МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет “Львівська політехніка”

*Кафедра САПР*



**Курсова робота**

з дисципліни

«Технологія створення програмних продуктів»

на тему:

«Система автоматизації роботи студії візажу»

Виконали:

студенти групи КНз-31

Вонс Петро

Чалий Михайло

Чеховський Павло

Прийняв:

Матвійків О.М.

Львів 2015

ЗМІСТ

1. Mетаданої курсової роботи (проекту)
2. Індивідуальне завдання
3. Послідовність виконання та зміст проекту
4. Опис моделі команди, опис моделі процесу
5. Актуальність даної теми
6. Постановка задачі
7. Порівняльний аналіз існуючих програмних рішень, аналіз ризиків
8. Схеми варіантів використання ПП
9. Функціонально-логічна схема
10. Архітектура ПП. «Клієнт-серверна» архітектура
11. Плани, бюджет
12. Індивідуальна частина
13. Тестування
14. Висновок
15. Використана література

**Метаданої курсової роботи (проекту)**

Вивчення та практична участь в процесі розроблення програмного продукту в невеликій команді розробників.

**Індивідуальне завдання**

|  |  |
| --- | --- |
| Система автоматизації студії візажу Nadia Gordon | * реєстрація директорів по філіях і адміністраторів по містах * реєстрація викладачів по школах і їхніх курсів * наповнення клієнтської бази даних * розрахунок зарплат та премій * розрахунок витрат та доходів * формування звітів. |

**Послідовність виконання та зміст проекту:**

1. Сформувати команду (2-4 чол.) та отримати (вибрати) варіант індивідуального завдання.

2. Проаналізувати індивідуальне завдання, вибрати модель процесу розроблення ПП, визначити стадії, головні та додаткові віхи кожної стадії, розподілити ролі учасників команди та описати індивідуальні обов’язки кожного учасника згідно моделей MSF.(*Опис моделі команди, опис моделі процесу*).

3. Формулюється “Постановка задачі”, в якій описують необхідність розроблення даного програмного продукту, а також його предметна область. (*Табличка "Постановка задачі"*).

4. Виконується “Передпроектне дослідження”, в якому дається детальний огляд уже існуючих програмних продуктів і рішень в даній предметній області, аналіз необхідних технологій, а також здійснюють аналіз ризиків. (*Порівняльний аналіз існуючих програмних рішень, Аналіз ризиків*).

5. Здійснюють опис бізнес процесів, в яких буде працювати система; виявляють зовнішні і внутрішні фактори, що мають відношення до побудови програмної системи, розробляють рисують та аналізують варіанти використання ПП. (*Схеми варіантів використання ПП*).

6. Проводять архітектурне проектування - розглянути, описати і пояснити вибір архітектури ПП. (*Архітектура ПП*)

7. Функціональне проектування – розробити функціонально-логічну схемуПП з деталізацією програмних модулів до третього рівня. Виділити зовнішні та внутрішні сутності в програмній системі (напр., блок з тінню для зовнішньої сутності, а без тіні - для внутрішньої). Взаємозв’язки між блоками (або процеси в системі) повинні мати напрямки. Кожен блок повинен мати коротке позначення на схемі й розгорнутий опис в тексті. (*Схема, опис блоків в тексті записки*).

8. Інфологічне проектування – при необхідності вибрати БД, розробити структуру даних ПП, та схеми потоків даних в системі. (*Інфологічна схема, опис структури даних, опис потоків даних*).

9. Провести контроль спроектованої програмної системи (ПП) на коректність і відсутність протиріч.

10. Сформувати зведений план розроблення ПП і розрахувати бюджет в MS Project (*Плани, бюджет*):

- План розроблення програмних модулів;

- План тестування модулів програмної системи;

- План дистрибуції та впровадження ПП.

11. Програмна реалізація: (*структурна схема або схема взаємодії класів, програмний код з коментарями (≈ 1 коментар на 10 рядків коду), результати компіляції - екзешка, процедура, компонента, тощо*)

- Спроектувати інтерфейс користувача та розробити головну програму (головну сторінку сайту) - для менеджера команди;

- Здійснити повну програмну реалізацію одного вибраного модуля програмного продукта - для кожного учасника команди.

12. Зібрати і скомпілювати розроблені модулі та провести робоче тестування розробленого ПП. (*Протокол тестування*).

13. Провести верифікацію роботи системи згідно з Постановкою задачі та Функціональною специфікацією. (*Протокол верифікації*).

14. Сформувати документацію на розроблений ПП. (*Документація*).

15. Розробити інструкцію користувача ПП. (*Інструкція*, *help-система*).

16. Підготовити індивідуальний звіт (*пояснювальну записку*) про виконану роботу для захисту проекту у відповідності до держ. стандартів. Обсяг записки не менше 20 стор. Схеми оформляються у відповідності з держ. стандартами та згідно з правилами вибраної методології розроблення ПП.

