи, персоналізувати роботу з клієнтами, вести облік бронювання столиків; з'являється можливість мати безперервний моніторинг роботи всіх складових ресторану, аналізувати і прогнозувати результати діяльності закладу.

Основними задачами програмного комплексу для автоматизації ресторану є наступні:

- підвищення прибутковості і зниження витрат ресторану;
- аналіз та оптимізація запасів ресторану;
- збільшення продуктивності праці персоналу;
- покращення якості обслуговування відвідувачів ресторану;
- створення систем лояльності;
- аналіз та планування подальшого розвитку бізнесу.

Передбачувані результати роботи системи автоматизації ресторану: оптимізація запасів ресторану; оптимізація бухгалтерського обліку; зниження кількості помилок персоналу; оптимізація умов і якості всіх виконуваних робіт; збільшення прибутку.

Також актуальним для закладів громадського харчування є завдання автоматизованого створення актуальної бази даних (Рис. 1). База даних повинна бути взаємозалежною з терміналами офіціантів та бармена-касира, робочим місцем адміністратора, бухгалтера та друкуванням замовлень на кухні. Необхідно проаналізувати попередні запити клієнтів для замовлення продуктів та формувати меню для оптимального управління запасами ресторану. Це можливо при роботі з базою даних за технологічними картами ресторану і базою даних, що містить статистичний масив.

Рис. 1. Залежності бази данних

Основою системи автоматизації ресторану є механізм формування звітів. Звіти допомагають контролювати запаси ресторану та витрати, збирати маркетингову інформацію. Результатом створення програмного комплексу системи автоматизації ресторану та основною перевагою є скорочення надмірності збережених даних, збільшення ступеня достовірності і збільшення швидкості обробки інформації. Система реалізує реєстрацію подій, наприклад оформлення і моніторинг виконання замовлень, замовлення і витрата продуктів і т. д. За допомогою автоматизації цих процесів вся інформація буде зберігатися в одній базі, дані в яку будуть записуватися за допомогою зручного інтерфейсу. Отже, для співробітників ресторану та власників бізнесу, система автоматизації надає безліч можливостей, які полегшують робочий процес, змінюють та спрощують стиль роботи. Система автоматизації значно розширює можливості управління ресторанним бізнесом.

УДК 519.83

Марчук Д.К., магістр, група ПІ-47м
Марчук Г.В., старший викладач каф.ІПЗ
Житомирський державний технологічний університет

СТРАТЕГІЯ ПРИЙНЯТТЯ РІШЕНЬ НА ПРИКЛАДІ ГРИ ХРЕСТИКИ-НУЛИКИ

Якщо люди відмовляються вірити в простоту математики, то це тільки тому, що вони не розуміють всю складність життя.

Джон фон Нейман Теорія ігор (game theory) – розділ сучасної математики, що вивчає математичні моделі так званих конфліктних ситуацій (тобто ситуацій, при яких інтереси учасників або протилежні і тоді ці моделі називаються «антагоністичними іграми», або не збігаються, хоча і не протилежні, і тоді ці моделі називаються «ігри з протилежними інтересами»).

Засновниками теорії ігор вважаються Джон фон Нейман і Оскар Моргенштерн, які розвивали теорію ігор як адекватний формальний апарат для вивчення економічної поведінки і які у 1944 році видали монографію «Теорія ігор і економічна поведінка», в якій автори узагальнили і розвинули результати теорії ігор і запропонували новий метод для оцінки корисності благ.

Основними сферами застосування теорії ігор є економіка, політологія, тактичні і воєнно-стратегічні задачі, еволюційна біологія і, в останній час, інформатика та штучний інтелект.

На практиці часто доводиться зустрічатися із завданнями, в яких необхідно приймати рішення в умовах невизначеності, тобто виникають ситуації в яких дві (або більше) сторін переслідують різні ланки, а результати будь-якої дії кожної з сторін залежать від партнера. Такі ситуації, ще виникають підчас гри в шахи, доміно та ін. і відносяться до конфліктних: результат кожного ходу гравця залежить від відповідального ходу противника, мета гри – виграш одного з партнерів.

Наприклад, відома всім гра у камінь-ножиці-папір є предметом дослідження теорії ігор, оскільки в ній є гравці, відомі їх дії (вибір одного з трьох варіантів) та виграші (які залежать від дій інших). При цьому стратегія може бути фіксована - завжди вибирати один з варіантів, наприклад камінь, або ймовірнісна - вибирати з рівною ймовірністю з усіх варіантів (ця стратегія є рівновагою Неша для даної гри).

Найпростіша гра - хрестики-нулики на дошці 3х3. Партнери по черзі ставлять на поля квадрата (дошки) хрестики і нулики, і виграє той, хто першим збудує три своїх знаки в ряд. Зрозуміло, гра триває не більше дев'яти ходів. Якщо нікому з гравців не вдається досягти мети, партія закінчується внічию. Цікаво, що навіть на такому простому прикладі можна проілюструвати багато важливих понять математичної теорії ігор.

Гра «3 в ряд» або «Хрестики-нулики» відноситься до категорії кінцевих, детермінованих, переборних, стратегічних ігор двох осіб з повною інформацією. На ігровій дошці є вільні комірки, а є зайняти гравцями. Розглянемо момент, коли гравець має зробити хід, тобто обрати вільну комірку для подальшого ходу:

1. Беремо вільну комірку.
2. Для вибраної комірки розглядаємо всі можливі тріади (у даному випадку тріада це три комірки, які можуть дати виграшну комбінацію).
3. Для кожної тріади розраховуємо коефіцієнт (a[i]).

a[i] = 0,2+ abs(0,4 кількість _союзників – 0,4* кількість _ супротивників)*

4. Для того щоб розрахувати коефіцієнт для вибраної комірки необхідно розрахувати суму ймовірностей коефіцієнтів тріад.

