ЗМІСТ

ВСТУП

Питання дистанційного навчання є актуальним у наш час. Після появи у середині XX ст. ЕОМ - пристроїв для обробки інформації уперше в історії людства з’явився такий спосіб запису й довгострокового зберігання раніше формалізованих професійних знань, за допомогою якого ці знання дістали змогу безпосередньо без опосередкованих впливів на людину впливати на режим роботи виробничого обладнання. Масове впровадження персональних ЕОМ і особливо програм, які базуються на використанні штучного інтелекту, стало причиною нового унікального явища. Знання стали вироблятися штучним інтелектом - машиною. Знання почали саморозмножуватися, при чому із постійно зростаючою швидкістю. Це стало причиною стрибкоподібного зростання наукомістких технологій. Відбулася четверта інформаційна революція. Людство вступило на перший ступінь нового "інформаційного" суспільства. На такому етапі розвитку людства та інформатизації суспільства та освіти розвинулося дистанційне навчання, яке, поєднуючи у собі кращі риси інших форм, є найбільш перспективною, гуманістичною, інтегральною формою отримання освіти. У процесі навчання дистанційне навчання забезпечує виконання організаційної, навчальної, керуючої, контролюючої комунікативної, діагностичної, прогностичної функцій, використовуючи при цьому сучасну комп'ютерну техніку, телекомунікаційні мережі та програмне забезпечення.

У системі освіти воно відповідає принципу гуманістичності, згідно якого ніхто не повинен бути позбавлений можливості навчатися через бідність, географічну або тимчасову ізольованість, соціальну незахищеність та неможливість відвідувати освітні установи через фізичні вади або зайнятість виробничими та особистими справами. І тому говорити про дистанційне навчання зараз є актуальним і навіть потрібним, бо ця технологія є надзвичайно зручною і сучасною.

Мета дослідження - розглянути сутність технологій дистанційного навчання, існуючі види, визначити його значення і місце у сучасній системі навчання.

Завдання дослідження - визначення принципів та умов застосування дистанційного навчання, виявлення критеріїв ефективності такої системи.

Об’єкт дослідження - дистанційне навчання, як найбільш сучасна, зручна, гуманістична, інтегративна форма навчання.

Предмет дослідження - властивості та сторони дистанційного навчання, його технології.

Проект розділено на такі складові:

- єдина серверна частина(Java EE);
- WEB - кліент (JavaScript + JQuery);
- iOS - кліент (Objective - C + CocoaTouch);
- Android - кліент (Java SE + Android API);

Вебсервіс (англ. web service) — програмна система, що ідентифікується URI, та публічні інтерфейси та прив'язки якої визначені та описані мовою XML. Опис цієї програмної системи може бути знайдено іншими програмними системами які можуть взаємодіяти з нею відповідно до цього опису з використанням повідомлень, що базуються на XML та передаються за допомогою інтернет-протоколів.

Термін "веб-служба" організацією W3C застосовується до багатьох різних систем, але в основному термін стосується клієнтів та серверів, що взаємодіють за допомогою повідомлень протоколу SOAP. В обох випадках припускається що існує також опис доступних операцій у форматі WSDL. Хоча наявність цього опису не є вимогою SOAP, а радше передумовою для автоматичного генерування коду на платформах Java та .NET на стороні клієнта.

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

Android і iOS клієнти є мобільними додатками.

Мобільний додаток (англ. «Mobile app») — програмне забезпечення, призначене для роботи на смартфонах, планшетах та інших мобільних пристроях. Багато мобільних додатків встановлені на самому пристрої або можуть бути завантажені на нього з онлайнових магазинів додатків, таких як App Store, Google Play, Windows Phone Store та інших, безкоштовно або за плату.

Спочатку мобільні додатки використовувалися для швидкої перевірки електронної пошти, але їх високий попит призвів до розширення їх призначень і в інших областях, таких як ігри для мобільних телефонів і GPS, спілкування, перегляд відео та користування інтернетом. Ринок мобільних додатків сьогодні дуже розвинений і неухильно зростає.

1 ЗАГАЛЬНИЙ РОЗДІЛ

1.1 Аналіз існуючіх систем дистаційного навчання

Інформатизація освіти є однією з ключових умов успішного розвитку сучасного суспільства, проте наслідком стрімкого розвитку новітніх інформаційних технологій у світі стала нова соціальна економічна проблема - інформаційна нерівність. Основною метою впровадження дистанційної форми навчання є вирішення цієї проблеми, а також швидке й зручне поширення знань, забезпечення доступності освіти всім верствам населення. Значною мірою ця мета реалізується за допомогою програмних засобів, побудованих на сучасних інформаційно-комунікаційних технологіях, які одержали загальну назву «системи дистанційного навчання» (СДН). До найпопулярніших СДН можна віднести:

- Moodle
- Lotus Learning Space;
- Blackboard Learning System;
- REDCLASS;
- «Прометей»;
- «Віртуальний Університет»;
- ГЕКАДЕМ;
- e-University;
- «Веб-клас ХПІ».

Було поставлено за мету здійснити аналіз навчальних можливостей найбільш поширених СДН та з’ясувати особливості їх застосування відповідно до організації дистанційного навчання. В такому контексті охарактеризуємо їх докладніше.

Система дистанційного навчання Lotus Learning Space. Ця СДН, розроблена компанією IBM, надає можливість вчитися і викладати в асинхронному режимі, звертаючись до матеріалів курсів у зручний час, брати участь в он-лайн заняттях у режимі реального часу. Викладач може створювати зміст курсу в будь-яких програмах і потім розміщувати створений матеріал у Learning Space. Програма має гнучку систему редагування й адміністрування курсу, дозволяє вибирати різні режими викладання і відстежувати поточні результати роботи студентів. Курси організовані у вигляді послідовних занять, які можуть бути самостійними, інтерактивними або колективними. Самостійні заняття зазвичай містять матеріали для читання і тести, які необхідно виконати після вивчення.

Інтерактивні заняття включають лекції у віртуальному класі, участь в он-лайновій дискусії або чаті, роботу з віртуальною дошкою (Whiteboard) і системою сумісного перегляду Web- сайтів (Follow me). Інтерактивні заняття плануються на певну дату і час та проводяться викладачем у віртуальному класі в режимі реального часу. Поточні результати студентів (етап проходження курсу, оцінки, витрачений час, кількість звернень та ін.) зберігаються в базі даних. Ця інформація доступна викладачеві у будь-який час у вигляді звітів різної форми. Колективні заняття передбачають заняття в оф-лайновій і он-лайновій дискусіях, чаті [9]. У системі Lotus є п'ять спеціалізованих баз даних (БД) [1]:

- Schedule - дозволяє учасникам переглядати навчальні матеріали і вправи, брати участь у тестах, вирішувати завдання і проводити дослідження; відображає структуру курсу навчання, створену викладачем;

- У БД MediaCenter зберігаються статті, новини, глави книг, реферати та звіти; надає доступ до мережі World Wide Web й інших зовнішніх джерел інформації; може зберігати додаткову інформацію, яка виходить за рамки курсу навчання і дозволяє студентам проводити індивідуальні дослідження

- БД CourseRoom - інтерактивне середовище, в якому студенти спілкуються, ведуть дискусії між собою та з викладачем;

- Сучасні інформаційні технології та інноваційні методики навчання в підготовці фахівців: методологія, теорія, досвід, проблеми;

- БД Profiles містить інформацію про студентів і викладачів, контакти (адреса, номер телефону і т.д.), фотографії та відомості про процес навчання, отриманий досвід і захоплення;

- БД Assessment Manager є засобом, за допомогою якого викладачі оцінюють роботу кожного студента і повідомляють йому результати;

Поточна версія Lotus Learning Space 5.01 забезпечує міграцію дистанційних курсів із попередніх версій Lotus Learning Space; має можливість розробки дистанційних курсів із використанням програмного забезпечення Adobe Flash, Adobe AuthorWare, Adobe DreamWeaver; оснащена системою тестування; відповідає останнім міжнародним рекомендаціям в галузі стандартизації дистанційного навчання; має можливість вбудовування в дистанційний курс сесій із текстовим чи звуковим чатом, відеоконференціями; підтримує режим швидкого відображення веб-сторінок на комп'ютерах користувачів і режим копіювання фрагментів стільниці комп'ютера лектора на комп'ютери користувачів, а також тестування у реальному часі.

Система Blackboard Learning System. Особливістю віртуального

середовища навчання Blackboard, розробленого компанією Blackboard Inc., є наявність блоку керування, що налаштовується за принципом відкритої архітектури. Система Blackboard має такі компоненти [2]:

- «Керування курсами» - дозволяє створювати потоки студентів для проходження спільного навчання і надає можливість посеместрово відслідковувати міграцію студентів;

- «Редагування контенту» - за допомогою Wysiwig (візуального редактора); - «Адаптація потоку» - керування навчальним процесом, узгодження з навчальними програмами, заходами по звітності тощо;

- «Складання плану» - дозволяє використовувати збережений або створювати новий план занять;

- «Навчальні групи» - для встановлення послідовності занять;

- «Курсовий пакет» - весь пакетований контент у форматі Blackboard (додаткова література, мультимедійні матеріали, опитувальники);

- «Інструментарій педагога і студента» - глосарій, дошки оголошень, Electric Blackboard;

- «Керування особистісною інформацією» - календар, список задач, електронна пошта;

- «Атестація» - надає можливості з перевірки рівня знань тих, хто навчається за допомогою програмних пакетів «Оцінка», «Завдання», «Щоденник», «Дошка звіту і підготовки». Версія 9.0 програми Blackboard Learn була розроблена у співпраці з викладачами різних країн для вирішення загального для них непростого завдання - організації процесу навчання для різнотипних груп слухачів як в умовах аудиторних, так і позааудиторних занять. Blackboard 9.0 надає цілу низку ефективних засобів для вирішення цього завдання - від засобів соціального навчання до інновацій Веб 2.0, а також інтегровані компоненти, що дозволяють навчальним закладам здійснювати більш якісну оцінку окремих осіб, груп, програм і шкіл. Окрім того, випуск цієї версії значно підвищує відкритість і гнучкість платформи Blackboard, дозволяючи навчальним закладам так чи інакше посилювати закладені в програму можливості або ж використовувати її як розширену базу для додаткових технологій, які їм необхідні для підтримки свого власного підходу до навчання. Відкритість у цій версії Blackboard підвищена за рахунок включення в неї елементів, що забезпечують інтеграцію систем із відкритим вихідним кодом і програм, створених аматорами - включаючи системи управління курсами навчання Sakai і Moodle, доступ до яких не вимагає окремої реєстрації, а забезпечується єдиним входом у рамках платформи Blackboard [9];

Система дистанційного тренінгу (СДТ) REDCLASS. Цей комплекс програмноапаратних засобів, навчальних матеріалів і методик навчання дозволяє дистанційно навчатися, підвищувати кваліфікацію, контролювати знання в будь-яких галузях діяльності людини, а також напрацьовувати практичні навички з експлуатації й управління програмними продуктами, устаткуванням і технологіями. В основному ця СДТ застосовується для організації корпоративної системи дистанційного навчання і підвищення кваліфікації співробітників компаній; організації системи підвищення кваліфікації фахівців різних напрямів діяльності на базі спеціалізованих навчальних центрів; організації дистанційного тестування для контролю знань, набутих за допомогою як традиційного очного, так і дистанційного навчання, а також з метою використання у вищих і середніх навчальних закладах у процесі навчання і тестування студентів [3].

СДТ REDCLASS версії 2.1 має модульну структуру і може поставлятися замовникові в різній конфігурації залежно від його цілей і вимог.

Система дистанційного навчання «Прометей». За допомогою СДН «Прометей», що розроблена російською компанією «Віртуальні технології в освіті», можна побудувати в Інтернет або Інтранет віртуальний університет і проводити дистанційне навчання великої кількості слухачів, автоматизувавши весь навчальний цикл - від прийому заявок до позначки про видачу підсумкового сертифіката.

СДН «Прометей» версії 4.2 має такі нові можливості:

- управління доступом до курсів для різних груп користувачів;
- управління дистанційним навчальним процесом із використанням Інтернет або Інтранет;
- розміщення на навчальному порталі інформаційних і довідкових матеріалів;
- складання і контроль планів навчання і проведення занять;
- створення, імпорт тестів, а також навчальних матеріалів у різних форматах, в т.ч. імпорт електронних курсів у форматах IMS і SCORM;
- забезпечення взаємодії слухачів і викладачів за допомогою форумів (консультацій), чатів (семінарів) й інших електронних засобів спілкування;
- проведення екзаменаційного і самоперевірочного тестування, робота над помилками;
- формування різних звітів за наслідками навчання;

До переваг системи «Прометей» можна віднести 10 видів тестів, можливість використання графіки і мультимедіа в тестах, можливість

побудови додаткових звітів, можливість створення розподіленої системи дистанційного навчання (на базі центрального і філіальних вузлів), можливість інтеграції з кадровими, бухгалтерськими, інформаційними і ERP-системами тощо.

Для організації навчання у СДН «Прометей» застосовують:

- реєстрацію на курси за типом електронного магазину;
- календарні плани вивчення курсів;
- новий принцип організації навчально-методичних матеріалів - до курсу можна прикріпити будь-яку кількість електронних книг;
- гнучку підсистему обліку платежів (витрат);
- підсистему реєстрації/видачі сертифікатів;
- студент може входити до будь- якої кількості груп із одним логіном;
- можливість поєднання ролей (тьютор може одночасно бути і організатором);
- історію взаємодії зі слухачем, що заповнюється організатором;
- програми навчання, що об'єднують декілька курсів;
- тотальний контроль діяльності учасників навчального процесу;
- автоматизація виконання адміністративних операцій через веб-серверінтерфейс.

СДН «Прометей» ВНЗ коледжам та іншим навчальним закладам можна застосовувати для організації: дистанційних підготовчих курсів для абітурієнтів; дистанційного навчання і консультування студентів; самостійної роботи студентів денного навчання; мережного тестування; викладання за окрему платню додаткових дисциплін за вибором студента (крім офіційно затвердженої програми навчання); післядипломного навчання (дистанційні курси підвищення кваліфікації); залучення викладачів (з інших ВНЗ, міст і країн). Система «Прометей» має модульну архітектуру, тому вона легко розширюється, модернізується і масштабується. На сучасному етапі у СДН «Прометей» автоматично здійснено доступ до безкоштовних Інтернет-сервісів Microsoft Live@Edu. Причому доступ до цих сервісів здійснюється безпосередньо з інтерфейсу СДН «Прометей». Така інтеграція значно розширює можливості для спільної роботи слухачів і викладачів, одночасно суттєво знижуючи витрати на мережеву інфраструктуру [4].