17. Розробити та підготувати спільну (об’єднану) презентацію в PowerPoint процесу розроблення ПП та отриманих результатів. (*Індивідуальна презентація кожного учасника групи*).

18. Сформувати власні висновки. Для цього провести аналіз створеного ПП і зробити висновки про виконану роботу, в яких потрібно сформувати чіткі вказівки щодо доцільності застосування в даному проекті тих чи інших методів, середовищ або технологій, їх функціональної надлишковості чи недостатності й т.д., а також дати рекомендації про шляхи усунення виникаючих обмежень. У висновках необхідно відмітити актуальність/неактуальність і подальше використання реалізованого проекту. (*Висновки*).

19. Представити список використаної літератури.

**Опис моделі команди, опис моделі процесу**

У проекті розроблення ПП було вирішено використовувати водоспадну модель (рис.1)., тому що, технічне завдання залишається незмінним впродовж всього процесу розроблення ПП.

Дуже часто класичний життєвий цикл називають каскадною або водопадом моделлю, підкреслюючи, що розробка розглядається як послідовність етапів, причому перехід на наступний, ієрархічно нижній етап відбувається тільки після повного завершення робіт на поточному етапі.

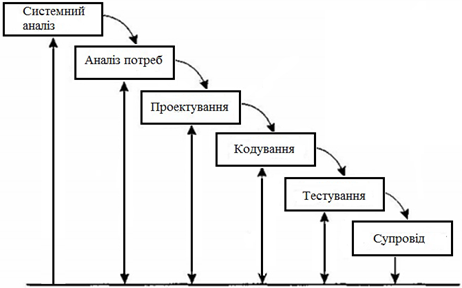


Рис.1.Схема водоспадної моделі

Розробка починається на системному рівні і проходить через аналіз, проектування, кодування, тестування і супровід. При цьому моделюються дії стандартного інженерного циклу.

Системний аналіз задає роль кожного елементу в комп'ютерній системі, взаємодія елементів один з одним.На етапі системного аналізу та аналізу вимог в нас була задіяна вся команда, кожен з членів приймав участь у обговореннях. В ході планування проекту визначили об'єм проектних робіт і їх ризик, необхідні трудовитрати, сформували робочі завдання і план-графік робіт. Уточнили і деталізували функції ПП, характеристики і інтерфейс.

Зазвичай в проектуванні виділяють два ступені: попереднє проектування і детальне проектування. Попереднє проектування формує абстракції архітектурного рівня, детальне проектування уточнює ці абстракції, додає подробиці алгоритмічного рівня. Крім того, у багатьох випадках виділяють інтерфейсне проектування, мета якого — сформувати графічний інтерфейс користувача (GUI). Для веб-сайту графічний інтерфейс є дуже важливим. Оформленням та дизайном займались всі члени групи. Програмуванням виділених функцій займалися Вонс П., Чалий М. Над тестуванням працював Чеховський П., про створення інструкції користувача (help-системи) подбали всі члени групи.

**Актуальність даної теми**

У наш час автоматизація роботи компаній – це засіб для грамотного та ефективного управління сучасним підприємством. Одним з важливих етапів є розробка веб-сайтів. Автоматизація роботи за допомогою веб-сайту чи додатку - ефективне рішення для роботи студії візажу. Як показує практика, автоматизація роботи підвищує задоволеність клієнта.

Веб-додаток - дуже зручний спосіб, лля того щоб покращити роботу студії і залишати її успішність на високому рівні. Адже розробка такого додатку дасть можливість легко здійснювати :

* реєстрацію директорів по філіях і адміністраторів по містах;
* реєстрацію викладачів по школах і їхніх курсах;
* наповнення клієнтської бази даних;
* розрахунок зарплат та премій;
* розрахунок витрат та доходів;
* формування звітів.

Всі ми стикалися із ситуацією, коли заходимо на сайт, але неможемо в ньому розібратися, тому що все заплутано. Але розробка цього веб-додатку дасть клієнтам зручність, простоту, швидкість і комфорт.

З аналізу даної тематики виходить, що дана курсова робота є актуальною на сьогоднішній день .