Sum(1) = a[1]
*Sum(2) = Sum(1) + (1 – Sum(1)) * a[2]*
*Sum(3) = Sum(2) + (1 – Sum(2)) * a[3]*
...
*Sum(n) = Sum(n-1) + (1 – Sum(n-1)) * a[n]*

Проаналізувавши коефіцієнти для всіх вільних комірок, ми обираємо потрібну для наступного ходу.

Приклад:

Дана матриця де вже є декілька зроблених ходів (рис 1.) і наступний хід потрібно зробити «0».

1	**0**	**x**
x	2	**x**
3	**0**	4

Рис. 1. Матриця гри «Хрестики-нулики»

Перш за все необхідно розрахувати коефіцієнт на перемогу для вільної комірки під номером 1. Для неї є три тріади:

a[1] 1-0-x
a[2] 1-2-4
a[3] 1-x-3

Розрахуємо коефіцієнт на перемогу для кожної тріади. Для цього треба порахувати супротивників і союзників в кожної тріаді (рис. 2).

	союзники	супротивники
a[1]	1	1
a[2]	0	0
a[3]	0	1

Рис. 2. Таблиця супротивників і союзників в тріаді для комірки під номером 1 a[1] = 0,2

a[2] = 0,2
a[3] = 0,6
На наступному етапі необхідно скласти оцінки всіх тріад цієї комірки: Sum(1) = a[1] = 0,2
Sum(2) = Sum(1) + (1 – Sum(1)) * a[2] = 0,36
Sum(3) = Sum(2) + (1 – Sum(2)) * a[3] = 0,744
Оцінка Sum(3) = 0,744
Розрахуємо коефіцієнт на перемогу для вільної комірки під номером «2». Для неї є чотири тріади: a[1] 1-2-4
a[2] 0-2-0
a[3] x-2-x
a[4] x-2-3
Таблиця супротивників і союзників в кожної тріаді для комірки 2 представлена на рис. 3.

	союзники	супротивники
a[1]	0	0
a[2]	2	0
a[3]	0	2
a[4]	0	1

Рис. 3. Таблиця супротивників і союзників для комірки під номером 2.

Відповідно коефіцієнт на перемогу для кожної тріади будуть дорівнювати:
a[1] = 0,2
a[2] = 1,0
a[3] = 1,0
a[4] = 0,6
Розрахуємо оцінки всіх тріад цієї комірки: Sum(1) = 0,2
Sum(2) = 0,36
Sum(3) = 1,0
Sum(4) = 1,0
Оцінка Sum(3) = 1,0
Наступні оцінки для комірок знаходяться у таблиці рис. 4.

1	0,744
2	1,000
3	0,936
4	1,000

Рис. 4. Таблиця оцінок вільних комірок

В результаті обчислення ми отримали дві комірки з коефіцієнтом, що дорівнює 1. Це означає, що для поточного ходу «0», є два оптимальних варіанта.

Зробивши аналогічні розрахунки для «х», ми приходимо до висновку, що «0» потрібно ставити в комірку 2 і закінчити гру вигравши.

Звичайно, для аналізу гри «Хрестики-нулики» можна обійтися без спеціальних методів, більше години він не займе. Легко виявити, що при правильній грі обох партнерів партія закінчується внічию. Її результат вирішується вже на першому ході. У хрестиків три принципових початку – зайняти кут, центр або бічну клітку дошки. Найнебезпечніший дебют хрестиків - в кутову комірку. З восьми можливих відповідей правильним для нуликів є лише хід в центр дошки. Після цього нічия досягається без праці.

УДК 004.42

Е.О. Мендрик, магістр, гр. ПІ-48м
О.І. Грабар, к.т.н., доц. каф.ІПЗ
Житомирський державний технологічний університет

ОСОБЛИВОСТІ РОЗРОБКИ МОБІЛЬНИХ ДОДАТКІВ НА ANDROID

Android – одна з сучасних швидкорозвиваючих операційних систем для мобільних пристроїв. Android заснований на unix і має свої особливості, які необхідно враховувати при створенні додатків для цієї платформи. При створенні інтерфейсу програми для будь-якої операційної системи необхідно дотримуватися наступних основних принципів розробки призначеного для користувача інтерфейсу: контроль користувачем інтерфейсу; зменшення завантаження пам'яті користувача; послідовність призначеного для користувача інтерфейсу. Для дотримання цих основних принципів, необхідно враховувати особливості взаємодії користувача з мобільним додатком, яке має досить широкий спектр змінних параметрів завдяки використанню різних апаратних пристроїв. Для операційної системи Android виділимо наступні рекомендації проектування і реалізації призначеного для користувача інтерфейсу (UI):

1. дотримання основоположних принципів створення UI;
2. орієнтація UI на взаємодію за допомогою сенсорного екрану;
3. підтримка інших способів введення крім сенсорного;
4. використання повідомлень статусного рядка;
5. обробка зміни орієнтації екрану;
6. використання розмітки, адаптується для різних пристроїв.
7. скорочення часу відгуку інтерфейсу і підвищення його юзабіліті;
8. використання фонових потоків для виконання вимогливих до ресурсів або тривалих дій;
9. підтримка взаємодії між додатками;