Система диференційованого Інернет-навчання ГЕКАДЕМ. Ця система розроблена у Байкальській міжнародній бізнес школі Іркутського державного університету. Вона дозволяє забезпечити сучасний рівень освіти в навчальних закладах і навчальних центрах на основі застосування сучасних телекомунікацій та інформаційних технологій. Засобами системи ГЕКАДЕМ

можна ефективно вирішити завдання управління освітньою діяльністю, розробки адаптивних навчальних курсів, їх розвитку, індивідуального контрольованого навчання й аналізу навчального процесу. Система побудована на основі авторської моделі подання знань навчального курсу, яка дозволяє організувати диференційоване, індивідуальне навчання Інтернет-технології. ГЕКАДЕМ може використовуватися не тільки для організації дистанційного навчання, але й бути інструментом підвищення ефективності класичного денного навчання. ГЕКАДЕМ забезпечує надійну роботу наступних груп користувачів: викладачі - розробники Інтернеткурсів; студенти, що навчаються на конкретних курсах; викладачі, що супроводжують курс; керівники й адміністрація освітньої установи; системний адміністратор. Для кожного користувача система підтримує авторизований доступ до своїх ресурсів відповідно до його повноважень. Система ГЕКАДЕМ дає можливість керівництву освітньої установи реалізувати свою політику в галузі Інтернет-навчання, а саме:

- готувати навчальні програми для спеціальностей і спеціалізацій;
- визначати перелік вимог до навчальних курсів;
- будувати навчальні плани для кожної програми
- вести роботу з розробниками курсів
- організувати процес навчання відповідно до навчальних планів.

У системі Гекадем існує 4 підсистеми:

- деканат дає можливість керівництву освітнього закладу реалізувати свою політику в галузі дистанційного навчання.

- конструктор курсів, організований для викладача - розробника курсу, в якій формується структура навчального курсу з навчальних блоків, розміщується навчальний матеріал у відповідній формі: текст, графіка, звук, відео, гіпертекст, ігри і т.п.

- тьютор дає можливість викладачу, який супроводжує курс (тьютору) контролювати процес вивчення курсу кожного студента, оцінювати виконані ним індивідуальні завдання, його роботу на семінарах і при необхідності надавати йому допомогу або давати пораду.

- студент дозволяє студенту вибрати для себе найбільш зручний шлях вивчення і працювати в індивідуальному режимі в зручний для себе час.

Мережна освітня платформа (МОП) e-University. МОП eUniversity призначена для навчання і тестування з використанням сучасних інформаційних технологій. Вона забезпечує вирішення наступних завдань:

- авторизований доступ до ресурсів МОП;
- управління зареєстрованими користувачами;

- створення навчальних курсів;
- підготовка та надання навчальних матеріалів учням;
- надання засобів комунікації;
- тестування рівня знань;
- моніторинг результатів тестування;
- контроль організації навчання;
- захист інформації;

МОП e-University можна застосовувати з метою:

- навчання та тестування студентів вищих навчальних закладів із використанням сучасних навчальних засобів;
- організації дистанційного навчання в очному і заочному навчанні;
- перепідготовки кадрів на базі випускаючих кафедр;
- довузівської підготовки і тестування;
- організації відкритого дистанційного навчання та платного навчання на додаткових курсах;

Платформа e-University може використовуватися як безпосередньо для дистанційного навчання, коли учні фізично віддалені від центру навчання, підписані на певний набір навчальних курсів, отримують методичні вказівки, виконують тести і завдання, так і для очних форм навчання. eUniversity надає необхідний набір засобів для реалізації концепції дистанційного навчання, взаємодії управління, викладання і навчання. Адміністратор системи керує навчальним процесом, налаштовує систему на структуру установи, створює бібліотеку навчальних ресурсів, керує користувачами і групами, курсами, інформаційними потоками. Викладач розробляє власні навчальні курси: лекції з розділів, електронні підручники, аудіо-відео матеріали, систему тестування й оцінки знань. Він підтримує тісний контакт зі слухачами за допомогою електронної пошти, onlineконсультацій і дошок оголошень, виявляє переваги, рекомендує додаткові навчальні ресурси, аналізує результати тестування. Слухач самостійно вивчає курс, взаємодіючи з викладачем. У його розпорядженні знаходяться всі інформаційні ресурси: навчальні плани, лекції, електронні підручники, тести й їх статистика, рейтинги. Він отримує рекомендації викладача, консультується з ним, веде листування, проходить контрольні та навчальні тести у зручний для нього час [6].

Веб-система дистанційного навчання корпоративного рівня «Віртуальний університет». «Віртуальний університет» є однією з СДН вітчизняного походження, що призначена для вирішення завдань із дистанційного навчання у компаніях і навчальних закладах України будь-

яких масштабів і рівнів. Розроблена за участі професорського складу кафедри АСОІУ ФІОТ НТУУ «КПІ», система відповідає вимогам Міністерства освіти і науки України та має розманітні можливості для тестування учнів, обліку та розвитку курсів, подання статистики і формування необхідних звітів. Основними перевагами даної системи є: незначні витрати на встановлення та обслуговування системи дистанційного навчання; швидкість і висока якість надання/викладання навчальних матеріалів; зручний і ефективний рівень оцінювання засвоєних знань і виконання самостійних робіт; залучення більшої кількості абітурієнтів; зручний інтерактивний інтерфейс; доступ до системи з мережі Інтернет дозволяє брати участь у навчальному процесі з будь-якого куточку світу; збільшення конкурентоздатності навчального закладу та застосування в роботі новітніх сучасних інформаційних технологій. «Віртуальний Університет» складається з таких додаткових модулів: Бібліотека, Статистика, Синхронізація, Публікатор, Пошук, Менеджер розкладів, Аналітичний модуль, Резервування даних. Завдяки цим модулям можна автоматизувати будь-який навчальний процес. Варіанти комплектації дозволяють створити віртуальне середовище будь-яких масштабів: група / клас; потік / кафедра; факультет / підприємство; університет / корпорація. СДН «Віртуальний Університет» дозволяє реалізувати будь-які напрямки у освітньому процесі: починаючи від створення внутрішнього комунікаційного середовища до повноцінного віртуального університету у всесвітній мережі Інтернет [7].

Віртуальне навчальне середовище - «Веб-клас-ХПІ». Ця СДН розроблена проблемною лабораторією дистанційного навчання НТУ «ХПІ» і призначена для створення динамічного інформаційного простору, який має на меті забезпечувати продуктивну навчальну діяльність і враховувати всі пізнавальні потреби слухачів, а саме: презентацію структурованих і мотивованих навчальних матеріалів; підтримку пізнавальної і діяльнісної активності користувачів; необхідну комунікацію і співробітництво учасників навчального процесу у різних формах; засоби адміністрування навчального процесу та його активного супроводження; система має динамічно настроювану мову інтерфейсу (українську, російську або англійську). До складу «Веб-клас-ХПІ» входять наступні підсистеми: доступу до довідкової інформації (передмова курсу, відомості про авторів і тьюторів); адміністрування (реєстрація нових студентів, контроль активності); доступу до базових інформаційних матеріалів - курсу; доступу до динамічно сформованих навчальних матеріалів через курс- меню; підготовки і проведення он-лайн тестування (включає тестування знань, адаптивне і

психологічне тестування, анкетування, інтерв'ю); доступу до системних веб-словників; внутрішньокурсової пошти; форуму - курсового дискусійного клубу; рядків чатів.

У «Веб-клас-ХПІ»є кілька груп користувачів. Адміністратор системи має найвищий пріоритет для доступу до ресурсів системи. Його діяльність спрямована на підтримку цілісності системи, збереження баз даних, здійснення загального контролю за навчальним процесом. Він контролює процес реєстрації слухачів, відстежує можливі порушення правил роботи в даній системі. Автор курсу створює дистанційний курс, який включає інформаційні матеріали, набір тестів, теми для обговорення у Форумі та Рядку чатів. Система надає авторові можливість формувати частину занять динамічно, якщо цього вимагає індивідуальний підхід до окремого студента або групи студентів. Тьютор відповідає за успішне проведення дистанційного навчання. Він відстежує наповнення баз даних курсу слухачами і, у разі потреби, очищає частину записів. Тьютор контролює хід Форуму, організовує і проводить чати. При необхідності, він може створювати динамічні уроки. Слухачі є тими, хто записується на вивчення дистанційного курсу. Перша версія системи «Веб-Клас-ХПІ» була використана в травні-вересні 2001 року для організації та проведення дистанційного курсу «Практичний курс дистанційного навчання». З курсами, які реалізовані в даному середовищі, можна ознайомитися на сайті університету (http://dl.kpi.kharkov.ua) [8].

Отже, описані СДН мають спільну мету - програмне забезпечення процесу дистанційного навчання, проте мають різні параметри та можливості.

1.1 Технічне завдання

Функціональні вімоги до серверної частини:

- віділення ролей студнет, викладач, адміністратор;
- делегування доступу;
- реестер користувачів;
- захист курсів від плагіату;
- інтеграція покупок;
- система сертифікації;
- система курсів;
- категерування курсів за змістом;

- єдиний формат обміну данних(JSON - JavaScript Object Notation);

- асинхронна робота з базою данних;

- реалізація COMET концепції;

- підсистеми повинні реалізовувати концепцію CRUD (Create Update Delete)

- підтримка HTTP та HTTPS;

Функціональні вимоги до WEB - кліенту:

- реальзація зовнішного вигляду у стилі Google Material Design;

- гнучкість;

- реальзація усього функціоналу серверної частини;

- підтримка WEB - оглаядачів останніх версій Safari, Chrome, Opera, Firefox;

Функціональні вимоги до Android кліенту:

- Підтримка OS Android версії 4.4.2 та вище;

Функціональні вимоги до iOS кліенту:

- Підтримка iOS версії 9.0 та вище;

2 ХАРАКТЕРИСТИКА ПРОГРАМНОГО СЕРЕДОВИЩА

2.1 Мова программування Java

Java (вимовляється Джава; інколи — Ява) — об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. З 2009 року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems». В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

«Oracle» надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцензією GNU General Public License.

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що в порівнянні, наприклад, з C++ зменшує швидкість роботи програм. За необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування.

Java вплинула на розвиток J++, що розроблялась компанією «Microsoft». Роботу над J++ було зупинено через судовий позов «Sun Microsystems», оскільки ця мова програмування була модифікацією Java. Пізніше в новій платформі «Microsoft» .NET випустили J#, щоб полегшити міграцію програмістів J++ або Java на нову платформу. З часом нова мова програмування C# стала основною мовою платформи, перейнявши багато чого з Java. J# востаннє включався в версію Microsoft Visual Studio 2005. Мова сценаріїв JavaScript має схожу із Java назву і синтаксис, але не пов'язана із Java.

Спочатку мова називалася Oak («дуб») і розроблялася Джеймсом Гослінгом для програмування побутових електронних пристроїв. Згодом вона була перейменована в Java і стала використовуватися для написання клієнтських застосунків і серверного програмного забезпечення. Названа на честь марки кави Java, яка, в свою чергу, отримала найменування.

Однойменного острова (Ява), тому на офіційній емблемі мови зображена чашка з паркою кавою. Існує й інша версія походження назви мови, пов'язана з алюзією на кава-машину як приклад побутового влаштування, для програмування якого спочатку мова створювалася. Мова програмування Java зародилася в 1991 р. в лабораторіях компанії Sun Microsystems. Розробку проекту започаткував Джеймс Ґослінґ, сам проект мав назву «Green» (Зелений). Створення першої робочої версії, яка мала назву «Oak» (дуб), зайняло 18 місяців. Оскільки виявилось, що ім'я Oak уже використовувалось іншою фірмою, то в результаті тривалих суперечок навколо назви нової мови з-поміж ряду запропонованих було вибрано назву Java, у 1995 р. мову було офіційно перейменовано.

Головним мотивом створення Java була потреба в мові програмування, яка б не залежала від платформи (тобто від архітектури) і яку можна було б використовувати для створення програмного забезпечення, що вбудовується в різноманітні побутові електронні прилади, такі як мобільні засоби зв'язку, пристрої дистанційного керування тощо.

Досить скоро майже всі найпопулярніші тогочасні веб-оглядачі отримали можливість запускати «безпечні» для системи Java-аплети всередині веб-сторінок. У грудні 1998 р. Sun Microsystems випустила Java 2 (спершу під назвою J2SE 1.2), де було реалізовано декілька конфігурацій для різних типів платформ. Наприклад, J2EE призначалася для створення корпоративних застосунків, а значно урізана J2ME для приладів з обмеженими ресурсами, таких як мобільні телефони. У 2006 році в маркетингових цілях версії J2 було перейменовано у Java EE, Java ME та Java SE відповідно.

13 листопада 2006 року Sun випустили більшу частину Java як вільне та відкрите програмне забезпечення згідно з умовами GNU General Public License (GPL). 8 травня 2007 корпорація закінчила процес, в результаті якого всі початкові коди Java були випущенні під GPL, за винятком невеликої частини коду, на який Sun не мала авторського права.

Період становлення Java збігся у часі з розквітом міжнародної інформаційної служби World Wide Web. Ця обставина відіграла вирішальну роль у майбутньому Java, оскільки Web теж вимагала платформо-незалежних програм. Як наслідок, були зміщені акценти в розробці Sun з побутової електроніки на програмування для Інтернет.

У створенні мови програмування Java було чотири початкові цілі:

- Синтаксис мови повинен бути «простим, об'єктно-орієнтовним та звичним»;

- Реалізація має бути «безвідмовною та безпечною»;

- Повинна зберегтися «незалежність від архітектури та переносимість»;

- Висока продуктивність виконання;

- Мова має бути «інтерпретованою, багатонитевою, із динамічним зв'язуванням модулів».

Під «незалежністю від архітектури» мається на увазі те, що програма, написана на мові Java, працюватиме на будь-якій підтримуваній апаратній чи системній платформі без змін у початковому коді та перекомпіляції.

Цього можна досягти, компілюючи початковий Java код у байт-код, який є спрощеними машинними командами. Потім програму можна виконати на будь-якій платформі, що має встановлену віртуальну машину Java, яка інтерпретує байткод у код, пристосований до специфіки конкретної операційної системи і процесора. Зараз віртуальні машини Java існують для більшості процесорів і операційних систем.

Основна перевага використання байт-коду — це портативність. Тим не менш, додаткові витрати на інтерпретацію означають, що інтерпретовані програми будуть майже завжди працювати повільніше, ніж скомпільовані у машинний код, і саме тому Java одержала репутацію «повільної» мови. Проте, цей розрив суттєво скоротився після введення декількох методів оптимізації у сучасних реалізаціях JVM.

Одним із таких методів є just-in-time компіляція (JIT, що перетворює байт-код Java у машинний під час першого запуску програми, а потім кешує його. У результаті така програма запускається і виконується швидше, ніж простий інтерпретований код, але ціною додаткових витрат на компіляцію під час виконання. Складніші віртуальні машини також використовують динамічну рекомпіляцію, яка полягає в тому, що віртуальна машина аналізує поведінку запущеної програми й вибірково рекомпілює та оптимізує певні її частини. З використанням динамічної рекомпіляції можна досягти більшого рівня оптимізації, ніж за статичної компіляції, оскільки динамічний компілятор може робити оптимізації на базі знань про довкілля періоду виконання та про завантажені класи. До того ж він може виявляти так звані гарячі точки (англ. hot spots) — частини програми, найчастіше внутрішні цикли, які займають найбільше часу при виконанні. JIT-компіляція та динамічна рекомпіляція збільшує швидкість Java-програм, не втрачаючи при цьому портативності.