**ПОСТАНОВКА ЗАДАЧІ**

|  |  |
| --- | --- |
| **Елемент** | **Опис** |
| **Проблема полягає в** складності ведення роботи | Складності введення звітності роботи компанії, розрахунок доходів і розходів, при чому, коли штат збільшується, то і обсяг роботи, і ймовірність помилитись при розрахунках виростає, якщо не автоматизувати роботу. При цьому цей весь розрахунок здійснює директор компанії з великою кількістю звітних журналів і на калькуляторі, що і затягує сам процес. |
| **Проблема зачіпає** весь персонал | Директор, викладачі, адміністратори, менеджери, web-адміністратори |
| **В результаті -** низька якість і швидкість роботи. Тому нами було вирішено створити ПП веб-додатку | Затягування виплати зарплат та звітування про кількість відроблених годин і які послуги були наданні учасником. Учасники введуть звітність у різних журналах, що додає об’ємну роботу. Велика кількість розхідних матеріалів для ведення цієї звітності. Все це впливає на якість роботи і швидкість її виконання. Замість того, щоб провести додатковий курс, чи витратити цей час на особисте життя, працівник повинен заповнити кожну документацію і здати вкінці місяця. При цьому цей час також оплачується і закуплений канцелярський набір також приносить збиток компанії.. |
| **Перевагами нового рішення є** можливість оперативної зміни інформації, великий об’єм інформації, спрощення роботи, розвинута система пошуку, зрозумілий інтерфейс. | Спрощення введеня звітності доходів/розходів і інформації про працівників. Зберігання всіх даних в одній базі і швидкому доступі та пошуку по них. Автоматичний кінцевий підрахунок зарплат, доходів і витрат. Виведення статистики по заданих критеріях за певний період. Зручний і спрощений інтерфейс, що не буде потребувати спеціального навчання працівники для його освоєння. Рішення буде реалізовано як веб-додаток, що дозволяє доступитись в будь який час через інтернет і з будь якого пристрою. |

## 

## Передпроектне дослідження

### Порівняльний аналіз існуючих програмних рішен

В даний час на ринку представлені такі програми, які можуть стати рішенням даної проблеми: 1С Бухгалтерія, Соната, Арт-Звіт Плюс, M.E.Doc IS, АРМ Звіт.

Всі ці програми вже здобули частку ринку і популярність серед компаній. Але всі вони так чи інакше володіють недоліками, і незважаючи на оновлення працюють за старим принципом і із застарілими технологіями.

Насамперед, перед роботою всі вище перелічені програми повинні бути встановленні на комп’ютер і постійно обновлятись до потрібних версій. На деяких програмах взагалі нема можливості автоматичного оновлення і це потрібно робити за принципом: знайти оновлення – завантажити – здійснити оновлення, іноді вимагається перед оновленням видалити попередню версію. Це завдає певного клопоту.

Далі кожен користувач, що повинен мати доступ до даних та можливість їх редагувати, повинен здійснити встановлення даної програми собі на комп’ютер, який повинен відповідати програмним та апаратним вимогам: апаратні характеристики, операційна системі, додаткові бібліотеки, програми та фреймворки. Це ускладнює доступ потребуючих сторін до даних, а також завжди вчасно отримати останні оновленні дані. Також виникає проблема синхронізації, коли дві сторони одночасно оновлюють дані з різних комп’ютерів, при цьому, якщо це сталось, то постає проблема як об’єднати різні оновлення.

Якщо робота зі звітністю здійснюється на кількох комп’ютерах, то її потрібно засинхронізувати, або в гіршому випадку необхідно скористатись спеціальними функціями імпорту/експорту, які зберігають все у відповідний формат і переносяться за допомогою носіїв на інші комп’ютери, де вподальшому і обробляється.

Наше рішення з базою даних розміщується в єдине місці– сервер. Доступ і управління здійснюється через мережу Інтернет за допомогою будь-якого пристрою, що має браузер і доступ до Інтернету. Оновлення даних зберігається безпосередньо в базі даних і їх тут же можна переглянути в будь-який момент часу.

Програми також потребують деякої підготовки людей і знань, які збираються працювати на них. Наше рішення буде простим і інтуїтивним, де не потрібно великих зусиль, щоб осилити роботу.

Отже підведемо підсумок по кожній з цих програм, і наведемо їх опис.