Розглянемо докладніше ці рекомендації. Розробнику надається досить широкий спектр можливостей для створення UI, так як Android є відкритою платформою і не обмежує вибір способів створення інтерфейсу. Проте уніфікований інтерфейс між додатком і операційною системою створює комфортні умови і є передбачуваним для кінцевого користувача. При створенні іконок, віджетів, діяльностей і меню слід керуватися, що відповідає рекомендації проходження основоположним принципам створення UI. BciAndroid пристрої мають сенсорні екрани, та дотики є одним з найважливіших принципів в Android UI. У загальних випадках в сенсорному режимі відсутній постійний фокус, але переміщення по елементах UI може призводити до появи і зникнення фокуса. Винятком є особлива ситуація яка називається «фокус в сенсорному режимі». Кнопки та інші елементи інтерфейсу повинні мати достатній розмір для управління за допомогою дотиків. Необхідно щоб кнопки відображалися однаково на екранах з різним дозволом і щільністю. Як доповнення до обробки сенсорного введення, UI повинен сприймати і фізичні методи введення. Залежно від конфігурації пристрою, фізичні режими можуть включати в себе клавіатуру, джойстик, навігаційну панель, трек. Приклад повідомлень: мобільні пристрої володіють набагато меншим об'ємом пам'яті і потужністю процесора, ніж настільні комп'ютери. Android UI повинен бути швидким і ефективним, так як користувачі працюють з додатком на ходу, тримаючи пристрій однією рукою. Якщо програма не відповідає на дії користувача протягом 5 секунд - Android відображає діалог з інформацією про те, що програма не відповідає на запити і пропонує завершити його. Діалог завершення роботи програми Більшість пристроїв Android дозволяють додаткам змінювати орієнтацію екрану з портретної на альбомну і назад. Поведінка залежить від настройки опції автоматичного повороту екрану. Додатки можуть обертатися разом з пристроєм або завжди залишатися в одному положенні, також орієнтація може змінюватися при відкритті або закритті фізичної клавіатури, при наявності такої. Це означає, що призначений для користувача інтерфейс повинен бути розроблений з урахуванням змін орієнтацій, якщо тільки додаток не призначений для використання тільки в певному режимі. Android дозволяє створити різні файли розмітки UI для альбомної і портретної орієнтацій, що рекомендується застосовувати при розробці інтерфейсів. UI повинен розроблятися з можливістю адаптації до різних типів дисплеїв. Не рекомендується використовувати абсолютну розмітку, яка може привести до неможливості використання програми при великому чи малому дозволі. При завданні розмірів рекомендується користуватися щільність-незалежними пікселями - dp, для того, щоб зробити розміри макета або його положення незалежними від фізичного дозволу і розміру пристрою виведення. Використання абсолютної розмітки на пристроях з різним дозволом екрану Один з кращих методів збереження чуйного UI - це використання фонових потоків для виконання вимогливих до ресурсів або тривалих дій. Таким чином, розробка додатків стає легше за рахунок можливостей використання та враховування можливої нелінійності UI, тому що користувачі при роботі з додатком можуть переходити в іншу програму і через стек додатків повертатися назад.

УДК 004.42

І.А. Пєвчев, магістр, група ІПЗ-1
О.І. Грабар, к.т.н., доц. каф.ІПЗ
Житомирський державний технологічний університет

ВЕБ-ОРІЄНТОВАНА СИСТЕМА ПЛАНУВАННЯ ЛОГІСТИЧНИХ ОПЕРАЦІЙ НА ПІДПРИЄМСТВІ

Web-oriented logistics operations planning system for enterprise

Сьогодні веб технології швидко розвиваються. Незалежність від платформи та доступність веб сайтів роблять їх дуже привабливими для розробників. Існуючі веб технології дозволяють будувати не лише веб сайти у звичайному сенсі цього слова, тобто сайт, що складається зі сторінок, а й високо інтерактивні сайти – веб-сервіси, які можна називати повноцінними додатками. Для роботи з інтернет-додатками не вимагається установки якого або програмного забезпечення, його налаштування і інших подібних дій. Для роботи використовується тільки інтернет-браузер.

З одним і тим же додатком і з однією і тією ж базою даних зможуть працювати десятки і сотні користувачів, що знаходяться в різних місцях. При цьому все, що їм необхідно для роботи – доступ в Інтернет. Користувачі можуть вносити дані в будь-який час доби. Власники та керівники можуть отримувати звіти та аналітику з будь-якого місця земної кулі.

Відкриваючи такий сайт користувачу доводиться лише раз завантажити його користувацький інтерфейс, а зміни на сторінці, які не потребують нових даних, виконуються швидше, оскілки не чекають обробки запитів на сервері. Саме в контексті веб-додатку розглянемо можливість автоматизації задач промислового сегменту де швидкість і точність отриманих даних прямопропорційно пов’язана з отриманням прибутку.

Сучасний бізнес вимагає автоматизації в усіх напрямках. У напрямі логістики, а зокрема внутрішньої операційної логістики підприємства, це стосується в першу чергу, так як саме тут знаходиться відправна точка руху товарів і готової продукції до споживача. Тому на даному етапі особливо гостро стоїть питання застосування новітніх технологій в даному напрямку, в першу чергу, для економії грошей і часу. Однак, навіть зараз, у багатьох компаніях логісти працюють «по-старому» – розкладають паперові карти і малюють маршрути, використовують друковані логістичні довідники для розрахунку логістичних даних. Більш обізнані користуються таблицями MS Excel та картами від Google. Такими способами можна прорахувати приблизний набір базових даних, однак вони не будуть повністю задовольняти потреби компанії. До того ж, цей варіант підходить, тільки коли є кілька стабільних маршрутів і їх не потрібно міняти кожен день. Відповідно, виробничим компаніям, потрібно шукати спеціалізовані та автоматизовані програмні рішення, які добре б інтегрувалися з діючою обліковою системою.

Перед відділом логістики в компанії зазвичай ставлять ряд завдань:

- розподілити заявки клієнтів за доступними автомобілів
- врахувати вимоги до часу доставки
- простежити фактичну доставку продукції
- при необхідності виконати аналіз різкого збільшення кількості заявок або перенесення складу в нове місце і т.д. Розрахувати економічний ефект таких дій з точки зору логістики і надати розрахунки керівнику.

Все це можна робити вручну, однак чи варто? Навіть найкращий логіст не зможе бездоганно розрахувати порядок об'їзду торгових точок, врахувати всі тимчасові вікна роботи точок, водіїв і складів, особливо коли умови кожен день нові і потрібно вкластися в заявлений час доставки.