Існує ще одна технологія оптимізації байткоду, широко відома як статична компіляція, або компіляція ahead-of-time (AOT). Цей метод передбачає, як і традиційні компілятори, безпосередню компіляцію.

Це забезпечує хороші показники в порівнянні з інтерпретацією, але за рахунок втрати переносності: скомпільовану таким способом програму можна запустити тільки на одній, цільовій платформі.

Швидкість офіційної віртуальної машини Java значно покращилася з моменту випуску ранніх версій, до того ж, деякі випробування показали, що продуктивність JIT-компіляторів у порівнянні зі звичайними компіляторами у машинний код майже однакова. Проте ефективність компіляторів не завжди свідчить про швидкість виконання скомпільованого коду, тільки ретельне тестування може виявити справжню ефективність у даній системі. На противагу C++, Java об'єктно-орієнтованіша. Всі дані і дії групуються в класи об'єктів. Виключенням з повної об'єктності (як скажімо в Smalltalk) є примітивні типи (int, float тощо). Це було свідомим рішенням проектувальників мови задля збільшення швидкості. Через це Java не вважається повністю об'єктно-орієнтовною мовою.

У Java всі об'єкти є похідними від головного об'єкта (він називається просто Object), з якого вони успадковують базову поведінку і властивості.

Хоча у C++ вперше стало доступне множинне успадкування, але у Java можливе тільки одинарне успадкування, завдяки чому виключається можливість конфліктів між членами класу (методи і змінні), які успадковуються від базових класів. У намірах проектувальників Java мала замінити C++ — об'єктного наступника мови C. Проектувальники почали з аналізу властивостей C++, які є причиною найбільшого числа помилок, щоби створити просту, безпечну і безвідмовну мову програмування.

В Java існує система винятків або ситуацій, коли програма зустрічається з неочікуваними труднощами, наприклад:

- операції над елементом масиву поза його межами або над порожнім елементом;
- читання з недоступного каталогу або неправильної адреси URL;
- ввід недопустимих даних користувачем;

Одна з особливостей концепції віртуальної машини полягає в тому, що помилки (виключення) не призводять до повного краху системи. Крім того, існують інструменти, які «приєднуються» до середовища періоду виконання і кожен раз, коли сталося певне виключення, записують інформацію з пам'яті для відлагодження програми. Ці інструменти автоматизованої обробки виключень надають основну інформацію щодо виключень в програмах на Java.

Проте мову програмування Java не рекомендується використовувати в системах, збій в роботі яких може призвести до смерті, травм чи значних

фізичних ушкоджень (наприклад, програмне забезпечення для керування атомними електростанціями, польотами, систем життєзабезпечення чи систем озброєння) через ненадійність програм, написаних на мові програмування Java, пункт ліцензії Microsoft 7.7.h.

Java використовує автоматичний збирач сміття для керування пам'яттю під час життєвого циклу об'єкта. Програміст вирішує, коли створювати об'єкти, а віртуальна машина відповідальна за звільнення пам'яті після того, як об'єкт стає непотрібним. Коли до певного об'єкта вже не залишається посилань, збирач сміття може автоматично прибирати його із пам'яті. Проте, витік пам'яті все ж може статися, якщо код, написаний програмістом, має посилання на вже непотрібні об'єкти, наприклад на об'єкти, що зберігаються у діючих контейнерах.

Збирання сміття дозволене у будь-який час. В ідеалі воно відбувається під час бездіяльності програми. Збірка сміття автоматично форсується при нестачі вільної пам'яті в купі для розміщення нового об'єкта, що може призводити до кількасекундного зависання. Тому існують реалізації віртуальної машини Java з прибиральником сміття, спеціально створеним для програмування систем реального часу. Java не має підтримки вказівників у стилі C/C++. Це зроблено задля безпеки й надійності, аби дозволити збирачу сміття переміщувати вказівникові об'єкти.

Java-програми записуються в Юнікоді, також надається лексичне перетворення, яке дозволяє записувати символи Юнікоду керівними кодами Unicode за допомогою лише множини символів ASCII. Мова Java представляє текст послідовностями 16-бітних кодових одиниць, використовуючи кодування UTF-16. За винятком коментарів, ідентифікаторів та вмісту символьних та рядкових літералів, всі вхідні елементи програми на Java складаються із символів ASCII або відповідних їм керівних кодів Unicode.

Java є суворо типізованою мовою, кожна змінна та вираз має тип, відомий на етапі компіляції.

Типи даних Java належать до двох категорій: прості (primitive) та вказівникові (reference). До простих типів належить булевий(логічний) тип, числові типи та символьний тип.

Числові типи складаються із цілих типів byte, short, int, long та дійсних типів float, double. Символьний тип представлений типом char. Вказівникові типи складаються із класів, інтерфейсів, масивів. Значенням вказівникового типу є вказівник на об'єкт — екземпляр класу чи масиву. Рядки є об'єктами класу String.

2.2 Мова програмування JavaScript

JavaScript (JS) — динамічна, об'єктно-орієнтована мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується як частина браузера, що надає можливість коду на стороні клієнта (такому, що виконується на пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки. Мова JavaScript також використовується для програмування на стороні сервера (подібно до таких мов програмування, як Java і C#), розробки ігор, стаціонарних та мобільних додатків, сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite), всередині PDF-документів тощо.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Незважаючи на схожість назв, мови Java та JavaScript є двома різними мовами, що мають відмінну семантику, хоча й мають схожі риси в стандартних бібліотеках та правилах іменування. Синтаксис обох мов отриманний «у спадок» від мови C, але семантика та дизайн JavaScript є результатом впливу мов Self та Scheme.

1995 року компанія Netscape поставила завдання вбудувати мову програмування Scheme чи «якусь схожу» в браузер Netscape. Для цього був запрошений Брендан Айк, американський розробник, що спеціалізувався на системному програмуванні. Також, для прискорення розробки, Netscape почали співробітництво з компанією Sun Microsystems.

З часом, концепція розроблюваної мови програмування була розширена до можливості використання безпосередньо в HTML-коді сторінки. Компанії мали на меті створити мову, що могла зв'язати різні частини веб-сайтів: зображень, Java-аплетів, об'єктної моделі документа. Ця мова повинна була стати зручною для веб-дизайнерів та некваліфікованих програмістів. Робочою назвою нової мови була Mocha, яка була змінена на LiveScript в перших двох бета-версіях браузера Netscape 2.0. А дещо пізніше, користуючись популярністю бренду Java, LiveScript був перейменований на JavaScript і третя бета-версія (2.0B3) Netscape 2.0 вже вийшла з сучасною

назвою. Для цього була придбана відповідна ліцензія у компанії Sun Microsystems, що володіла брендом Java.

1992 року компанією Nombas була розроблена скриптова мова програмування Cmm (англ. C-minus-minus, гра слів навколо мови C++), яка пізніше була перейменована на ScriptEase та могла вбудовуватися в веб-сторінки. Існує хибна думка, що JavaScript був створений під впливом Cmm. Насправді, Брендан Айк ніколи не чув про Cmm до того, як він створив LiveScript. Пізніше, Nombas зупинили розробку Cmm та почали використовувати JavaScript, а згодом брали участь у групі зі стандартизації JavaScript.

У листопаді 1996 року Netscape заявила, що відправила JavaScript в організацію Ecma International для розгляду мови як промислового стандарту. В результаті подальшої роботи з'явилась стандартизована мова з назвою ECMAScript. У червні 1997 року, Ecma International опублікувала першу редакцію специфікації ECMA-262. Рік по тому, у червні 1998 року, щоб адаптувати специфікацію до стандарту ISO/IEC-16262, були внесені деякі зміни і випущена друга редакція. Третя редакція побачила світ в грудні 1999 року.

Четверта версія стандарту ECMAScript так і не була закінчена і четверта редакція не вийшла. Тим не менш, п'ята редакція з'явилася в грудні 2009 року.

На сьогодні, актуальна версія стандарту — 5.1. Вона була випущена в червні 2011 року. JavaScript, наразі, є однією з найпопулярніших мов програмування в інтернеті. Але спочатку багато професіональних програмістів скептично ставилися до мови, цільова аудиторія якої складалася з програмістів-любителів. Поява AJAX змінила ситуацію та повернула увагу професійної спільноти до мови.

JavaScript має низку властивостей об'єктно-орієнтованої мови, але завдяки концепції прототипів підтримка об'єктів в ній відрізняється від традиційних мов ООП. Крім того, JavaScript має ряд властивостей, притаманних функціональним мовам, — функції як об'єкти першого класу, об'єкти як списки, каррінг, анонімні функції, замикання (closures) — що додає
мові додаткову гнучкість.

JavaScript має С-подібний синтаксис, але в порівнянні з мовою Сі має такі корінні відмінності:

- об'єкти, з можливістю інтроспекції і динамічної зміни типу через механізм прототипів;

- функції як об'єкти першого класу;
- обробка винятків;
- автоматичне приведення типів;
- автоматичне прибирання сміття;
- анонімні функції.

JavaScript містить декілька вбудованих об'єктів: Global, Object, Error, Function, Array, String, Boolean, Number, Math, Date, RegExp. Крім того, JavaScript містить набір вбудованих операцій, які, строго кажучи, не обов'язково є функціями або методами, а також набір вбудованих операторів, що управляють логікою виконання програм. Синтаксис JavaScript в основному відповідає синтаксису мови Java (тобто, зрештою, успадкований від C), але спрощений порівняно з ним, щоб зробити мову сценаріїв легкою для вивчення. Так, приміром, декларація змінної не містить її типу, властивості також не мають типів, а декларація функції може стояти в тексті програми після неї

Семантика мови схожа з семантикою мови Self.

При використанні в рамках технології DHTML JavaScript код включається в HTML-код сторінки і виконується інтерпретатором, вбудованим в браузер. Код JavaScript вставляється в теги <script></script> з обов'язковим по специфікації HTML 4.01 атрибутом type="text/javascript", хоча в більшості браузерів мова сценаріїв за умовчанням саме JavaScript.

При розробці великих і нетривіальних веб-застосунків з використанням JavaScript, критично важливим є доступ до інструментів зневадження. Оскільки браузери від різних виробників дещо відрізняються у поведінці JavaScript і реалізації Об'єктної Моделі Документа, треба мати в руках зневаджувач для кожного браузера, якщо веб-застосування орієнтовано на нього.

На даний час Internet Explorer, Firefox, Opera, Google Chrome та Safari мають зневаджувачі для себе.

Internet Explorer має три зневаджувача для себе: Microsoft Visual Studio є найпотужнішим з цих трьох, слідом йде Microsoft Script Editor (компонента Microsoft Office), і нарешті існує безкоштовний Microsoft Script Debugger з базовими функціями. Веб-застосування для Firefox допоможе привести до розуму додаток Firebug (зручно вбудований безпосередньо в браузер), або давніший відладчик Venkman, котрий також працює з браузером Mozilla. Drosera — це зневаджувач з WebKit engine, що супроводжує Apple Safari.

Також існують кілька інструментів, як вільних, наприклад JSLint,

інструмент перевірки якості коду, що сканує JavaScript програму, шукаючи проблеми коду, так і комерційних продуктів типу інструменту з назвою JavaScript Debugger.

Оскільки JavaScript є інтерпретатором, без строгої типізації, і може виконуватися в різних середовищах, кожне зі своїми власними особливостями сумісності, програміст має бути дуже уважним, і повинен перевіряти, що його код виконується як очікується в широкому переліку можливих конфігурацій. Дуже часто трапляються випадки, коли скрипт, що чудово працює в одному середовищі, видає некоректні результати в іншому.

Кожен блок сценарію інтерпретатор розбирає окремо. На веб-сторінках, коли треба комбінувати блоки JavaScript та HTML, синтаксичні помилки знайти легше, якщо тримати функції сценарію в окремому блоці коду, або (ще краще) використовувати багато малих пов'язаних .js файлів. В такий спосіб синтаксична помилка не спричинятиме «падіння» цілої сторінки, і можна надати допомогу, елегантно вийшовши зі сторінки.

2.3 Мова програмування Objective-C

Objective-C — рефлективна, високорівнева об'єктно-орієнтована мова програмування загального призначення, розроблена у вигляді набору розширень стандартної C.

Розроблена компанією Apple, використовується в основному у Mac OS X та GNUStep — середовищах, розроблених на основі стандарту OpenStep, та Cocoa — бібліотеки компонентів для розробки програм. Програму на Objective-C що не використовує цих бібліотек можна скомпілювати для будь-якої платформи, яку підтримує gcc компілятор з підтримкою Objective-C.

Objective-C є розширенням C і тому будь-яку програму на C можна скомпілювати компілятором Objective-C.

ООП в Objective-C включає інтерфейси, класи, категорії. Реалізовано одиничне, невіртуальне спадкування. Немає єдиного базового класу для всіх об'єктів. Всі методи в класі — віртуальні. Категорія — парадигма яка дозволяє описувати інтерфейс з методами які «необов'язково» імплементувати.

Синтакс Objective-C породжений одночасно від C та Smalltalk. Від останньої взято основний семантичний конструкт мови — замість виклику методу об'єктові надсилається повідомлення.

На початку 1980-х років було популярним структурне програмування.

Воно дозволяло «розбивати» алгоритм на малі частини, в основному щоб виділити етапи алгоритму в окремі блоки і працювати з ними. Однак, із зростанням складності завдань, структурне програмування призводило до зниження якості коду. Доводилося писати все більше функцій, які дуже рідко могли використовуватися в інших програмах.

З іншого - використання віртуальних машин сильно гальмувало роботу системи і вимагало величезних ресурсів. ObjC був створений Бредом Коксом на початку 1980-х в його компанії Stepstone. Він намагався вирішити проблему повторного використання коду.

Метою Кокса було створення мови, яка підтримуватиме концепцію software IC. Під цією концепцією розуміється можливість збирати програми з готових компонентів (об'єктів), подібно до того як складні електронні пристрої можуть бути легко зібрані з набору готових інтегральних мікросхем. При цьому така мова має бути досить простою і заснованою на мові С, щоб полегшити перехід розробників на неї. Однією з цілей було також створення моделі, в якій самі класи також є повноцінними об'єктами і підтримувалася б інтроспекція і динамічна обробка повідомлень.

Отримана в результаті мова Objective-C виявилась вкрай простою - на її "освоєння" у С-програміста піде всього кілька днів. Вона є саме розширенням мови С - в мову С просто додані нові можливості для об'єктно-орієнтованого програмування. При цьому будь-яка програма на С є програмою і на Objective-C.