### Порівняльний аналіз альтернативних програм

* 1С Бухгалтерія
  + Недоліки:
    - Основні застосування 1С: рішення задач автоматизації податкового та бухгалтерського обліку.
    - Налаштування, впровадження і запуск повинні здійснюватись кваліфікованим 1С-програмістом.
    - Необхідність замовляти послуги підтримки 1С (виправлення помилок в конфігураци, консультації персоналу відносно роботи з програмою, установка оновлень програми).
    - Платні оновлення продуктів.
    - Низька безпека і захищеність інформації, використовуваної 1С.
  + Переваги:
    - Наявність великої кількості потенційного персоналу для роботи в компанії, де впроваджені рішення на базі 1С.
    - Максимально швидка і якісна підтримка бухгалтерських рішень з боку Компанії 1С.
    - Можливість створювати або допрацьовувати індивідуальні проекти , які враховують бізнес-процеси кожній організації.
    - Вбудована об'єктно-орієнтована мова, спеціально розроблена компанією 1С.
    - Для зручності роботи програміста в оболонці "1С: Конфігуратор" розроблені різні допоміжні інструменти (Синтакс-help, колірне підсвічування синтаксису, вбудований дебагер і т.п.).
    - Повна відкритість програмних продуктів 1С
    - Єдина технологічна платформа
* Соната
  + Недоліки:
    - Вартість
    - Незручний і заплутаний інтерфейс (<http://sonata.biz.ua/forum/topic/763?page=1>).
    - Відсутність планового оновлення, або хоча б повідомлення про необхідність його.
  + Переваги:
    - Робота з багатьма підприємствами
    - Імпорт профілів та документів з інших програм електронної звітності (ІВК Звітність, ОПЗ, M.E.Doc, Арт-Звіт)
    - Підтримка електронних ключів різних центрів сертифікації
    - Актуальні довідники даних, що використовуються в звітності
* Арт-Звіт
  + Переваги:
    - Зручний в користуванні
    - Простий інтерфейс
    - Не вимагає значних ресурсів для роботи
  + Недоліки:
    - Оновлення встановлюються вручну.
    - Оновлення є послідовними. Тобто, якщо версія 10 і потрібно оновитись до 20, то перед цим треба встановити оновлення версій з 11 до 19.
    - Цей проект більше не підтримується.
    - Відсутність в подальшому нових оновлень.
* M.E.Doc IS
  + Недоліки:
    - Конфліктує з антивірусами і фаєрволами, тому їх потрібно відключати під час роботи.
    - Складний в налаштуваннях.
    - Щоб освоїти роботи необхідно отримати спеціальні навички і пройти курс навчання
  + Переваги:
    - Надійний засіб накладання ЕЦП, що дозволяє конфіденційно обмінюватися інформацією з контрагентами.
    - Функція «Мобільний підпис» дає можливість переглянути і підписати будь-який документ в програмі з будь-якої точки світу за допомогою мобільного телефону, що спрощує роботу керівника, який відсутній на робочому місці в момент відправки створення і відправки документа (звіту, договору, податкової накладної, актами) .
    - Реалізована можливість прийняття повідомлень на мобільний телефон про прийом звіту контролюючим органом.
    - Для створення договорів використовуються шаблони, які можна редагувати, а також імпортувати з Microsoft Office, що заощадить Ваш час при створенні нового договору.
    - Вбудований модуль «Журнал подій» призначений для збереження історії подій в системі.
    - У системі «MEDос» є довідка з роботи з будь-яким модулем програми, що допоможе Вам швидко розібратися з призначенням того чи іншого меню.

### 

### 

### Аналіз необхідних технологій

Для вибору технлогічного стеку були враховані такі фактори:

1. Безкоштовні рішення з відкритим кодом для зменшення вартості утримування
2. Спрощення та здешевшення змін в бізнеслогіку
3. Наявність людських ресурсів, та зменшення часу на старт проекта
4. Попередній досвід такого типу проектів.

|  |  |  |
| --- | --- | --- |
| **Частина** | **Технологія** | **Мотивація** |
| **HTML5 UI** | | |
| **Паттерни/Архітектура** | - SPA Шапблон  - Flux Architecture | - Можливість будувати “річ” HTML додатки. |
| **Відображення** | - ReactJS | - Швидка Virtual DOM реалізація  - Немає обжень на інших рівнях системи. |
| **Стилі** | - Twitter Bootstrap | - Good basis for application styling |
| **Application** | | |
| **Application Framework** | - ExpressJS | - Default choice for platform  - Good support for JSON EndPoints |
| **Платформа** | - NodeJS | - Simple deployment  - Non-blocking, resource effective execution |
| **Розгортяння** | - Amazon Beanstalk | - Management of resources by AWS  - Simple deployment |
| **Storage** | - MongoLab | - Хмарний хостінг MongoDb |
| **Data Protection** | - Data redundancy using Amazon Multi-AZ | - High availability of data |
| **Cross-cutting concerns** | | |
| **Хостінг** | - Amazon AWS | - Feet current requirements |
| **Hosting OS** | - Amazon Linux/Ubuntu | - Open Source |
| **Monitoring** | - Amazon Cloud Watch | - Default option |
| **Service Communication** | - ZeroMQ | - Fast protocol  - Cross platform (native implementations) |
| **Розгортання рішення** | | |
| **Automated Testing** | - Unit Testing/TDD | - Proven methodology |
| **Continuous Integration** | - TeamCity | - Default choice |
| **Desired State Management** | - Puppet, Chef | - Good selection of recipes  - Supports AWS |

### Аналіз ризиків

Нижче подано перелік ризиків, які можуть появитись при розробці спланованого рішення.