Безумовно, крім уже згаданих карт від Google (які, звичайно ж, не є логістичними сервісами), на ринку існують рішення від великих компаній, що автоматизують роботу логіста. Для великого бізнесу вони підходять, проте невеликим і середнім компаніям вони навряд чи будуть корисні саме через свою громіздкість.

По-перше, такі програми досить важко налаштувати самостійно. По-друге, вартість впровадження і підтримки, як правило, порівнянна з ціною ПО. Додайте до того необхідність навчання логіста і придбання додаткових серверів – далеко не всі компанії можуть собі дозволити такі дорогі і складні в плані обслуговування рішення.

Виходячи з вищесказаного можна зробити висновок, що розробка веб-додатку є найбільш оптимальним і економічним рішенням для невеликих компаній, яке дозволяє скоротити витрати на утримання автопарку, підвищити керованість і прозорість транспортної логістики та в цілому підвищити рентабельність бізнесу за рахунок оптимізації логістичних процесів.

УДК 004.942

А.Г. Проценко, студ., гр. ПІ-50
В.О.Скачков, ст. викладач кафедри ІПЗ
Житомирський державний технологічний університет

ЗАСТОСУВАННЯ НЕЙРОННИХ МЕРЕЖ В РОЗПІЗНАВАННІ РУКОПИСНОГО ТЕКСТУ

На сьогоднішній день великої популярності в світі набула така галузь штучного інтелекту як нейронні мережі. Актуальність розробок в галузі нейронних мереж обумовлена перш за все тим, що застосування даної моделі широко використовуються в найрізноманітніших областях. За допомогою вирішення задач на основі нейронних мереж функціонування будь-якої системи стає ефективнішим.

Сьогодні відома велика кількість галузей застосування штучних нейронних мереж. Найбільш розповсюдженими серед них є: фінанси, економіка, медицина, наукові дослідження, інформаційні технології, штучний інтелект та ін.

Необхідно зазначити що існує велика кількість програмного забезпечення, що використовує можливості технологій штучних нейронних мереж. Прикладом можуть бути універсальні програми, що вирішують задачі від розпізнавання рукописного тексту до задач прогнозування.

Проте розглядаючи розпізнавання рукописного тексту ми маємо деякі його властивості які в подальшому викликають проблеми при розпізнаванні.

Так, однією з проблем яка викликає труднощі при розпізнаванні є те, що всі рукописні символи мають статичні і динамічні властивості. Статичні можуть полягати в розмірі або формі символу, а відмінності в динаміці можуть бути в кількості штрихів та їх порядку.

Проблеми розпізнавання символів можуть виникати, наприклад, і через великий алфавіт мови. Варто зазначити що найбільшою проблемою розпізнавання рукописного тексту була і буде неоднозначність при читанні. Іноді у людей виникають труднощі при спробі прочитати навіть власний почерк.

Ряд проблем виникає через той факт, що одні і ті ж самі символи можуть бути написані по-різному. Також рідко можна зустріти двох людей з однаковим почерком. Ця задача пов'язана з різницею шрифтів в класичній задачі розпізнавання тексту.

На відміну від шрифтів, кожен символ в рукописному тексті може мати зовсім інший стиль і вигляд в залежності від контексту, в якому здійснюється написання букв і багатьох інших факторів.

Найбільш універсальний підхід до вирішення завдання про розпізнавання рукописного тексту є нейронні мережі. Основні переваги нейронних мереж полягають в здатності навчатися самостійно і автоматично на основі вибірок, бути продуктивними на зашумлених або нечітких даних, мати можливість паралельної реалізації і бути ефективними інструментами для обробки великих баз даних.

Найпопулярнішою нейронною мережею що набула широкого використання є багатошаровий персептрон – Multi-Leyer Perceptron (MLP). Ця структура навчається за допомогою зворотного розповсюдження помилки, є однією з найбільш популярних і універсальних форм нейронних мереж- класифікаторів. Її часто використовують для розпізнавання рукописного тексту. Однак такі системи мають і суттєві недоліки. Насамперед, це те, що немає гарантії, що мережа може бути навчена за вказаний час.

При навчанні такої мережі вхідна множина сигналів розглядається як вектор. В якості вектора для розпізнавання рукописного тексту обираються закодовані значення пікселів. Навчання здійснюється шляхом послідовного представлення вхідних векторів з одночасним налаштуванням ваг відповідно визначеній процедурі. Мережа має вхідний, прихований та вихідний шари. В процесі навчання кожен вхідний вектор трансформується у вихідний вектор. Маючи вихідний вектор необхідно визначити який з нейронів вихідного шару має найвище значення активації – це і є результатом роботи нейронної мережі. Набори навчаючих векторів представляються послідовно нейронній мережі, до тих пір, поки мережа не навчиться класифікувати їх правильно.

Нейронна мережа використовує навчаючу вибірку для автоматичного виводу правил щоб розпізнати рукописний текст. Чим більше навчаючих прикладів, тим більше нейронна мережа може дізнатися про рукописний текст і в результаті це підвищить точність розпізнавання.

В нашій роботі ми використовуємо багатошаровий персептрон (MLP) для розпізнавання рукописного тексту та множину зображень бази MNIST. Під час тестування такої мережі було виявлено що на 10 000 тестових зображеннях точність розпізнавання досягає приблизно 92%. Проте, чим більша кількість прихованих шарів в мережі, тим більша точність розпізнавання. Але необхідно зважати що велика кількість прихованих шарів впливає на швидкість навчання тому в нашій мережі емпіричним методом було обрано оптимальну кількість прихованих шарів в мережі – 2.

УДК 004.942

А.В. Сотніченко, бакалавр, гр. ПІ-50
О.І. Грабар, к.т.н., доц. каф. ІПЗ
Житомирський державний технологічний університет

МОЖЛИВОСТІ РОЗШИРЕННЯ NAVMESH COMPONENTS ДЛЯ РЕДАКТОРА UNITY

Possibilities for expansion NAVMESH components for UNITY editor

В 2017 році для редактора Unity вийшло розширення розроблене командою Unity Technologies. Воно доповнює можливості стандартного компонента NavMesh і знаходиться в відкритому доступі.