Однією з відмінних рис Objective-C є її динамічність - цілий ряд рішень, які зазвичай приймаються на етапі компіляції, тут відкладаються безпосередньо до етапу виконання. Ще однією особливістю мови є те, що вона message-oriented в той час як C++ - function-oriented. Це означає, що в ній виклики методу інтерпретуються не як виклик функції (хоча до цього, зазвичай, все зводиться), а саме як відправлення повідомлення (з ім'ям і аргументами) об'єкту, подібно до того, як це відбувається в Smalltalk. Такий підхід дає цілий ряд плюсів - будь-якому об'єкту можна послати будь-яке повідомлення. Об'єкт може замість обробки повідомлення просто переслати його іншому об'єкту для обробки (так зване делегування), зокрема саме так можна легко реалізувати розподілені об'єкти (тобто об'єкти, що знаходяться в різних адресних просторах і навіть на різних комп'ютерах). Прив'язка повідомлення до відповідної функції відбувається безпосередньо на етапі виконання.

Мова Objective-C підтримує нормальну роботу з метаінформацією - так у об'єкта безпосередньо на етапі виконання можна запитати його клас,

список методів (з типами переданих аргументів) і instance-змінних, перевірити, чи є клас нащадком заданого і чи підтримує він заданий протокол і т. ін. У мові є нормальна підтримка протоколів (тобто поняття інтерфейсу об'єкта та протоколу чітко розділені). Для об'єктів підтримується успадкування (не множинне), для протоколів підтримується множинне спадкування. Об'єкт може бути успадкований від іншого об'єкта і відразу декількох протоколів (хоча це скоріше не успадкування протоколу, а його підтримка). На даний момент мова Objective-C підтримується компіляторами Clang і GCC (під управлінням Windows Windows використовується у складі MinGW або cygwin). Досить багато в мові перенесено на runtime-бібліотеку і сильно залежить від неї. Разом з компілятором gcc поставляється мінімальний варіант такої бібліотеки. Також можна вільно завантажити runtime-бібліотеку компанії Apple: Apple's Objective-C runtime.

2.4 Розмітка HTML5

HTML5 — наступна версія мови HTML. До складу робочої групи з HTML5 увійшли AOL, Apple, Google, IBM, Microsoft, Mozilla, Nokia, Opera та кілька сотень інших виробників.

Існує деяка плутанина щодо версійності, оскільки існують дві незалежні групи розробників — WHATWG та W3C.
WHATWG відмовились від принципу «версійності», в користь «вічної розробки» при прийнятті HTML специфікації. Таке рішення було спричинено намаганням пришвидшити втілення стандарту в життя, тобто розробникам веб браузерів не потрібно чекати допоки вийде офіційна затверджена версія специфікації (специфікація перейде в стан recomendation), вони можуть втілювати певні частини специфікації вже зараз. Тому за версією WHATWG існує тільки одна специфікація, яка постійно розвивається — HTML.

Ці дві групи працювали в тандемі, WHATWG писав специфікації в режимі «живого стандарту», а W3C брав ці специфікації як «знімки», й впроваджував їх у чіткі версії своєї специфікації. W3C працював значно повільніше, бо повинен забезпечувати вимоги більшого спектра користувачів, а не тільки веб-браузерів.

28 жовтня 2014 консорціум W3C оголосив про надання набору специфікацій HTML5 статусу рекомендованого стандарту. Цікаво, що у цьому вигляді специфікації HTML 5.0 були сформовані ще два роки до того, після чого робота була зосереджена на проведенні тестування та оцінки

сумісності доступних реалізацій. На час стандартизації HTML5 вже давно став стандартом де-факто і активно використовується у веб-застосунках. Фактичне затвердження стандарту лише формально поставило крапку в просуванні HTML5 і підтвердило повсюдність і коректність його реалізації.

Специфікації HTML5 не обмежуються тільки розміткою і включають в себе низку веб-технологій, котрі у сукупності формують відкриту Веб-платформу — програмне оточення для роботи крос-платформових застосунків, здатних взаємодіяти з обладнанням, і які підтримують засоби для роботи з відео, графікою і анімацією, що надає розширені мережеві можливості.

Робота над HTML5 почалася в кінці 2003 року як доказ концепції, що можна розширити форми HTML4 багатьма можливостями з XForms 1.0, не вимагаючи змін несумісних з існуючими веб-сторінками. На цій ранній стадії, хоча проект уже був доступний для громадськості, й приймав критичні зауваження, специфікація була захищена авторськими правами Opera.

Невдовзі після цього, Apple, Mozilla та Opera оголосили про свій намір спільно продовжити роботу. Було створено відкритий список розсилки й проект було перенесено на сайт WHATWG. В авторське право була внесена поправка, що робота буде спільно належати всім трьом постачальникам і допускатиме повторне використання специфікацій.

У 2006 році W3C висловило зацікавленість у специфікації, а також створила робочу групу для роботи разом із WHATWG над розробленням специфікації HTML5. Робоча група відкрита в 2007 році. Apple, Mozilla та Opera дозволили W3C опублікувати специфікацію під ліцензією W3C, зберігаючи при цьому версію з менш обмежувальною ліцензію на сайті WHATWG. З того часу обидві групи працюють разом.

22 січня 2008 представлена чернетка чергової, п'ятої версії мови гіпертекстової розмітки HTML.

Робота над HTML5, що почалася у 2004 році, зараз доводиться до кінця спільними зусиллями W3C HTML WG і WHATWG. У розробленні також беруть участь багато ключових фігур, включаючи представників найбільших розробників браузерів: Apple, Google, Microsoft, Mozilla та Opera; також інші організації й користувачі з різними інтересами та досвідом. У загальному з нової версії мови розмітки пропонується прибрати близько 15 тегів.

При прийнятті рішення про введення нових тегів було розглянуто більшість популярних сайтів і виділено основні елементи, які були спільними для всіх веб-сторінок.

Розмічаючи області на сторінці за допомогою певних елементів, ця

технологія може допомогти полегшити користувачеві навігацію. Наприклад, він може легко пропустити розділ навігації або швидко переходити від однієї статті до іншої без необхідності для авторів робити відповідні посилання. Автори також отримують вигоду в результаті заміни великої кількості div-ів одним з декількох відповідних елементів, що також приводить до чистого і легкого для автора початкового коду.

Елементами header є заголовки розділів. Вони можуть складатися з декількох частин — наприклад, було б виправдано розділяти блок заголовка на підзаголовки, історію версій або вказання авторства. Елемент footer визначає нижню частину розділу, до якого він відноситься. Зазвичай він містить інформацію про розділ — наприклад, ім'я автора, посилання на схожі документи, копірайт і тому подібне. Блок nav містить список посилань для навігації. Підходить, наприклад, для навігації по сайту, або для змісту. Елемент aside підходить для розміщення вмісту яким-небудь чином спорідненого основному контенту. У звичайному випадку буде корисний для розмітки бічної колонки. Тег section представляє загальний розділ документа або додатку, наприклад, такий як розділ. Тег article відзначає незалежний розділ документа, сторінки або сайту. Застосуємо для такого вмісту як новини, запису блога, повідомлення у форумі або коментарі користувачів.

2.5 Каскадні таблиці стилів CSS

Каскадні таблиці стилів (англ. Cascading Style Sheets або скорочено CSS) - спеціальна мова, що використовується для опису сторінок, написаних мовами розмітки даних.

Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів.

Специфікації CSS були створені та розвиваються Консорціумом Всесвітньої мережі.

CSS має різні рівні та профілі. Наступний рівень CSS створюється на основі попередніх, додаючи нову функціональність або розширюючи вже наявні функції. Рівні позначаються як CSS1, CSS2 та CSS3. Профілі - сукупність правил CSS одного або більше рівнів, створені для окремих типів пристроїв або інтерфейсів. Наприклад, існують профілі CSS для принтерів, мобільних пристроїв тощо.

CSS (каскадна або блочна верстка) прийшла на заміну табличній

верстці веб-сторінок. Головна перевага блочної верстки - розділення змісту сторінки (даних) та їхньої візуальної презентації. CSS використовується авторами та відвідувачами веб-сторінок, щоб визначити кольори, шрифти, верстку та інші аспекти вигляду сторінки. Одна з головних переваг - можливість розділити зміст сторінки (або контент, наповнення, зазвичай HTML, XML або подібна мова розмітки) від вигляду документу (що описується в CSS).

Таке розділення може покращити сприйняття та доступність контенту, забезпечити більшу гнучкість та контроль за відображенням контенту в різних умовах, зробити контент більш структурованим та простим, прибрати повтори тощо. CSS також дозволяє адаптувати контент до різних умов відображення (на екрані монітора, мобільного пристрою (КПК), у роздрукованому вигляді, на екрані телевізора, пристроях з підтримкою шрифту Брайля або голосових браузерах та ін.).

Один і той самий HTML або XML документ може бути відображений по-різному залежно від використаного CSS. Стилі для відображення сторінки можуть бути:

- стилі автора (інформація надана автором сторінки): - зовнішні таблиці стилів (англ. stylesheet), найчастіше окремий;

- файл або файли .css: внутрішні таблиці стилів, включені як частина документу або блоку;

- стилі для окремого елементу;

- стилі користувача: локальний .css-файл, вказаний користувачем для використання на сторінках і вказаний в налаштуваннях браузера (наприклад Opera);

- стилі переглядача (браузера): стандартний стиль переглядача, наприклад стандартні стилі для елементів, визначені браузером, використовуються коли немає інформації про стиль елемента або вона неповна;

2.6 Середовище для розробки IntelliJ IDEA

IntelliJ IDEA — комерційне інтегроване середовище розробки для різних мов програмування (Java, Python, Scala, PHP та ін.) від компанії JetBrains. Система поставляється у вигляді урізаної по функціональності безкоштовної версії «Community Edition» і повнофункціональної комерційної версії «Ultimate Edition», для якої активні розробники відкритих

проектів мають можливість отримати безкоштовну ліцензію. Сирцеві тексти Community-версії поширюються рамках ліцензії Apache 2.0. Двійкові збірки підготовлені для Linux, Mac OS X і Windows.

Перша версія IntelliJ IDEA з'явилася у січні 2001 року й швидко здобула популярність, як перша Java IDE із широким набором інтегрованих інструментів для рефакторингу, що дозволяла програмістам швидко реорганізовувати сирцевий код програм. Дизайн середовища орієнтовано на продуктивність праці програмістів, дозволяючи їм сконцентруватися на розробці функціональності, тоді як IntelliJ IDEA бере на себе виконання рутинних операцій.

Починаючи з шостої версії продукту IntelliJ IDEA надає інтегрований інструментарій для розробки графічного користувацького інтерфейсу.

З версії 9.0 є безкоштовний варіант Community Edition з відкритими кодами. Сирцеві коди відкритої версії IntelliJ IDEA Community Edition поширюються рамках ліцензії Apache 2.0. Бінарні пакунки підготовлені для Linux, Mac OS X і Windows.

До складу IntelliJ IDEA включені напрацювання, створені в результаті спільної роботи з компанією Google, яка використовувала IntelliJ IDEA як базис для своєї нового відкритого середовища розробки Android Studio. Завдяки співпраці істотно розширені штатні можливості IntelliJ IDEA з розробки застосунків для платформи Android.

Community версія середовища IntelliJ IDEA підтримує інструменти (у вигляді плагінів) для проведення тестування TestNG[en] і JUnit, системи контролю версій CVS, Subversion, Mercurial і Git, засоби складання Maven, Ant, Gradle, мови програмування Java, Scala, Clojure, Groovy і Dart. Підтримується розробка застосунків для мобільної платформи Android. До складу входить модуль візуального проектування GUI-інтерфейсу Swing UI Designer, XML-редактор, редактор регулярних виразів, система перевірки коректності коду, система контролю за виконанням завдань і доповнення для імпорту та експорту проектів з Eclipse. Доступні засоби інтеграції з системами відстеження помилок JIRA, Trac, Redmine, Pivotal Tracker, GitHub, YouTrack, Lighthouse.

Комерційна версія «Ultimate Edition» відрізняється наявністю підтримки додаткових мов програмування (наприклад, PHP, Ruby, Python, JavaScript, CoffeeScript, HTML, CSS, SQL), підтримкою технологій Java EE, UML-діаграм, підрахунок покриття коду, можливістю роботи з фреймворками (Rails, Grails, Google Web Toolkit, Spring, Play Framework і Hibernate), засобами інтеграції з Perforce[en], Microsoft Team Foundation.

2.7 Середовище для розробки Xcode

Xcode — інтегроване середовище розробки (IDE) виробництва Apple. Дозволяє створювати програмне забезпечення з використанням таких технологій як GCC, GDB, Java та ін. На сьогодні є єдиним засобом написання «універсальних»(Universal Binary) прикладних програм для Mac OS X.

Xcode включає в себе більшу частину документації розробника від Apple та Interface Builder — застосунок, який використовується для створення графічних інтерфейсів.

Пакет Xcode містить змінену версію вільного набору компіляторів GNU Compiler Collection і підтримує мови C, C++, Objective-C, Swift, Java, AppleScript, Python і Ruby з різними моделями програмування, включаючи (але не обмежуючись) Cocoa, Carbon і Java. Сторонніми розробниками реалізована підтримка GNU Pascal, Free Pascal, Ada, C #, Perl, Haskell і D. Пакет Xcode використовує GDB як back-end для свого відналагоджувача.

У серпні 2006 Apple оголосила про те, що DTrace, фреймворк динамічного трасування від Sun Microsystems, випущений як частина OpenSolaris, буде інтегрований в Xcode під назвою Xray. Пізніше Xray був перейменований в Instruments.

3 РОЗРАХУНКОВИЙ РОЗДІЛ

3.1 Серверна частина

Серверна частина уявляє собою коллекцію HTTP-сервлетів. Перелік сервлетів:

- FileController;
- Authorizator;
- TestController；
- CertificateController；
- CourseController;
- ThemeController；
- StatisticController;
- NotificationController;

FileController - контроллер який реагує на запити GET і POST. GET запит приймає параметр key і відправляє файл який має такий індетифікатор. Якщо не має файлу по заданаму key, або key порожній то відправляється статус 404 та картинка за замовченням. POST запит приймає key та файл у процессі виконання ассоціація key - file записуються у базу данних та файл записується на файлову систему серверу.

Authorizator - контроллер який реагує на запити GET, POST, ADD, TRACE, DELETE. GET запит повертає інформацію о поточної сесії авторизації. POST запит проводить операцію авторизації, приймаючи такі параметри login та хеш сумму від паролю (що важливо не пароль а хеш сумму) яка генерується кліентом і перевіряє дані у випадку вдалої авторизації серверна частина додає атрибут до сесії де записано роль користувача та привелегії користувача. У невдалому випадку авторизації повертається код - 1. ADD запит проводить операцію регістрації який приймає два набору параметрів це студент та викладач тип параметрів встановлюється автоматично, якщо вхідні данні недопустимі або не повні відправляється код 0 (перевірте вхідні данні), у випадку співпадання логіну або номеру телефону відправляється код -1 (телефон або логін зайняті) у випадку вдалої перевірки проводиться запис до бази данних користувачів і проводиться операція вдалої авторизації. TRACE цей запит перевіряє наявність логіну та номеру телефону у вже зареестрованих користувачів у випадку співдання телефону віправляється 1 (телефон зайнятий), якщо співдає логін то відправляється код 2 (логін зайнятий) якщо співпадають два

параметри то відпраляється код 3 (ці реєстраційні дані зайняті) і у випадку можливості реєстрації віправляється 10 (реєстрація можива). DELETE - операція виходу з сесії відправляє оперецію перенаправлення на головну сторінку. TestController - контроллер який реагує на запити GET і POST де GET це отримання курсів за параметрами а POST - це создання/оновлення курсу.