Єдиним важливим недоліком нашого рішення – гнучкість. Так як наше рішення є веб-додатком, то він не може мати такої великої можливості конфігурацій, налаштувань під необхідний режим роботи. Отже, якщо рішення через певний час застаріє і буде потребувати конкретного оновлення і змін, то це забере час і залучення самих розробників, в той час як програми-конкуренти легко налаштовуються навченою людиною.

Також варто звернути увагу на надійність сервера, де буде стояти веб рішення, і його пропускна можливість. Якщо буде збій, то відповідно вся робота зупиниться, в той час як альтернативні програми можуть працювати і без доступу до інтернету. Хоча надійність серверів в даний час дуже висока, але ймовірність такого існує.

Постійний доступ до мережі інтернет. Тут все залежить від провайдера і інших факторів, які можуть впливати на її роботу.

Швидкість роботи з базою даних і доступ до неї. Звичайно, що локальне розміщення на комп’ютері матиме переконливу перевагу над віддаленим розміщенням, але враховуючи, що локальна обмежується лише одним користувачем, то це можна віднести до невисокого ризику.

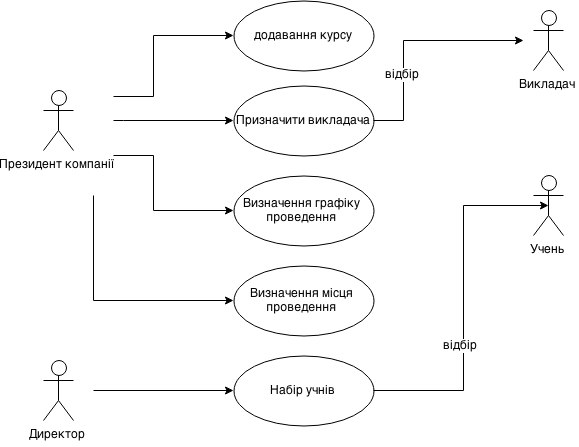
## Схеми варіантів використання ПП

### Таблиця ролей

|  |  |
| --- | --- |
| **Роль** | **Опис** |
| Президент | Власник компанії. Проводить основну роботу по її управлінню. Відкриває нові філії. Затверджує кандидатів на роботу. Дбає про розвиток компанії. Стежить за виконанням сальдо. |
| Директор | Управляє філією. Планує розклади курсів, шукає викладачів, проводить з ними співбесіди. |
| Адміністратор | Закріплений за містом. Оновлює новини на сайті. |
| Викладач | Проводить курси. |
| Клієнт | Навчається на курсах. |