NavMesh відповідає за створення поверхні яка видима тільки в редакторі. Ця поверхня відповідає за знаходження найкоротшого шляху до вказаної точки. Використовується для обрахування переміщення NavMesh Agent. NavMesh дуже спрощує роботу з переміщенням і пошуком шляхів. Але в нього є один великий недолік. Він абсолютно статичний.

Що б працювати з динамічними об'єктами доводилось викручуватись. Наприклад, при генерації випадкової локації потрібно було завчасно створювати невеличкі шматки території і на них прораховувати розташування NavMesh, а вже потім генерувати локацію з цих завчасно підготовлених фрагментів.

Завдяки новому розширенню з'явилась можливість динамічно формувати NavMesh прямо з коду.

Розширення не йде одразу з редактором. Його потрібно скачувати та встановлювати окремо.

Основними компонентами розширення є:

1. NavMeshSurface
2. NavMeshModifier
3. NavMeshModifierVolume
4. NavMeshLink

NavMeshSurface відповідає за створення NavMesh на певній області для переміщення агентів певного типу. Кожну область можна настроїти по своєму і переміщення персонажа на них буде відрізнятись. Цей компонент прикріпляється до об'єкта на сцені. До нього можна звертатися з коду. І вже в коді викликати створення NavMesh. Його можна створювати відносно графіки або відносно фізичних коллайдерів. Коллайдери це компоненти які описують форму об'єкта для фізичних зіткнень. Вони не дають об'єктам проходити одне через одного. Колайдери краще підійдуть коли необхідно моделювати моменти зв'язані з фізикою. Якщо використовувати графіку то NavMesh побудується по геометрії моделей на які буде накладатись. Також в налаштування можна виставити відносно якого шару це буде працювати. Можна зробити так що б об'єкти з певним шаром ігнорувались.

NavMeshModifier дозволяє точно налаштовувати поведінку конкретного об'єкта під час генерації NavMesh. Цей компонент призначається до ігрових об'єктів. Він є заміною стандартного, який доступний в редакторі. Але на відміну від стандартного він є динамічним. Налаштування стандартного можна змінити тільки в редакторі і за його допомогою згенерувати статичний NavMesh. А що б змінити його доведеться перезапускати гру і заново створювати новий NavMesh.

NavMeshModifierVolume дозволяє відмічати певну область як тип в той час як NavMeshModifier призначає тип області певним ігровим об'єктам. Його можна використовувати для позначення якихось рухомих областей. Такою областю може бути область дверей і за певних умов персонаж не зможе пройти цю область. В нього можна налаштувати розмір області яка буде модифікована. Також можна вказати центр модифікатора відносно об'єкта до якого він буде прикріплений.

NavMeshLink дозволяє створювати зв'язок між двома віддаленими точками NavMesh. Завдяки цьому можна реалізувати стрибки, переміщення по мостах. Коли об'єкт входить в одну з точок то одразу ж переміщується в іншу, ігноруючи будь-які перешкоди на шляху. В параметрах можна вказати тип агента який може використовувати цей елемент. Завдяки цьому можна легко зробити так що б тільки певні персонажі змогли переходити через якусь область. Початкову та кінцеву точки. Між ними і буде відбуватись переміщення об'єкта. Можна вказати довжину цього переходу. Також можна змінити в яку сторону буде працювати цей компонент. Він може працювати як в обидві сторони. А може працювати тільки в одну. І вже коли об'єкт переміститься то не зможе потрапити назад тим самим шляхом.

Кожен з модифікаторів впливає на всю ієрархію об'єктів. Тобто якщо об'єкт має дочірні об'єкти то обраний модифікатор застосується до кожного дочірнього. Вони отримають такі ж властивості як і їх батьківський об'єкт.

Підсумовуючи все вищесказане можна зробити висновок що нові NavMesh Components є дуже зручними і корисними в застосуванні. Вони легко вирішують складні проблеми. В них не важко розібратись. І завдяки ним не доводиться викручуватись і придумувати обхідні шляхи для реалізації певних задумок. То ж це розширення можна вважати життєво необхідним для редактора.

УДК 004.942

А.О. Тимченко, студ., гр. ПІ-50
В.О. Скачков, ст. викладач кафедри ІПЗ
Житомирський державний технологічний університет

ФУНКЦІЯ АКТИВАЦІЇ НЕЙРОНА RELU

Штучна нейронна мережа – це загальна назва для цілого класу моделей. Як правило, вони представляють собою комбінацію нелінійних перетворень вхідних даних. Останнім часом все більшої популярності набуває глибоке навчання (deep learning). Глибоке навчання є галуззю в машинному навчанні і ґрунтується на деякому наборі алгоритмів. За допомогою глибоких нейронних мереж успішно вирішуються різні завдання, пов'язані з класифікацією даних, прогнозування подій, розпізнавання мови, обробки текстів і т.д:

Класичний алгоритм зворотного поширення помилки працює добре з двошаровими та тришаровими нейронними мережами, але при подальшому збільшенні глибини мережі можуть виникнути проблеми. Одна з яких це так зване згасання градієнту. В процесі поширення помилки від початкового шару до вихідного відбувається множення поточного результату на похідну функції активації. Використовуючи традиційну сигмоїдну функцію активації, похідна якої має область визначення менше 5 одиниць, помилка після проходження декількох шарів може бути близькою до нуля. Якщо ж навпаки взяти функцію активації в якої похідна необмежена (як, наприклад, гіперболічний тангенс), то може статися збільшення помилки в процесі навчання, що призведе до нестійкого навчання мережі.