CertitficateController - це система отримання/видачі сертифікату реагує на запити GET і POST - де GET отримання сертифікату за параметрами, а POST створення сертифікату.

CourseController - це система роботи з курсами реагує на запити GET, POST, UPDATE, DELETE де GET - отримання курсів за параметрами, POST - додавання курсу, UPDATE - оновлення параметрів курсу та DELETE - видалення курсу.

ThemeContoller - це система роботи з темами курсу реагує на запити GET, POST, UPDATE, DELETE. GET - отримання тем курсу з контентом.POST - додавння тем до курсу. UPDATE - оновлення тем курсу. DELETE - видалення тем курсу.

StatisticController - сервлет статистики реагує на запити GET, POST, PUT. GET - отримання статистики користувача. POST - додавання елементу статистики. PUT - установка флагу “було просмотрено”.

NotificationController - сервлет повідомлень, реагує на запити GET, POST, TRACE, PUT. GET - отримання усіх повідомлень, POST - додавання повідомлення, TRACE - перевірка наявності нових повідомлень, PUT - установка флагу, було просмотренно.

Підключення до бази данних асинхронне, реалізоване за допомогою ORMLite. Де ORM - Object Relation Mapping. Реалізовно як нотування моделі та створення незалежних singletone-контроллерів.

Мова написання Java (розширення JAVA EE з Servlet API 3.0).

Рисунок 3.1 - приклад роботи FileServlet

3.2 WEB - кліент

WEB - кліент уявляє собою HTML5 розмітку з фреймворком JQuery та каскадними табицями стилів CSS.

Кліент має такі сторінки:

- index.html;
- sing_in.html;
- reg.html;
- certificate.html;
- course.html;
- course_viewer.html;
- create_course.html;
- create_theme.html;
- personal.html;
- create_test.html ;
- test.html;
- statistic.html;
- dashboard.html.

Перша сторінка це index.html - це перша сторінка яку бачить користувач
після завантаження сторінки. Ця сторінка має основну інформацію яка предоставлена в вигляді короткої справки о проекту та можливістю входу або реєстрації та просмотру курсу з привелегіями аноніма.

Sign_in.html - сторінка входу до особистого кабінету. Є можлівість переходу до головної сторінки та реєстрації.

Reg.html - сторінка реєстрації, яка предоставляє можливість реєстрації студента та викладача. Також є переходи на головну сторінку та сторінку входу до особистого кабінету.

Certificate.html - сторінка перевіки наявності сертифікату в базі данних для підверження справжності сертифікату, та можливості його відновлення у випадку підтверження отримання цього сертифікату.

Course.html - сторінка перегляду та пошуку курсів у вигляді маленьких карт. Є можливість продвиненого пошуку курсу за категоріями та влаштований автопошук після обиряння курсу курс відображається за допомогою course_viewer.html

Course_viewer.html - сторінка перегляду курсу. Відображення змінюється від ролі користувача та у випадку викладача в залежності від

авторства. Є можливість переглянути теми, додати курс, пройти курс або тестування та отримати/сформувати сертифікат.

Create_course.html - сторінка створення курсу. Є можливість створити курс, обрати час для проходження курсу, аватар курсу.

Create_theme.html - сторінка добавлення теми до матеріалу курсу. Є можливість додавання декількох тем та предперегляду.

Personal.html - сторінка особистого кабінету. Це основна сторінка користувача. З цієї сторінки можливо перевірити нові повідомлення, зміни статистики, створення або покупку курсу.

Create_test.html - сторінка створення тесту. Підтримуються два варіанти тестових питань - з варіантами відповідей та відкрите питання.

Test.html - сторінка проходження тестування. У процессі навчання студент повинен пройти тест який перевіряє його успішність у навчанні. Після проходження тесту створюється та зберігається сертифікат.

Statistic.html - сторінка статистики користувача. У викладача відображається його курсу, кількість прохорджень курсу, та запити на перевірку тестів. Також є рейтинг який генерується від успішность студентів. У студента це кількість придбаних курсів, пройдені курси, отримані сертифікати.

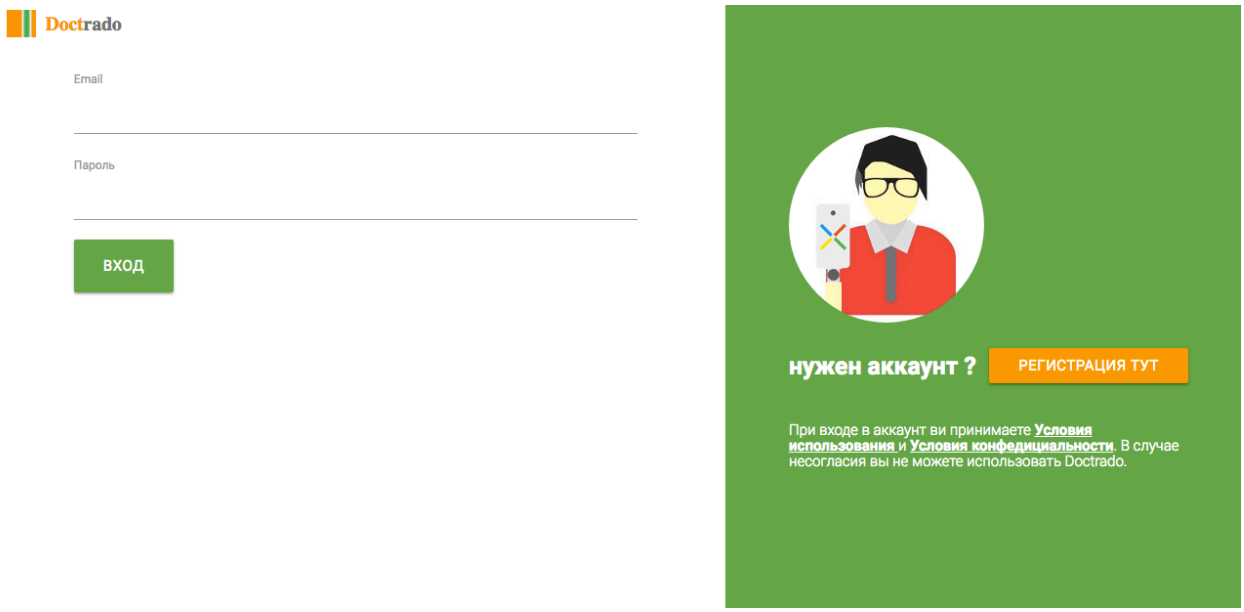

Рисунок 3.2 - sign_in.html приклад роботи

3.3 Android - кліент

Android - кліент є додатком залежним лише від серверної частини. Він уявляє собою одну Activity та Fragment вкладками.Activity - це один екран додатку. Для зберігання ресурсів та збільшення продуктивності було вирішено зробити лише одну активніть та для відображення зміни стану додатку використовувати Fragment сутності.

Перелік фрагментів:

- HomeFragment;
- CourseFragment;
- CourseFullFragment;
- LoginFragment;
- RegFragment;
- CertFragment;
- AboutFragment;
- MenuFragment;
- TestFragment;
- StatisticFragment;
- NotificationFragment;

Для управління фрагментів було розроблено FragmentManager.

HomeFragment - як можно побачити із назви це “домашня сторінка” яка уявляє собою особистий кабінет з якого є зручний доступ до усіх функцій додатку.

CourseFragment - це фрагмент який відображає доступні курси. З аналогією до WEB-кліента це фрагмент відображення зменшених представлень курсу та інтелліктуального пошуку до курсу. Після обирання курсу відбувається перехід до CourseFullFragment.

CourseFullFragment - це фрагмент який відображає один курс з подтемами та вбудованою системою тестування та отримання сертифікату.

LoginFragment - фрагмент авторизації, має можливість переходу до RegFragment. У випадку вдалої авторизації відбувається перехід до HomeFragment.

CertFragment - фрагмент центру сертифікації. Має можливіть просмотру сертифікатів поточного користувача та перевірки валідності сертифіката в базі данних.

AboutFragment - фрагмент який відображає інформацію про розробни - ка, поточну версію, назву додатку та іншу інформацію проекта.

MenuFragment - це фрагмент який реальзує концепцію NavigationDrawer. Ця концепція уявляє собою меню відкривається жестом “свайп” з зліва направо виконує функцію навігації додатку, також відображає поточного користувача у верхній частині.

TestFragment - система створення та проходження тестів у наявних курсів в залежності від ролі та авторських прав користувача.

StatisticFragment - фрагмент статистики, який відображає статистику поточного користувача.

NotificationFragment - фрагмент сповіщень користувача о змінах в статистиці клієнта.

Також в системі використовуються допоміжні утіліти такі як HTTPClient, DBWrapper, GSON.

Проект був виконаний на мові програмування JAVA з допомогою Android API.

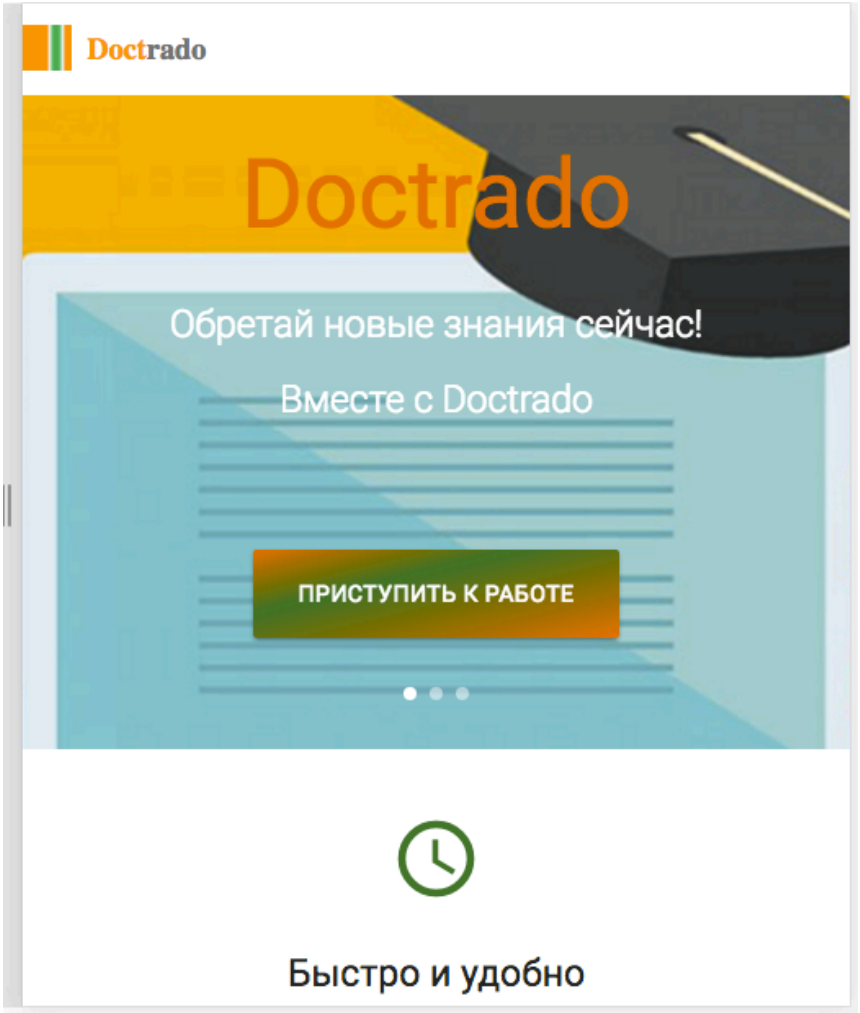

Рисунок 3.3 - приклад роботи HomeFragment

3.4 iOS - кліент

iOS - кліент являє собою повний аналог Android кліента за функціоналом, але на за реалізацією. Проект являє собою структурований набір ViewController’s. Із-за специфіки розробки проектів на мові програмування Objective - C буде використоватися префікс NSD (скорочено від NullStackDev - мого псевдоніму).

Перелік ViewController’s:

- NSDHomeViewController;
- NSDCourseViewController;
- NSDCourseFullViewController;
- NSDLoginViewController;
- NSDRegViewController;
- NSDCertViewController;
- NSDAboutViewController;
- NSDTestViewController;
- NSDStatisticViewController;
- NSDNotificationViewController.

Для управління процесами навігації було розроблено NSDNavigation-Controller, який на відміну від Android клієнту імплементує SlideMenu концепцію (у Android це NavigatonDrawer) за домогою MenuView та UIGestureRecognizer.

NSDHomeViewController - як можно побачити із назви це “домашня сторінка” яка уявляє собою особистий кабінет з якого є зручний доступ до усіх функцій додатку.

NSDCourseViewController - це фрагмент який відображає доступні курси. З аналогією до WEB-кліента це фрагмент відображення зменшених представлень курсу та інтелліктуального пошуку до курсу. Після обирання курсу відбувається перехід до NSDCourseFullViewController.

NSDCourseFullViewController - це фрагмент який відображає один курс з подтемами та вбудованою системою тестування та отримання сертифікату.

NSDLoginViewController - фрагмент авторизації, має можливість переходу до RegFragment. У випадку вдалої авторизації відбувається перехід до HomeFragment.

NSDCertViewController - фрагмент центру сертифікації. Має можливіть просмотру сертифікатів поточного користувача та перевірки валідності

NSDAboutViewController - фрагмент який відображає інформацію про розробни -ка, поточну версію, назву додатку та іншу інформацію проекта.

NSDTestViewController - система створення та проходження тестів у наявних курсів в залежності від ролі та авторських прав користувача.

NSDStatisticViewController - фрагмент статистики, який відображає статистику поточного користувача.

NSDNotificationViewController - фрагмент сповіщень користувача о змінах в статистиці клієнта.

Також в системі використовуються допоміжні утіліти такі як HTTPDefaultClient, DBCoreDataWrapper.

Проект був виконаний на мові програмування Objective - C з допомогою CocoaTouch.