### Use case

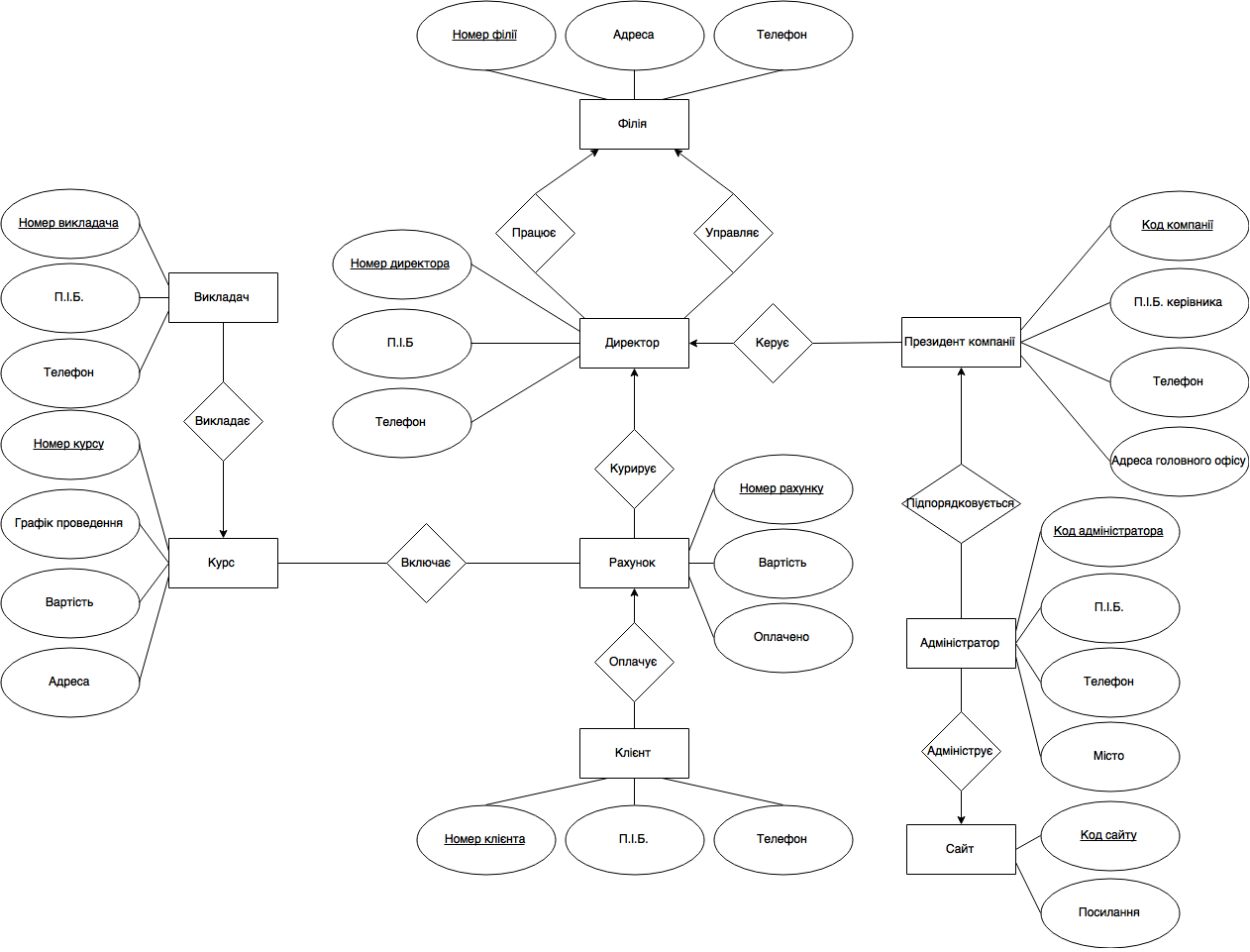
#### По курсах



#### По управлінню компанією



## ER-модель



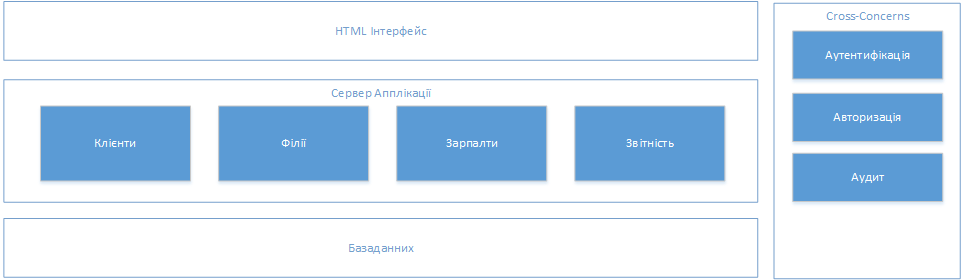
## Архітектура ПП

Високорівнева архітектра програмного забезпечення буде подана у вигляді 4+1 (<http://en.wikipedia.org/wiki/4%2B1_architectural_view_model>).

1. Логічне представлення. Представлення системи з точки зору користувачів;
2. Представлення розробки. Опис моделі даних, компонентні моделі;
3. Представлення процесів. Моделі обробки даних;
4. Фізичне представлення. Способи та засоби розгортання продукту;
5. Високо рівневі сценарії. Опис сценаріїв які впливають на архітектуру.

### Логічне представлення

Ця секція описує архітектуру системи з точки зору користувачів. Подаються основні інтерфейси взаємодії між системою і користувачем.



#### Модулі

**HTML Інтерфейс** -веб інтерфейс користувача для взаємодії з системою.

**Сервер Апплікації** - логічний блок з Бізнес Логікою системи. Складається з таких блоків:

**Клієнти** - логіка клієнтської базиданниї

**Філії** - логіка керування філіями

**Зарпалати** - логіка керування та нарахування зарплати

**Звітність** - модуль з користувальницікою звітністю

**Базаданних** - внтурішня базаданних системи для збереження довідкової, оперативної та історичної інформації.

**Cross-Concerns** - логічний блок з реалізацію не функціональних вимог. Складається з таких модулів:

**Аутентифікація** - реєстрація та ідентифікація користувачів системи. Дозволяє логуватися в систему за допомогою ідентифікатора та пароля.