За останні роки великої популярності набула функція активації ReLU (rectified linear unit). Її похідна дорівнює або одиниці, або нулю, і тому не може статися розростання або загасання градієнтів. Більш того, використання даної функції приводить до проріджування ваг.

ReLU має наступну формулу: o(x) = max(0, x). (рис 1.1)

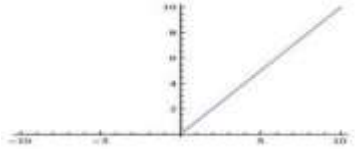

Рис. 1.1 графік функції ReLU Розглянемо позитивні та негативні сторони ReLU.

Позитивні сторони:

1. Сигмоїд і гіперболічний тангенс потребує для виконання операцій велику кількість системних ресурсів, таких як піднесення до степення, що при великій кількості шарів та нейронів сповільнюють процес навчання, в той час як ReLU може бути реалізований за допомогою простого порогового перетворення матриці активацій в нулі.

2. Застосування ReLU істотно підвищує швидкість збіжності стохастичного градієнтного спуску в порівнянні з сигмоид і гіперболічним тангенсом. Вважається, що це обумовлено лінійним характером і відсутністю насичення даної функції.

Негативні сторони:

1. На жаль, ReLU не завжди достатньо надійні і в процесі навчання можуть виходити з ладу («вмирати»). Наприклад, великий градієнт, що проходить через ReLU, може привести до такого оновлення ваг, що даний нейрон ніколи більше не активується. Якщо це станеться, то, починаючи з даного моменту, градієнт, що проходить через цей нейрон, завжди буде дорівнює нулю. Відповідно, даний нейрон буде необоротно виведений з ладу. Наприклад, при дуже великій швидкості навчання (learning rate), може виявитися, що до 40% ReLU «мертві» (тобто, ніколи не активуються). Ця проблема вирішується за допомогою вибору належної швидкості навчання.

В даний час існує декілька різновидів даної функції:

1. ReLU з втратами є спробою вирішити проблему, яка пов'язана з втратами звичайної являє собою одну зі спроб вирішити виходу з ладу звичайних. ReLU на інтервалі $x < 0$ на своєму виході дає 0, а LReLU на цьому інтервалі має невелике від'ємне значення (кутовий коефіцієнт близько 0,01). Тобто функція для LReLU має вигляд $f(x) = \alpha x$ при $x < 0$ і $f(x) = x$ при $x \geq 0$, де α – мала константа. Деякі дослідники повідомляють що успішно застосували дану функцію, але результати не завжди стабільні.

2. Для ReLU з параметрами – кутовий коефіцієнт не задається на початку, а обраховується на основі даних які отримуються під час. Процес зворотного поширення помилки і оновлення для PReLU досить простий і подібний до процесу для традиційних ReLU.

В своїй роботі, по розпізнаванню рукописних цифр на базі mnist та багатошарової нейронної мережі, було використано функцію активації ReLU. Вона дозволила збільшити швидкість навчання мережі в порівнянні з сігмоідом, при цьому помилка навчання була менша.

УДК 004

І.С. Хоменко, студ. гр. ПІ-49в
В.О. Сидорчук, студ. гр. ПІ-49в
Науковий керівник: к.т.н., доц., доцент кафедри ІПЗ Сугоняк І.І.
Житомирський державний технологічний університет

СИСТЕМА МОНІТОРИНГУ РОБОТИ СНІГОПРИБИРАЛЬНОЇ ТЕХНІКИ

Технології стрімко розвиваються, і сьогодні ні для кого не є проблемою відслідкувати потрібний тролейбус, трамвай або маршрутне таксі за допомогою свого смартфону. Технології GPS трекінгу широко використовуються в соціальних сферах з метою покращення сервісу, також однією з переваг слідкування є моніторинг ресурсів. Автоматизована система слідкування за снігоприбиральною технікою надасть можливість оптимізувати процес прибирання снігу з наших вулиць, а також допоможе відслідковувати ресурси які були затрачені.

Однією з основних задач системи є прокладання оптимального маршруту, що зменшить не цільове використання палива, а також дає можливість швидше і ефективніше прибрати найбільш завантажені вулиці нашого міста.

Другою, але не менш важливою проблемою є не цільове використання ресурсів. Система вирішила проблему автоматичним розрахунком потрібної кількості палива та хімікатів, що зробить не можливим використання ресурсів недобросовісними працівниками в свою користь.

Важливим фактором системи є трьохсторонній зв'язок, який дозволяє диспетчеру, системі та водію транспортного засобу завжди залишатися на зв’язку. Якщо виникне не передбачувана ситуація, водій зможе подати сигнал, система автоматично призначить на цей маршрут іншу техніку, а також сповістить диспетчера який зможе вислати на місце ремонтну бригаду.

До цього часу всі три задачі не були вирішені, вирішені частково або окремо одна від одної, що не давало повної автоматизації та погіршувало процес прибирання доріг. Це в свою чергу створювало проблеми пересування вулицями міста.

Головним об’єктом дослідження був процес побудови оптимальних маршрутів для снігоприбиральної техніки. Прокладені маршрути оптимізують процес прибирання. При прокладанні маршруту враховується декілька факторів:

- засніженість вулиць,
- не пересікання з іншою технікою,
- мінімізація повторного проїзду по тому самому маршруті.

Це допоможе скоротити витрати на пальне, та зменшити час за який всі вулиці будуть прибрані.

В досліджені було реалізоване слідкування за снігоприбиральною технікою за допомогою технологій GPS та ГЛОНАС. Данні надходять з GPS трекеру встановленого на транспорті, або смартфону водія.

На смартфон водія було встановлено розроблений нами android додаток, який буде збирати та відправляти на сервер інформацію про техніку.

Також за допомоги додатку водій має прямий зв'язок с диспетчером та зможе при необхідності викликати допомогу.