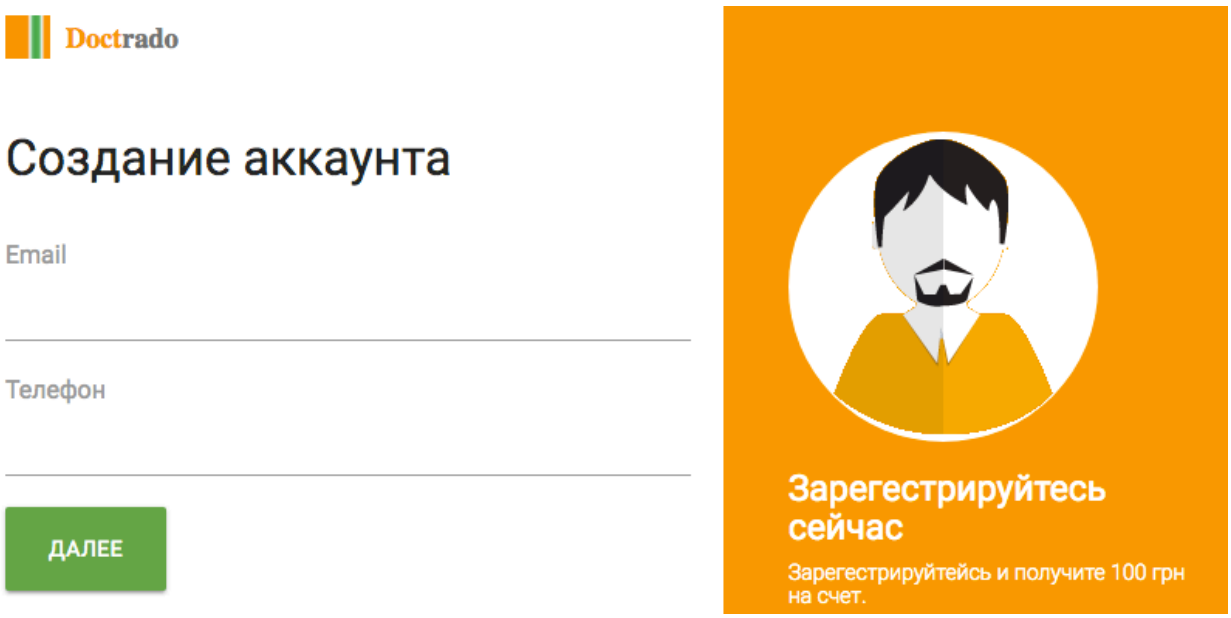

Рисунок 3.4 - NSDRegViewController приклад роботи (iPad)

4 ОХОРОНА ПРАЦІ

4.1 Безпеки при роботі на комп'ютері

Під час роботи з комп'ютером найбільшому ризику піддаються зорова, опорно-рухова, нервово-психічна системи і репродуктивна функція у жінок.

Дисплей - головне джерело небезпеки. Він випускає випромінювання декількох видів: рентгенівське, ультрафіолетове, інфрачервоне, електромагнітне. Для кожного з цих випромінювань розроблені гранично допустимі норми, проте вони досить умовні й різняться у кожній країні. Норми передбачають, що опромінюється весь організм людини, тоді як на ділі впливу піддається лише верхня частина тулуба. Згадані норми встановлені з розрахунку на кожен вид опромінення в окремо, хоча реально всі поля діють одночасно, а їх комплексний вплив досі не досліджено.

Крім того, відеодисплейний термінал порушує рівновагу між позитивно і негативно зарядженими іонами в повітрі. Електростатичне поле дисплея притягає негативні іони, порушуючи тим самим загальний баланс атмосфери. Це також шкодить здоров'ю. Вже через годину роботи біля монітора спостерігається майже повне зникнення негативних іонів. Ось чому необхідно, щоб до робочого місця за комп'ютером проникав свіже повітря. У зв'язку з усіма цими небезпеками досить чітко регламентовані розміри столу і стільця для роботи з комп'ютером.Адже "закам'яніли" постава шкідливо впливає на скелетно-м'язову систему. Стіл повинен бути просторим, із спеціальною підставкою для ніг, а робочий стілець - мати відрегульовану висоту, певний кут нахилу сидіння і спинки.

4.2 Випромінювання комп'ютера

Спочатку краще визначити, що таке «випромінювання комп'ютера».

Джерел випромінювання два. Системний блок і монітор.

- Системний блок створює тільки електромагнітне поле (випромінювання). Правда є ще й шум від вентиляторів, але ця тема всім зрозуміла і не вимагає знань електроніки. Шкода від електромагнітного поля однозначно є при високому рівні поля. Однак поле комп'ютер створює набагато менше, ніж мобільний телефон. Тобто йому далеко до мікрохвильової печі по потужності;

- Монітор має два основних шкідливих фактора. Бета-випромінювання

(а простіше, потік електронів), яке власне кажучи і створює картинку на екрані, і висока напруга (як і в будь-якому телевізорі, воно досягає 16-20 кіловольт), що викликає іонізацію повітря.

Бета-випромінювання поширюється монітором у двох напрямках - вперед і назад. У старих телевізорах і моніторах випромінювання досягало одного або двох метрів від екрану (всі пам'ятають рекомендації не сидіти ближче двох, а то й трьох метрів від телевізора?). Тобтовиходив такий собі потужний прожектор, що стріляє в нас шквалом електронів. По дорозі вибиваючи електрони з молекул повітря, перетворюючи їх у позитивні іони, так шкідливі для людини. На даний момент монітори мають дуже низький рівень бета-випромінювання, тобтоелектрони вилітають за межі екрану на пару сантиметрів. Основне випромінювання монітора направлено тому. Там «зона поразки» поширюється на метр-півтора. От її і слід уникати. Висока напруга примудряється відхоплювати у молекул повітря електрони, також перетворюючи молекули на шкідливі позитивні іони. До виробників моніторів і телевізорів пред'являються все більш жорсткі вимоги щодо використання високих напруг, і це не може не радувати.

4.3 Радіаційний опромінення

Сучасний комп'ютер - незамінна річ. І можливості його необмежені. Але є небезпека радіаційного опромінення (нехай і в дозах, які вважаються допустимими) при тривалій роботі людей біля екранів комп'ютерів.

Сучасні екрани комп'ютерів надійно захищені. Невелике (дуже маленьке) опромінення, звичайно йде. Але все це в межах санітарно допустимих норм. Нічого страшного не треба панікувати.

Тому слід вживати дієвих заходів з охорони особистої безпеки.

4.4 Захворювання опорно-рухової системи людини

Будь-яка поза при тривалій фіксації шкідлива для опорно-рухового апарату, веде до застою крові в органах. Це особливо проявляється при фізіологічному положенні різних частин тіла і тривало повторюваних одноманітних рухах. Небезпека для здоров'я представляє не тільки втома тих

груп м'язів, які ці рухи виконують, але і психологічна фіксація на них (утворення стійких вогнищ збудження ЦНС з компенсаторним гальмуванням інших її ділянок). Хоча найбільш шкідливі саме повторювані одноманітні навантаження. Під час роботи за комп'ютером людина сидить кілька годин поспіль в незручному становищі. Це не тільки загрожує втомою і загальним втомою, а й може призвести до розвитку остеохондрозу різних ділянок хребта - шийного, грудного, попереково-крижового.

У зв'язку з цим лікарі надають великого значення підтримці правильної пози під час роботи за комп'ютером. Дотримання цього правила - важливий елемент профілактики захворювань. Щоб робота за комп'ютером не шкодила здоров'ю, необхідно постійно стежити за своєю поставою. Правильна постава максимально розвантажує м'язи і дозволяє працювати довше, менше втомлюючись.

Вважається, що при правильній поставі вуха повинні розташовуватися точно в площині плечей, а плечі - точно над стегнами. Голову слід тримати рівно по відношенню до обох плечах. Коли ви дивитеся вниз, голова не повинна нахилятися вперед.

Якщо в процесі роботи ви постійно горбиться, навантаження на хребет збільшується, що приводить до надмірного розтягування м'язів. Згорблене положення може стати причиною синдрому зап'ястного каналу, грижі міжхребцевих дисків поперекового і шийного відділів.

Багато хто, дивлячись на екран монітора, витягають шию вперед. Часто це пов'язано з тим, що монітор відсунутий занадто далеко. У результаті навантаження на м'язи підстави голови і шиї зростає приблизно в три рази,
судини шиї здавлюються, погіршуючи кровопостачання голови.Крім того, людині, що сидить у такій позі, доводиться щоразу відкидати голову назад, щоб розгледіти, наприклад, лежить прямо перед ним паперовий документ. Це посилює прогин шийного відділу хребта. Згодом це може привести до головних болів і болей у руках, оскільки нерви, що відходять від спинного мозку в області шиї, простягаються до кінчиків пальців.

Сутулість викликає надмірне навантаження на плечові сухожилля і м'язи
плеча. Тривала робота в такій позі може призвести до розвитку синдрому зап'ястного каналу і защемлення плеча. Хочеться ще раз нагадати: під час роботи не горбиться, не сутультеся, не витягуйте шию.Можливо, що, почавши сидіти з правильною поставою, ви раптом відчуєте біль у м'язах. Не турбуйтеся: окремим м'язам потрібен якийсь час, щоб пристосуватися до нових навантажень.Однак після того як м'язи звикнуть до нового положення.

Тепер детально зупинимося на особливостях підйому з стільця або крісла, на яких багатьом з нас доводиться проводити значну частину часу на роботі і вдома. Хребет, суглоби разом з навколишніми м'язами і зв'язками \ "звикає" до даної позі, тому перехід у вертикальне положення вимагає плавності і точності рухів, щоб "застояні" структури опорно-рухового апарату встигли включитися в новий режим роботи. Спочатку треба пересунутися у кріслі, сівши на передню його частину. Стійко поставивши ступні на ширину плечей, нахиліть тулуб вперед приблизно до кута 70 градусів по відношенню до підлоги, намагаючись не згинати поперек. У досягнутій позиції можна вважати, що колінні суглоби зігнуті під кутом 90 градусів, а тазостегнові знаходяться в оптимальному стані.

Далі не важко відірвати від сидіння сідниці і плавно встати, стежачи за синхронністю руху між колінних і тазостегнових суглобах і випрямленням тулуба. При дотриманні цієї умови ви можете зупинитися в будь-якій точці даної траєкторії, відчуваючи себе відносно комфортно і стійко, без надмірного напруження в ділянці нирок. Ви забезпечите свій хребет від ушкоджень, якщо навчитеся сідати з вертикального положення, дотримуючись зворотну послідовність дій, рекомендованих для підйому з положення сидячи.

Пропонований спосіб підйому з крісла є навчальним і вимагає наполегливості в оволодінні цим навиком. У період виражених болів в спині він може бути складним. Найбільш поширеною помилкою є випереджаюче випрямлення колінних суглобів з подальшим випрямленням тулуба, що загрожує пошкодженням поперекових сегментів хребта. Якщо крісло з підлокітниками, можна допомогти собі встати, впираючись у них долонями.

При відсутності підлокітників можливий підйом, при якому треба зробити вихідний упор долонями у власні коліна і по черзі перемістити кисті рук вгору по стегнах, щоб допомогти випрямленню тулуба. Цей спосіб допомагає в період гострих болів у спині, але навчитися його виконання можна і в спокійний період, щоб чітко розміряти випрямлення тулуба з розгинанням в колінних і тазостегнових суглобах.

Для сором'язливих людей, які не бажають звертати на себе увагу оточуючих своїми зосередженими діями при підйомі з стільця, запропонуємо ще один спосіб. На відміну від попередніх методів він характеризується витонченістю і стрімкістю при збереженні безпеки.При цьому не треба широко розставляти ступні - досить одну ногу поставити на носок на 10 см назад під стілець, використовуючи її при підйомі зі стільця в якості стартової опори. Цей спосіб дозволяє при мінімальних зусиллях швидко опинитися.

4.5 Вимоги, дотримувані при роботі з комп'ютером

При роботі з комп'ютером шкідливими і небезпечними чинниками є:

- електростатичні поля;
- електромагнітне випромінювання;
- наявність потужних іонізуючих випромінювань;
- локальне стомлення, загальна втома;
- стомлюваність очей;
- небезпека ураження електричним струмом;
- пожежонебезпека.

Режими праці та відпочинку при роботі з комп'ютером повинні організовуватися в залежності від виду та категорії трудової діяльності.

Види трудової діяльності поділяються на 3 групи:

- група А - робота з зчитування інформації з екрана комп'ютера з попереднім запитом;
- група Б - робота з введення інформації;
- група В - творча робота в режимі діалогу.

За основну роботу з комп'ютером слід приймати таку, яка займає не менше 50% часу протягом часу роботи компьтера.

Для видів трудової діяльності встановлюється 3 категорії тяжкості і напруженості роботи з комп'ютером, які визначаються:

- для групи А - по сумарному числу прочитуються знаків за час роботи з комп'ютером, але не більше 60 000 знаків;
- для групи Б - по сумарному числу зчитуються або вводяться знаків за час роботи з комп'ютером, але не більше 40 000 знаків;
- для групи В - по сумарному часу безпосередньої роботи з комп'ютером, але не більше 6 годин за час роботи з комп'ютером.

Для забезпечення оптимальної працездатності і збереження здоров'я протягом часу роботи з комп'ютером повинні встановлюватися регламентовані перерви.

Перед початком роботи необхідно переконатися, що монітори комп'ютера мають антіблоковое покриття (окрім групи А) з коефіцієнтом відбиття не більше 0,5.

Покриття повинне також забезпечувати зняття електростатичного заряду з поверхні екрана, іскріння і накопичення пилу.

Корпус монітора повинен забезпечувати захист від іонізуючих та неіонізуючих випромінювань.

Необхідно перевірити робоче положення комп'ютера відстань між

стіною з віконними прорізами і столом має бути не менше 0,8 м. При невеликій кількості робочих місць бажано розташовувати столи біля протилежної стіни щодо віконних прорізів.

Відстань між робочими столами повинно бути не менше 1,2 м. Не допускається знаходження другого робочого місця з боку задньої стінки комп'ютера.

Екран відеомонітора повинен знаходитися від очей на оптимальній відстані 600-700мм, але не ближче 500мм.

Висота робочої поверхні столу повинна регулюватися а межах 680-800 мм, при відсутності такої можливості висота робочої поверхні столу повинна бути 725 мм.

Клавіатуру слід розташовувати на поверхні столу на відстані 100-300 мм від краю, зверненого до користувача.

Оптимальними параметрами температури в кабінеті є 19-21, допустимими 18-22, відносна вологість повітря 62-55% і соотв. 39-31%.

У кабінеті слід здійснювати наскрізне провітрювання для поліпшення якісного складу повітря, щодня проводити вологе прибирання.

Для підвищення вологості повітря слід використовувати зволожувачі. У кабінеті повинно бути штучне і природне освітлення. Основний потік природного світла повинне бути ліворуч, не допускається праворуч, ззаду і спереду працює на комп'ютері. на вікнах повинні бути завіси в два рази більше ширини вікна. Забороняється застосовувати для вікон чорні завіси.

Звернути увагу на заземлення, тому що в комп'ютері використовуються мікросхеми, чутливі до статичної електрики. Звернути особливу увагу на цілісність ізоляції всіх кабелів та роз'ємів, щоб не виявитися несподівано під напругою щодо землі. Забороняється самостійно відкрити комп'ютера, з-за високої напруги всередині. Виключається робота з комп'ютером і його периферійними пристроями з відкритим корпусом, самостійно перемикати силові та інтерфейсні кабелі, проливати рідини і т.д. Робоче місце працює на комп'ютері передбачено обладнати спеціальними меблями: обертовим стільцем із змінною висотою сидіння і кута нахилу спинки.

При роботі на комп'ютері працюючий повинен бути уважним, не відволікатися на побудову справи.