**Авторизація** - керування правами та граппами користувачів.

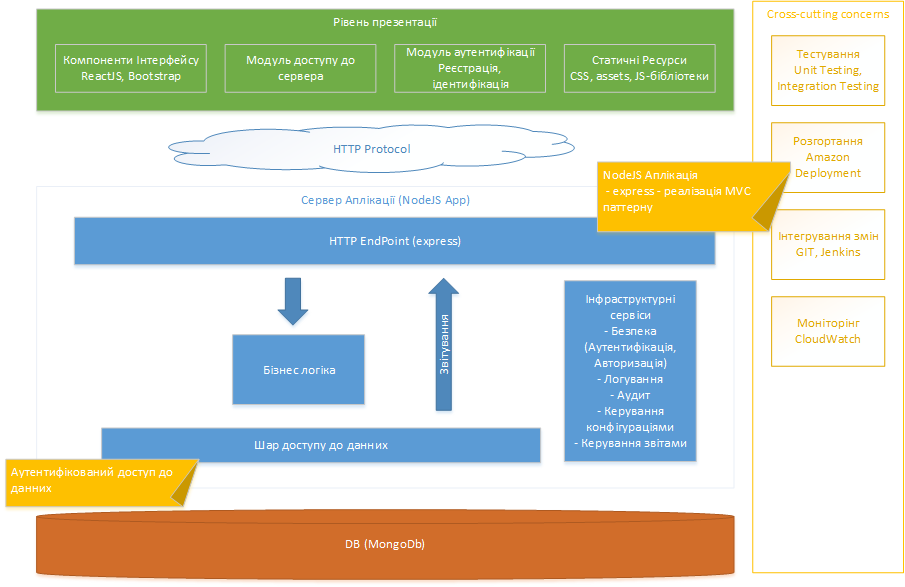
**Аудит** - збереження історії змін в налаштуваннях та данних

### Представлення розробки

Ця секіція описує систему з точки зору розробника. Система розроблема з використанням багато шарового дизайну з деякими виключеннями.

#### Високрівненве представлення

Діаграма описує головні блоки системи і взаємодію між ними.



**Рівень презентаці** - логічний рівень взаємодії з кінцевим користувачем, технічно реалізовано у вигляді тонкого клієнта який виконується в браузері. Використовуються прици Single Page Application (SPA). Для внутрішнього структурування використовуються компоненти, як головна бібліотека для написання компонент використовується ReactJS. Модульність додатку забезпечується requirejs.

**Сервер Апплікації** - фізична группа логічних рівнів, що виконується в контексті серверу апплікації

**HTTP Endpoint** - логічний рівень сервісів апплікації, який надає доступ до бізнес логіки, звітування, та інфраструткруи. Використовує expressjs для роутінга, обробки запитиів і структрування. Використовується JSON як формат данних, HTTP як протокол, Forms Authentication - для аутентифікації.

**Бізнес Логіка** - логічний рівень з імплементацією бізнес процесів, правил, тощо. Використовується підхід схожий на транзакшен скрипти. Данні для прийняття рішен абстраговані за допомогою уніфікованного шару доступу до данних.

**Звітність** - логічний рівень який предоставляє майже прямий доступ до данних, головна вимога це забезпечення безпеки (security trimming) та структурування.

**Шар доступу до данних(DAL) -** логічний рівнеь для уніфікації доступу до данних, представляє маппінг між сутностями бази данних, та сутностями бізнес логіки та звітності.

**Db** - рівень збереження данних. Використовується NoSQL база данних MongoDb.

**Cross-cutting concerns** - логічне группування модулів що перевикористовуютья між іншими рівнями.

### Фізичне представлення

Ця секція описує варіант фізичного розгортання системи. По замовчуванню система розгортається в хмарне середовище Amazon AWS. Для розгортання серверу аплікацій викристовується Amazon Beanstalk, це дає спрощенну модель розгортання і автоматичну підтримку моніторінгу, автоматичного керуваня ресурсами, високої доступності, тощо.

#### 

#### Группи

**HTML UI** - JavaScript SPA вебаплікація у вигляді пакету мінімізованих та оптимізованих HTML, JavaScript, CSS та інших ресурсів. По замовчуванню модуль розгортається як статичні файли на вебсервері, але повністю підтримуються сценарії розгортання на CDN чи в декстопний додаток.

**Група Серверу Апплікацій** - логічна группа серверів апплкації розгорнуті на вебсервері. По замовчуванню використовується розгортання в Amazon Beanstalk, це надає переваги автоматичного керування ресурсами (Elastic Autoscale), моніторінгу (CloudWathc), високої достпності (автоматична підтримка 4 нод), мінімізованного часу простоя (rolling updates), тощо. Ця група через фаервол опублікована в мережу інтернет(DMZ).

**Гупа Базданних** - логічна група баз данних MongoDb, розгорнута на інфарструктурі MongoLabs (<https://mongolab.com/>). Система спроектована з урахзуванням можливості розортання на інших інфраструтурах. Для розробки використовується локальна копія MongoDb. Доступ до цієї групи серверів доступний тільки з сереверу аплікацій, доступ з інтеренту заборонений.

**Група серверів підтримки** - логічна група северів для підтримки роботи системи. Включає в себе збір, аналіз та правила реагування на зміни в системі: перевикористання ресурсів, помилки, недоступність веб серверів, чи серверів баз данних.