Данні прийняті від додатку зберігаються та обробляються на сервері системи. Оператор, за допомогою веб додатку, може:

- переглянути місце знаходження всіх одиниць техніки
- переглянути всі прокладені маршрути
- переглядати яка частина вулиць вже прибрана, а яка потребує призначення більшої кількості техніки.
- призначити додаткову техніку на маршрут
- моніторити ресурси техніки (пальне, хімікати для посипання, та вільне місце в кузові(в тому випадку якщо це саме збиральна техніка)
- комунікувати з водіями(попередити про затори, аварію, а також отримати повідомлення про несправність та відрядити ремонтну бригаду на місце пригоди)

Система складається з 3х основних елементів:

1. Android додатку, який збирає данні з датчиків техніки.
2. Бази даних, в нашому випадку MySQL. В базі зберігаються данні які ми отримаємо від техніки.
3. Веб додаток, який обробляє, структурує та виводить потрібну інформацію оператору.

Щоб зробити систему універсальною та не прив’язуватися до одного або декількох виробників датчиків та систем супутниково стеження, наш Android додаток обробляє та структурує отримані дані і відправляти на сервер в потрібному для системи вигляді.

Практичне значення одержаних результатів. Система дозволяє слідкувати за вантажною технікою, спец технікою а також легковими автомобілями які належать комунальним підприємствам. На основі отриманих результатів, комунальне підприємство може систематизувати процес прибирання снігу, та витрату ресурсів.

УДК 004.42

Р.В. Шпирко, магістр, гр. ПІ-48м
О.І. Грабар, к.т.н., доц. каф.ІПЗ
Житомирський державний технологічний університет

РОЗРОБКА ДОДАТКУ ФІЛЬТРАЦІЇ СПАМУ

Проблема фільтрації спаму є дуже актуальною в наш час. Безкоштовні продукти дають незадовільні результати, або вимагають додаткової "доведення", для чого потрібна висока кваліфікація. Високий рівень фільтрації при установці "з коробки" досягається тільки на комерційних продуктах, причому не всіх.

Фільтрація здійснюється після повної передачі листа. Причому в багатьох випадках - після остаточного приймання листа, коли відправник отримав повідомлення про те, що лист прийнято. Тому такі фільтри не знижують обсяг отриманого трафіку. Крім того, в разі, якщо отримання листа підтверджено, а лист розпізнано як спам, поштовий сервер зобов'язаний сформувати квитанцію відправнику листа, а це призводить до високого рівня т.зв. "Вторинного" спаму - небажаних поштових квитанцій, вірусних звітів і т.п. Тонка фільтрація є надзвичайно ресурсномісткою. Найчастіше страждають бізнес-листи з легальними комерційними пропозиціями. Неможливо сказати, яке саме правило спричинило розпізнавання письма як спаму.

"Жорстка" фільтрація - це, в деякому роді, повернення до минулого. Її завдання – «очистити» трафік від часто зустрічається спаму простими методами, причому, по можливості, до того, як лист прийнято. Замість сигнатур всередині листи вона буде шукати сигнатури програми-розсильників. Розглянемо основні методи жорсткої фільтрації: 1. Фільтрація за IP адресою клієнта SMTP. Фільтрація за чорних списків і списків динамічних мереж. 2. Фільтрація по зворотному імені клієнта SMTP: перевірка можливості розв'язання IP адреси, перевірка імені по сигнатурам імен динамічних мереж / мереж кінцевих користувачів. 3. Фільтрація по команді HELO / EHLO: перевірка на відповідність стандартам, перевірка по базі сигнатур, перевірка відповідності HELO і доменного імені. 4. Фільтр по відправнику: фільтрація за чорних списків, фільтрація за SPF. 5. Грейлістинг (graylisting – перевірка на коректну обробку помилок). 6. Фільтрація за стійким сигнатурам в заголовках листа

Жорстке очищення трафіку дуже часто може проводитися засобами самої поштової програми, що робить можливим детектування спаму по пунктам 1–5 без отримання самого листа, а при фільтрації заголовків – без підтвердження отримання. Це сильно знижує поштовий трафік, навантаження на поштовий сервер (тому що фільтрація йде по дуже простим правилам) і усуває вторинний спам. Крім того, основний "безневинною жертвою" жорсткої очищення стає не бізнес-кореспонденція, а листи «роботів» – листи з форумів, повідомлення про реєстрацію, розсилки прайс-листів та інші автоматизовані розсилки. Недоліки жорсткої очищення: неможливо домогтися високого ступеня очищення без помилкових спрацьовувань. Необхідність ведення «білих списків», як по відправникам, так і щодо одержувачів, тому що жорстка очищення малопридатна для адрес з широкою географією (тобто адрес на які не ведеться активної переписки але приходить багато «разових запитів»). Необхідність постійної кваліфікованої підтримки. Висока швидкість реакції на запит користувача частково нівелює негативний ефект помилкових спрацьовувань. Аналіз спаму попався на спамтрапи, рапортувати користувачами або виявленого по заголовкам або методом тонкої фільтрації дозволяє створити в системі «зворотний зв'язок» і збільшити якість фільтрації. Складно домогтися маркування спам-листів без їх відсіву. Фільтри є «нестерпною». Тобто з них практично неможливо сформувати закінчений продукт, який можна було б поширювати. На практиці, за рахунок жорсткої очищення на великому вузлі (порядку декількох тисяч активних поштових скриньок), за умови постійної підтримки вдалося домогтися досить цікавих результатів. На живій системі, в якій використовується майже 100000 сигнатур по IP адресами та мереж, близько 1500 сигнатур по іменах хостів і HELO, понад 300 сигнатур у заголовках листів, без застосування грейлистинг і SPF: фільтрується 95–99 % спаму в залежності від обсягу спаму на поштову скриньку. Для сильно «засвічених» адрес рівень фільтрації трохи вище. Найгірше ловляться т.зв. «Нігерійські» листи (шахрайські) послані з легальних поштових серверів. З ними чудово справляється фільтр тонкого очищення. Результат спільної роботи – практично повна відсутність спаму. Фільтрується без додаткових дій понад 90 % нових поштових черв'яків in-the-wild, що істотно перевершує показники евристики будь-якого антивіруса. Відсоток помилкових спрацьовувань менше 1 %. При цьому втрати реальної кореспонденції не більше сотих часток відсотка, але досить високий рівень втрат «автоматичних» листів. Ці характеристики краще, ніж у деяких комерційних продуктів "тонкої" очищення. Думка про те, що «жорсткі» методи віджили свій вік, є не цілком точною. За умови кваліфікованої підтримки, методи жорсткого очищення можуть використовуватися на поштових серверах середнього розміру. На великих поштових вузлах використання жорсткої очищення в якості основної призведе до занадто великої кількості винятків (білих списків). Для дрібних поштових вузлів проблематично поповнення бази сигнатур (чорних списків), але можливо кооперування – тобто створення централізованого фільтра, який буде пропускати пошту для декількох поштових доменів.