Під час роботи комп'ютера забороняється:

- залишати комп'ютер без нагляду;
- проводити ремонт;
- знімати корпус з комп'ютера.

Тривалість безперервної роботи з комп'ютером без регламентованого перерви не повинна перевищувати 2 годин.

Під час регламентованого перерви з метою зниження нервово-емоційного напруження, стомлення зорового аналізатора, усунення впливу гіподинамії та гіпокінезії, запобігання розвитку познотопіческого втоми доцільно виконувати комплекси вправ. Рівень шуму в приміщенні під час роботи комп'ютерів не повинен перевищувати 50 дБА.

Конструкція відеомонітора повинна передбачати заходи, що забезпечують хорошу розбірливість зображення, незалежну від зовнішнього освітлення.

У залежності від призначення і області застосування відеотермінали можуть бути розділені на наступні групи:

- група А - кольорові монітори тільки для демонстраційних цілей;
- група Б - кольорові монітори для персональної роботи;
- група В - монохромні монітори.

Категорично забороняється використання на робочому місці електронагрівальних приладів з відкритим елементом, відкритим вогнем.

Користування електронагрівальними приладами з закритими нагрівальними елементами дозволяється тільки у спеціально відведених для цього місцях.

Недотримання вимог до мікроклімату приміщення може не тільки різко знижувати продуктивність праці, викликати втрати робочого часу через збільшеного числа помилок у роботі, але і приводити до функціональних розладів або хронічних захворювань органів дихання, нервової системи, імунної системи.

4.6 Дії в аварійних ситуаціях, що виникають при роботі на комп'ютері

У аварійних ситуаціях комп'ютер повинен негайно відключений від мережі: при відключенні електричної енергії; при пожежі; при появі запаху диму.

Людини потрапив під напругу, негайно звільнити від дії струму, відключивши комп'ютер або відкинувши електропроводу. Якщо це неможливо зробити швидко, постраждалого відтягнути від струмоведучих частин, діючи однією рукою, ізольованою гумовою рукавичкою / сухим одягом / торкаючись лише одягу потерпілого.

Потерпілому надати першу допомогу. У перші хвилини з моменту ураження необхідно почати штучне дихання, закритий масаж серця. Під час пожежі приступити до гасіння пожежі вуглекислотним вогнегасником і викликати пожежну команду за тел. 01.

Після закінчення роботи відключити комп'ютер від мережі.

4.7 Технічні методи збільшення безпеки роботи за комп'ютером

Щоб робота була комфортним та безпечним необхідно подбати про апаратне устаткування комп'ютера. Як правило, набольший шкоду здоров'ю користувача комп'ютера наносять пристрої введення-виведення: монітор, клавіатура, мишка.

У наш час, коли проблеми безпеки роботи за комп'ютером стоять як не можна гостро, з'являється безліч різних стандартів на екологічну безпеку обладнання персонального комп'ютера. Сучасний монітор повинен відповідати принаймні трьом загальноприйнятим стандартам безпеки і ергономіки:

FCC Class B - цей стандарт розроблений канадської федеральною комісією з комунікацій для забезпечення прийнятного захисту навколишнього середовища від впливу радіозавад у замкнутому просторі. Устаткування, що відповідає вимогам FCC Class B, не повинно заважати роботі теле і радіо апаратури.

MPR-II - цей стандарт був випущений Шведським національним департаментом. MPR-II накладає обмеження на випромінювання від комп'ютерних моніторів і промислової техніки, яка в офісі.

TCO'95 (а також сучасний TCO'99) - рекомендація, розроблена Шведської конференцією профспілок та Національною радою індустріального та технічного розвитку Швеції (NUTEK), регламентує взаємодію з
навколишнім середовищем. Вона вимагає зменшення електричного і магнітного полів до технічно можливого рівня з метою захисту користувача. Для того, щоб отримати сертифікат TCO'95 (TCO'99), монітор повинен відповідати стандартам низького випромінювання (Low Radiation), тобто мати низький рівень електромагнітного поля, забезпечувати автоматичне забезпечувати автоматичне зниження енергоспоживання при довгому не використанні, відповідати європейським стандартам пожежної та електричної безпеки

EPA Energy Star VESA DPMS - відповідно до цього стандарту монітор повинен підтримувати три енергозберігаючих режиму - очікування (stand-by), припинення (suspend) і "сон" (off). Такий монітор при довгому просте комп'ютера перекладається у відповідний режим, з низьким енергоспоживанням.

Необхідно також щоб монітор мав можливість регулювання параметрів зображення (яскравість, контраст і т.д.). Рекомендується, щоб при роботі з комп'ютером частота вертикальної розгортки монітора була не нижче 75Гц (при цьому користувач перестає помічати мерехтіння зображення, яке веде до швидкого уставанію очей).

В даний час багато фірм виробники моніторів почали масовий випуск так званих плоскопанельних моніторів (LCD), які позбавлені багатьох екологічних недоліків, властивих моніторів з електронно-променевою трубкою, як то: електромагнітне випромінювання, магнітне поле, мерехтіння і т.д.

Пристрої введення інформації. На відміну від моніторів для комп'ютерних пристроїв введення (клавіатура і миша) в даний час не є загальноприйнятих і широко поширених стандартів. У той же час багато виробників даного обладнання рекламуючи свою продукцію, описують різні конструктивні рішення, які підвищують ергономічність її використання: клавіатура з можливістю регулювання розташування клавіш, миша з формою, що зменшує втому кисті при тривалій роботі. Хоча деякі з них варто розглядати тільки як помітну рекламу, багато моделей дійсно є своєрідним технологічним стрибком вперед з точки зору безпеки роботи за комп'ютером.

Ергономічна організація робочого місця - навіть сааме ергономічне устаткування в світі не допоможе вам уникнути захворювань, якщо використовувати його неправильно. Слідуючи простим радам по ергономічної організації робочого місця, можна запобігти подальшому розвитку захворювань.

Робочий простір - наукова організація робочого простору базується на даних про середній зоні охоплення рук людини - 35-40 см. Ближньої зоні відповідає область, охоплювана рукою з притиснутим до тулуба ліктем, далекій зоні - область витягнутої руки.

Робота з клавіатурою - неправильне положення рук при друку на клавіатурі призводить до хронічних растяженим кисті. Важливо не стільки відсунути клавіатуру від краю столу і обперти кисті про спеціальний майданчик, скільки тримати лікті паралельно поверхні стола і під прямим кутом до плеча. Тому клавіатура повинна розташовуватися в 10-15 см

(залежно від довжини ліктя) від краю столу. У цьому випадку навантаження доводиться не на кисть, в якій вени і сухожилля знаходяться близько до поверхні шкіри, а на більш "м'ясисту" частина ліктя. Сучасні, ергономічні моделі мають оптимальну площу для клавіатури за рахунок розташування монітора в найширшій частині столу. Глибина столу повинна дозволяє повністю покласти лікті на стіл, відсунувши клавіатуру до монітора.

Розташування монітора як правило, розташовується надмірно близько. Існує кілька наукових теорій, по різному визначають значущі фактори та оптимальні відстані від ока до монітора. Наприклад, рекомендується тримати монітор на відстані витягнутої руки Але при цьому що людина повинна мати можливість сам вирішувати, наскільки далеко буде стояти монітор.

Саме тому конструкція сучасних столів дозволяє змінювати глибину положення монітора в широкому діапазоні. Верхня межа на рівні очей або не нижче 15 см нижче рівня очей.

Внутрішній об'єм є значимим фактором є під простір стільницею. Висота наших столів відповідає загальноприйнятим стандартам, і становить 74 см. Також необхідно врахувати, що простору під кріслом і столом повинно бути досить, щоб було зручно згинати і розгинати коліна.

Крісло здавалося б, вимоги до нього сформулювати гранично просто, - воно має бути зручним. Але це ще не все. Крісло повинно забезпечувати фізіологічно раціональну робочу позу, при якій не порушується циркуляція крові і не відбувається інших шкідливих впливів. Крісло обов'язково має бути з підлокітниками і мати можливість повороту, зміни висоти і кута нахилу сидіння і спинки. Бажано мати можливість регулювання висоти і відстані між підлокітниками, відстані від спинки до переднього краю сидіння. Важливо, щоб всі регулювання були незалежними, легко здійсненними і мали надійну фіксацію. Крісло повинне бути регульованим, з можливість обертання, щоб дотягнутися до далеко розташованих предметів.

Регульоване обладнання повинно бути таким, щоб можна було прийняти таке положення:

- Поставте ступні пласко на підлогу або на підніжку;
- Поперек злегка вигнута, спирається на спинку крісла;
- Руки повинні зручно розташовуватися по сторонах;
- Лінія плечей повинна розташовуватися прямо над лінією стегон;
- Передпліччя можна покласти на м'які підлокітники на такій висоті.

4.8 Вимоги до комп'ютерної техніки

Комп'ютерна техніка розвивається сьогодні особливо стрімко, з надзвичайною швидкістю з'являються, і також швидко застарівають і відмирають різні технічні рішення і стандарти. За прогнозами різних економіко-соціологічних організацій комп'ютерна техніка та телекомунікації будуть залишатися однією з найбільш розвинутих галузей світової індустрії ще принаймні протягом 10 - 15 років. Так що зменшення числа людей, що працюють за комп'ютерами чекати не доводиться. Навпаки, повальна комп'ютеризація, вже давно охопила бізнес-сектор, сьогодні все більше захоплює масового споживача. У подібній гонці, де немає нічого постійного, складно давати рекомендації, приймати які-небудь довговічні рішення, а тим паче встановлювати стандарти. А тому, поки комп'ютерний бум не піде на спад, перед ергономікою і ергономіста будуть вставати все нові завдання, що стосуються організації безпечних і комфортних умов для людей працюючих з комп'ютерами.

Зовсім нешкідливих комп'ютерів не буває. Мова може йти тільки про більш-менш небезпечних - стверджують фахівці Держстандарту.

Головна небезпека - у випромінюваннях відеомоніторів. Їх цілий набір: випромінювання електростатичного, електричного і магнітного полів, а також рентгенівське. І хоча ці електричні і магнітні поля фізики називають слабкими, вплив на організм вони надають саме сильне. При цьому особливо шкідливі для вагітних жінок і дітей.

Крім невидимого впливу на внутрішні процеси організму, усі без винятку монітори, навіть самі ідеальні, дуже шкідливі для зору. За статистикою, щоденна робота за комп'ютером погіршує зір в середньому на одну діоптрію на рік.

Найбільш жорсткі вимоги в світі до комп'ютерної техніки пред'являють шведи, які провели якнайповніші дослідження впливу всіх видів випромінювань на здоров'я людини. Країни Європейського союзу при створенні єдиного стандарту ЄС орієнтувалися саме на шведські норми ТСО 92. Російські вимоги на випромінювання від відеомоніторів поки що нижче, ніж у ТСО 92, однак у найближчих планах Держстандарту - підняти планку безпеки до рівня шведської.

А поки при покупці комп'ютерної техніки спробуйте самі визначити рівень її безпеки. На звороті відеомонітора на табличці з технічними характеристиками можна побачити кілька емблем з літерними та цифровими позначеннями. Що вони означають, пояснює заступник начальника відділу.

Якщо ви виявили напис: "MPR 1990:10", це означає, що монітор відповідає шведському стандарту по випромінюванню, а також за змінним електричному й магнітному полях (прийнятий як стандарт ЄС).

Напис: "ISO 9241-3" позначає міжнародний стандарт, який задовольняє ергономічним вимогам до дисплеїв і стоїть на сторожі вашого зору.

Написи: "TCO-1992" або "TCO-1995" розшифровуються як відповідність вимогам Шведського союзу професійних службовців (TCO) за візуальним ергономічним параметрам і змінним електричному й магнітному полях.

Якщо ви знайдете напис: "MPR II", це теж непогано, але врахуйте, що такий знак далеко не повністю відображає всі умови стандарту "MPR 1990:10".

Повторимо, знаки перерахованих міжнародних стандартів, а також російські сертифікати або знаки відповідності - це ще зовсім не свідчення абсолютної нешкідливості вашого екрану.Хоча в комп'ютерах останнього покоління використовуються досить дієві засоби захисту, тим не менш не треба обманюватися: небезпека лише зведена до можливого мінімуму, не більше того. Не випадково в багатьох країнах робота за комп'ютером включена до переліку найбільш шкідливих видів діяльності.

ВИСНОВКИ

У цьому дипломному проекті були досягнуты всі поставлені завдання, а саме: розроблено програмне забезпечення з продуманим і простим інтерфейсом, що реалізовує основні функції системи дистанційного інтерактівного навчання, дозволяє зручно користуватися сервісом, додавати курси.

Для досягнення мети були виконані наступні пункти: реалізація механізму авторизації с використанням бази даних, створення системи відображення курсу, а також розроблення максимально інформативної головної сторінки сайту, де користувач міг би максимально детально ознайомитися з найпопулярнішими курсами, за допомогою інтеллектуального пошуку орієнтуватися серед переліку всіх статей і швидко знаходити необхідну.

Також у проекті продемонстровано механізм монолітної і воднотчас максимально гнучкої декомпозованої системи.

ПЕРЕЛІК ПОСИЛАНЬ

1. *Кірілова Г.І.* Інформаційні технології та комп'ютерні засоби в освіті / Г.І. Кірілова. - 2000.

2. *Можаєва Г.В.* Проектна діяльність в системі дистанційної освіти. Теоретико - методологічні проблеми історичного пізнання. / Г.В Можаєва. - 2001. - 317 с.

3. *Демкін В.П* Технології дистанційного навчання. / В.П. Демкін, Г.В. Можаева. - 2002.

4. *Краснова Г.А.* Інформаційно - педагогічне забезпечення для дистанційного навчання. / Г.А. Краснова, М.И. Беляев. - 2001.

5. *Назаров П.О.* Роль практикуму в навчальному процесі та его реалізація в ДО. Наукове и методичне забезпечення системи дістанційної освіти. / П.О. Назаров, Т.В. Руденко. - 2000. - 112 с.

6. *Казанська О.В* Тестуючі програми для використання в мережі Інтернет. Відкрита і дистанційна освіта. / О.В. Казанська, А.С. Русанов, Л.Г. Макаревіч. - 2001.

7. Нові педагогічні та інформаційні технології в системі освіти / Е.С. Полат, М.Ю. Бухаркін, М.В. Моїсеєва, А.Є. Петров - 2001. - 272 с.

8. *Бовлінг Е.* Еволюція системи навчення Lotos : пер. з англ / Е. Бовлінг. - 2009 - 254 с.

Д О Д А Т К И

ДОДАТОК А

ПРОГРАМНИЙ КОД СЕРВЕРНОЇ ЧАСТИНИ

```
package Auth;
import DBSingletones.DBAdmin;
import DBSingletones.DBStudent;
import DBSingletones.DBTeacher;
import Model.Admin;
import Model.Student;
import Model.Teacher;
/**
 * Created by NSD on 17.05.17.
 */
public class Authorizator {
    static final String AUTH_ATTR = "auth";
    private static volatile Authorizator instance;
    public static Authorizator getInstance() {
        Authorizator localInstance = instance;
        if (localInstance == null) {
            synchronized (Authorizator.class) {
                localInstance = instance;
                if (localInstance == null) {
                    instance = localInstance = new Authorizator();
                }
            }
        }
        return localInstance;
    }
    public AuthRealm Auth(String login,String hexOfPass){
        Object retVal;
        retVal = DBStudent.getInstance().queryStudent(login);
        if(retVal==null){
            retVal = DBTeacher.getInstance().queryTeacher(login);
        }
        if(retVal==null){
            retVal = DBAdmin.getInstance().queryAdmin(login);
        }
        if(retVal!=null){
```

```
    if(retVal instanceof Student){
                Student tS = (Student) retVal;
              if(tS.getPasswordHex().equals(hexOfPass)){
                 //clear creditals
                 tS.setPasswordHex("");
                 tS.setCardCSV("");
                 tS.setCardNumber("");
                 tS.setCardExploedInfo("");
                 return new AuthRealm(tS,UserRole.STUDENT);
              }
           }
           if(retVal instanceof Teacher){
              Teacher tT = (Teacher) retVal;
              if(tT.getPasswordHex().equals(hexOfPass)){
                 //clear creditals
                 tT.setPasswordHex("");
                 tT.setCardCSV("");
                 tT.setCardNumber("");
                 tT.setCardExploedInfo("");
                 return new AuthRealm(tT,UserRole.TEACHER);
              }
           }
           if(retVal instanceof Admin){
              Admin tA = (Admin) retVal;
              if(tA.getPasswordHex().equals(hexOfPass)){
                 tA.setPasswordHex("");
                 //clear creditals
                 return new AuthRealm(tA,UserRole.ADMIN);
              }
           }
        }
        return new AuthRealm(); //ret Annonimous
     }
     public AuthRealm Reg(Object user,String login,String hexOfPass){

  if(DBAdmin.getInstance().hasAdmin(login)||DBStudent.getInstance().hasStudent(l
  ogin)|DBTeacher.getInstance().hasTeacher(login)){

           return new AuthRealm();
```

```
        }
        if(user instanceof Student){
            Student regStudent = (Student) user;

    if(regStudent.getPasswordHex()==null|regStudent.getPasswordHex().equals("")){
                regStudent.setPasswordHex(hexOfPass);
            }

            DBStudent.getInstance().addStudent(regStudent);

            regStudent.setCardNumber("");
            regStudent.setCardExploedInfo("");
            regStudent.setCardCSV("");
            regStudent.setPasswordHex("");

            return new AuthRealm(regStudent,UserRole.STUDENT);

        }

        if(user instanceof Teacher){


            Teacher regTeacher = (Teacher) user;

    if(regTeacher.getPasswordHex()==null|regTeacher.getPasswordHex().equals("")){
                regTeacher.setPasswordHex(hexOfPass);
            }

            DBTeacher.getInstance().addTeacher(regTeacher);
            regTeacher.setCardNumber("");
            regTeacher.setCardExploedInfo("");
            regTeacher.setCardCSV("");
            regTeacher.setPasswordHex("");
            return new AuthRealm(regTeacher,UserRole.TEACHER);
        }


        return new AuthRealm();
```

```
    }
}


package Auth;

import Model.Annonymous;

/**
 * Created by NSD on 17.05.17.
 */
public class AuthRealm {

    private Object user;
    private UserRole accessRole;

    public AuthRealm(Object user, UserRole accessRole) {
        this.user = user;
        this.accessRole = accessRole;
    }

    public AuthRealm(){
        this.user = new Annonymous();
        this.accessRole = UserRole.ANNONYMOUS;
    }

    public Object getUser() {
        return user;
    }

    public void setUser(Object user) {
        this.user = user;
    }

    public UserRole getAccessRole() {
        return accessRole;
    }

    public void setAccessRole(UserRole accessRole) {
        this.accessRole = accessRole;
    }
}
```

```
package Auth;

import DBSingletones.DBAdmin;
import DBSingletones.DBStudent;
import DBSingletones.DBTeacher;
import Model.*;
import NSDReqCodeUtils.ReqCode;
import com.google.gson.Gson;
import com.j256.ormlite.field.DatabaseField;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.security.acl.LastOwnerException;
import java.util.IllegalFormatCodePointException;
import java.util.Map;

/**
 * Created by NSD on 17.05.17.
 */

@WebServlet(name = "Auth",urlPatterns = "/API/Auth")
public class AuthServlet extends HttpServlet{

    private static final String U_NAME_PARAM = "login";
    private static final String U_PASS_PARAM = "pass";


    @Override // delete creditals
    protected void doDelete(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {

        boolean isSuccesfull = false;

        try {
            req.getSession().removeAttribute(Authorizator.AUTH_ATTR);
            isSuccesfull = true;

            req.changeSessionId();

        }catch (Exception e){
            e.printStackTrace();
```

```
    }

        if(isSuccesfull){
           resp.setCharacterEncoding("UTF-8");
           resp.setStatus(HttpServletResponse.SC_OK);
           PrintWriter out = resp.getWriter();
           out.print(ReqCode.getCodeWith(-1));
           out.flush();
           out.close();
        }
        else {
           resp.setCharacterEncoding("UTF-8");
           resp.setStatus(HttpServletResponse.SC_OK);
           PrintWriter out = resp.getWriter();
           out.print(ReqCode.getCodeWith(0));
           out.flush();
           out.close();
        }




    }

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        req.setCharacterEncoding("UTF-8");
        resp.setCharacterEncoding("UTF-8");
        AuthRealm authRealm = null;

        try {
           authRealm = (AuthRealm) req.getAttribute(Authorizator.AUTH_ATTR);
        }catch (Exception e){
          // authRealm = new AuthRealm();
        }

        if(authRealm == null){
           authRealm = new AuthRealm();
           authRealm.setAccessRole(UserRole.ANNONYMOUS);
        }

        Gson gson = new Gson();
```

```
    PrintWriter out = resp.getWriter();
        resp.setStatus(HttpServletResponse.SC_OK);

        switch (authRealm.getAccessRole()){
            case ADMIN:
                Admin admin = (Admin) authRealm.getUser();
                out.write(gson.toJson(new
RealmModelForJSON<Admin>(admin,"ADMIN")));
            break;
            case ANNONYMOUS:
                String respG = "ANNONYMOUS";
                out.write(respG);
            break;
            case STUDENT:
                Student student = (Student) authRealm.getUser();
                out.write(gson.toJson(new
RealmModelForJSON<Student>(student,"STUDENT")));
            break;
            case TEACHER:
                Teacher teacher = (Teacher) authRealm.getUser();
                out.write(gson.toJson(new
RealmModelForJSON<Teacher>(teacher,"TEACHER")));
            break;


        }
            out.flush();
            out.close();



    }


    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        req.setCharacterEncoding("UTF-8");
        resp.setCharacterEncoding("UTF-8");

        Map<String,String[]> parameterMap = req.getParameterMap();


        String login = "";
        String pass = "";
```

```
try {
    login = parameterMap.get(U_NAME_PARAM)[0];
    pass = parameterMap.get(U_PASS_PARAM)[0];
}catch (Exception e){
    e.printStackTrace();
}

Object result = null;
UserRole userRole = UserRole.ANNONYMOUS;

//TODO get exist user is db




AuthRealm resultRealm = Authorizator.getInstance().Auth(login,pass);

if(resultRealm.getAccessRole()==UserRole.ANNONYMOUS){
    PrintWriter out = resp.getWriter();
    resp.setStatus(HttpServletResponse.SC_OK);
    out.write("-1");
    out.flush();
    out.close();
}else{
    req.setAttribute("auth",resultRealm);
    PrintWriter out = resp.getWriter();
    resp.setStatus(HttpServletResponse.SC_OK);
    switch (resultRealm.getAccessRole()){
        case ADMIN:
            out.write("1"); break;
        case STUDENT:
            out.write("2"); break;
        case TEACHER:
            out.write("3"); break;
    }

    out.flush();
    out.close();


}
```

```
    }


    @Override // registration
    protected void doPut(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        req.setCharacterEncoding("UTF-8");
        resp.setCharacterEncoding("UTF-8");

        Map<String,String[]> parameterMap = req.getParameterMap();

        UserRole role = UserRole.ANNONYMOUS;


        try{
            String tempUserRole = parameterMap.get("role")[0];

            if(tempUserRole.equals("t")){
                role = UserRole.TEACHER;
            }
            if(tempUserRole.equals("s")){
                role = UserRole.STUDENT;
            }
        }catch (Exception e){
            e.printStackTrace();
            resp.setStatus(HttpServletResponse.SC_OK);
            PrintWriter out = resp.getWriter();
            out.write(-1);
            out.flush();
            out.close();
        }



        if(role.equals(UserRole.STUDENT)){
            String email = null;
            String passwordHex = null;
            String avatar = null;
            String name = null;
            String surname = null;
            String firstname = null;
```

```
            String cardNumber = null;
            String cardExploedInfo = null;
            String cardCSV =null ;
            String city = null ;
            String educationInfo = null ;
            String educationInfoDoc = null;
            String bornDate = null;
            String telNumber = null;
            int balance = 100;

            try {
               email = parameterMap.get("email")[0];
               passwordHex = parameterMap.get("password")[0];
               avatar = parameterMap.get("password")[0];
               name = parameterMap.get("name")[0];
               surname = parameterMap.get("surname")[0];
               firstname = parameterMap.get("firstName")[0];
               city = parameterMap.get("city")[0];
               educationInfo = parameterMap.get("eduInfo")[0];
               educationInfoDoc = parameterMap.get("eduInfoDoc")[0];
               bornDate = parameterMap.get("bornDate")[0];
               telNumber = parameterMap.get("telNumber")[0];


               if(DBAdmin.getInstance().hasAdmin(email)||
DBStudent.getInstance().hasStudent(email)||
DBTeacher.getInstance().hasTeacher(email)){
                  throw new IllegalFormatCodePointException(1);

               }else{

                  Student tempStudent = new Student();

                  tempStudent.setEmail(email);
                  tempStudent.setPasswordHex(passwordHex);
                  tempStudent.setCardNumber("");
                  tempStudent.setCardCSV("");
                  tempStudent.setCardExploedInfo("");
                  tempStudent.setName(name);
                  tempStudent.setFirstname(firstname);
                  tempStudent.setActivated(false);
                  tempStudent.setCity(city);
                  tempStudent.setBalance(100);
                  tempStudent.setAvatar(avatar);
                  tempStudent.setSurname(surname);
                  tempStudent.setBornDate(bornDate);
```

```
                tempStudent.setEducationInfo(educationInfo);
                tempStudent.setEducationInfoDoc(educationInfoDoc);
                tempStudent.setTelNumber(telNumber);
                DBStudent.getInstance().addStudent(tempStudent);
                AuthRealm resultRealm =
    Authorizator.getInstance().Auth(email,passwordHex);
                if(resultRealm.getAccessRole()==UserRole.ANNONYMOUS){
                    PrintWriter out = resp.getWriter();
                    resp.setStatus(HttpServletResponse.SC_OK);
                    out.write("-1");
                    out.flush();
                    out.close();
                }else{
                    req.setAttribute("auth",resultRealm);
                    PrintWriter out = resp.getWriter();
                    resp.setStatus(HttpServletResponse.SC_OK);
                    switch (resultRealm.getAccessRole()){
                        case ADMIN:
                            out.write("1"); break;
                        case STUDENT:
                            out.write("2"); break;
                        case TEACHER:
                            out.write("3"); break;
                    }

                    out.flush();
                    out.close();
                }
            }
        }
        catch (Exception e){

            if(e instanceof IllegalFormatCodePointException){

                PrintWriter out = resp.getWriter();
                resp.setStatus(HttpServletResponse.SC_OK);
                out.write("0");
                out.flush();
                out.close();

            }
            else {

                e.printStackTrace();
                PrintWriter out = resp.getWriter();
                resp.setStatus(HttpServletResponse.SC_OK);
```

```
                out.write("-1");
                 out.flush();
                 out.close();
              }



        }



    }

    if(role.equals(UserRole.TEACHER)){

        long id = -1;
        String email = null;
        String passwordHex = null;
        String avatar = null;
        String name = null;
        String surname = null;
        String firstname = null;
        String cardNumber = null;
        String telNumber = null;
        String cardExploedInfo = null;
        String cardCSV = null;
        String expirence = null;
        String bornDate = null;
        String educationInfo = null;
        String educationInfoDoc = null;
        int balance = 100;
        boolean isActivated = false;
        String city = null;

        if(DBAdmin.getInstance().hasAdmin(email)||
DBStudent.getInstance().hasStudent(email)||
DBTeacher.getInstance().hasTeacher(email)){
            throw new IllegalFormatCodePointException(1);

        }else{

            email = parameterMap.get("email")[0];
            passwordHex = parameterMap.get("password")[0];
```

```
            avatar = parameterMap.get("password")[0];
            name = parameterMap.get("name")[0];
            surname = parameterMap.get("surname")[0];
            firstname = parameterMap.get("firstName")[0];
            city = parameterMap.get("city")[0];
            educationInfo = parameterMap.get("eduInfo")[0];
            educationInfoDoc = parameterMap.get("eduInfoDoc")[0];
            bornDate = parameterMap.get("bornDate")[0];
            telNumber = parameterMap.get("telNumber")[0];
            expirence = parameterMap.get("expirence")[0];

            Teacher tempTeacher = new Teacher();

            tempTeacher.setEmail(email);
            tempTeacher.setPasswordHex(passwordHex);
            tempTeacher.setCardNumber("");
            tempTeacher.setCardCSV("");
            tempTeacher.setCardExploedInfo("");
            tempTeacher.setName(name);
            tempTeacher.setFirstname(firstname);
            tempTeacher.setActivated(false);
            tempTeacher.setCity(city);
            tempTeacher.setBalance(100);
            tempTeacher.setAvatar(avatar);
            tempTeacher.setSurname(surname);
            tempTeacher.setBornDate(bornDate);
            tempTeacher.setEducationInfo(educationInfo);
            tempTeacher.setEducationInfoDoc(educationInfoDoc);
            tempTeacher.setTelNumber(telNumber);
            tempTeacher.setExpirence(expirence);

            DBTeacher.getInstance().addTeacher(tempTeacher);

            AuthRealm resultRealm =
Authorizator.getInstance().Auth(email,passwordHex);
            if(resultRealm.getAccessRole()==UserRole.ANNONYMOUS){
               PrintWriter out = resp.getWriter();
               resp.setStatus(HttpServletResponse.SC_OK);
               out.write("-1");
               out.flush();
               out.close();
            }else{
               req.setAttribute("auth",resultRealm);
               PrintWriter out = resp.getWriter();
               resp.setStatus(HttpServletResponse.SC_OK);
               switch (resultRealm.getAccessRole()){
```

```
            case ADMIN:
                    out.write("1"); break;
                case STUDENT:
                    out.write("2"); break;
                case TEACHER:
                    out.write("3"); break;
                }

                out.flush();
                out.close();
            }
        }

    }


}
```