УДК 004.42

В.С. Шулятицький, бакалавр, гр. ПІ-50
О.І. Грабар, к.т.н., доц., каф.ІПЗ
Житомирський державний технологічний університет

РОЗРОБКА WEB-ОРІЄНТОВАНОЇ СИСТЕМИ ПОДАЧІ БЕЗКОШТОВНИХ ОГОЛОШЕНЬ НА БАЗІ PHP

Development of web-oriented free systems submitanad on the php base

З розвитком Web-систем, все більше й більше технологій та процесів переходить в онлайн роботу, і торгівля не стала виключенням. В наш час в інтернеті можна купити все, і так само можна продати будь-яку річ. Стан, колір, розмір – це все не важливо, головне бажання це зробити і в цьому нам допомагають онлайн системи безкоштовних оголошень так звані «дошки оголошень». Вони стали невід'ємною частиною нашого життя.

Актуальність обраної теми полягає в тому, що такі онлайн системи мають дуже великий попит у використанні та розробці. Web-орієнтована система подачі безкоштовних оголошень буде забезпечувати можливість користувачам знайти потрібну роботу або швидко та зручно продати чи купити товар у будь-який час навіть не виходячи з дому. Користуватися нею зможе будь-хто, незалежно від віку, статі чи професії. Для розробки даної системи було обрано мову програмування PHP.

PHP (рекурсивний акронім словосполучення PHP: HypertextPreprocessor) – це поширена мова програмування загального призначення з відкритим вихідним кодом. PHP сконструйований спеціально для ведення Web-розробок і його код може впроваджуватися безпосередньо в HTML. Замість рутинного виведення HTML-коду командами мови (як це відбувається, наприклад в C), скрипт PHP містить HTML з вставками та фрагментами коду. Код PHP відділяється спеціальними початковим і кінцевим тегами, які дозволяють "перемикатися" в "PHP-режим" і виходити з нього. PHP відрізняється від JavaScript тим, що PHP-скрипти виконуються на сервері і генерують HTML код, який надсилається клієнту.

Мова програмування PHP не вимагає обов'язкового оголошення типів даних на початку програми, як інші мови програмування. І тому ви можете відразу ж використовувати будь-яку кількість змінних і задавити їм будь-які значення, не оголошуючи, текстова це змінна чи ні - це за вас зробить сам інтерпретатор. Але краще буде, якщо ви будете десь запам'ятовувати який тип має кожна змінна і пам'ятати по ходу всієї програми яке значення вона може прийняти. Це допоможе уникнути при написанні великихскриптів для сайту безліч різних помилок, які дуже часто допускають початківці програмісти, і, без сумніву, допоможе захистити сайт від злому і крадіжки з нього корисної інформації. Використовуючи масиви в цьому середовищі розробки, ви, безсумнівно, можете обробляти одночасно величезну кількість різних значень, незалежно до якого типу даних вони належать. Використовувати масиви необхідно тільки при великому обсязі інформації, яку необхідно обробляти протягом роботи програми для отримання кінцевого результату. Це дуже сильно спрощує роботу з більшістю розрахунків.

Цікавою особливістю цього універсального і багатостороннього інтерпретатора є ще те, що він дуже непогано виконує роботу з текстовими значеннями. Безліч різних текстових операторів в мові програмування PHP дозволяють шукати входження в тексті, вирізати фрагмент з тексту і робити багато інших операцій з текстовими значеннями. А використовуючи регулярні вирази, ви зможете здійснювати незалежний пошук будь-якого потрібного фрагмента в тексті.

На PHP також можна розробляти GUI-додатки, хоча він і не дуже поширений в даній області. Для створення кросплатформних додатків служать пакети PHP-GTK і PHP-Qt, що представляють собою обгортки для відповідних популярних бібліотек віджетів. Також існує середовище розробки кросплатформних додатків Devel Next. Для створення графічних додатків для Windows існують вільні пакети WinBinder (написаний на Ci, фактично - обгортка для WinAPI), PQBuilder (написаний на PHP з використанням бібліотеки PHPQt5), а також попередник Devel Next - середовище швидкої розробки Devel Studio. Крім цього існує реалізація PHP для .NET / Mono - Phalanger і для JVM - JPHP, результатом компіляції PHP-коду в Phalanger може бути будь-який .NET-додаток, в той же час JPHP підтримує розширення Swing, майже повністю схоже з середовищем Java.

Отже на основі можливостей мови програмування PHP можна реалізувати досить великий та потужний функціонал WEB-системи і не тільки. PHP має також велику кількість фреймворків та додатків, які можуть досить легко спростити або полегшити написання програмного коду, та значно збільшити функціональні можливості системи. Деякі популярні системи управління контентом, такі як WordPress, Drupal та Joomla також створені на базі PHP. Згідно з дослідженнями в області веб-технологій, що проводяться W3Techs, WordPress домінує на ринку CMS систем і використовується в 25 % всіх сайтів.