# Плани, бюджет

Група складається з трьох осіб та замовника. Склад нашої команди та роль кожного зображено на рис.10 Розподіл обов’язків, етапи розробки і тривалість їх виконання описано на рис.11 На рис.12 зображений календар проекту. Звіт про бюджет – рис.13.1 та 13.2

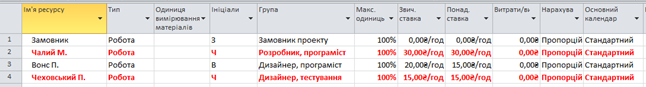
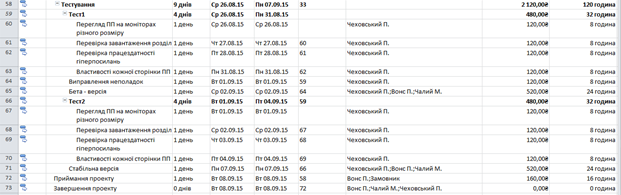
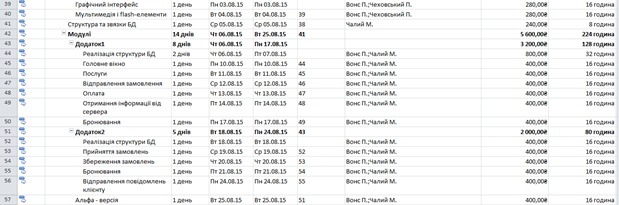
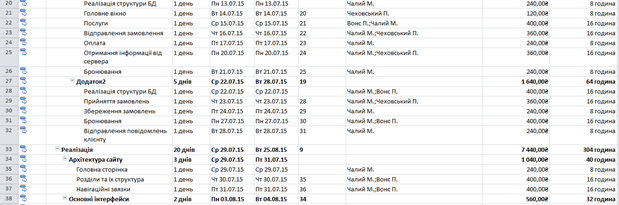
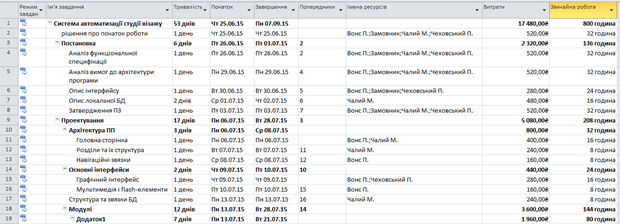
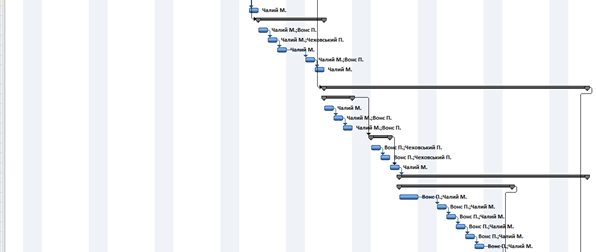
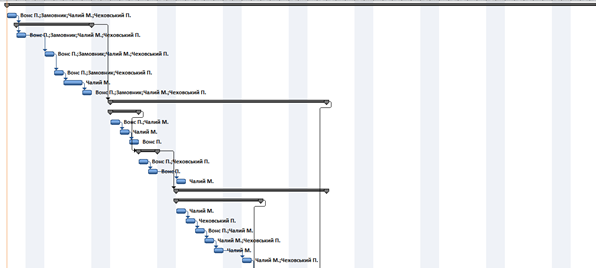
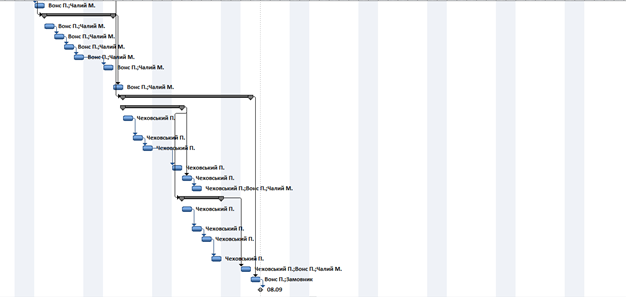


Рис. 10 Склад команди

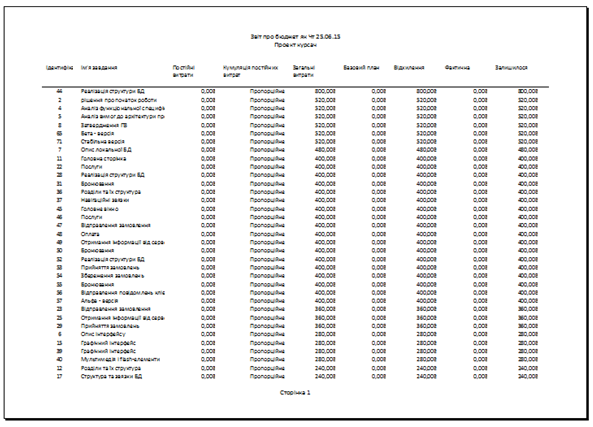


**Рис. 11 Планування проекту**

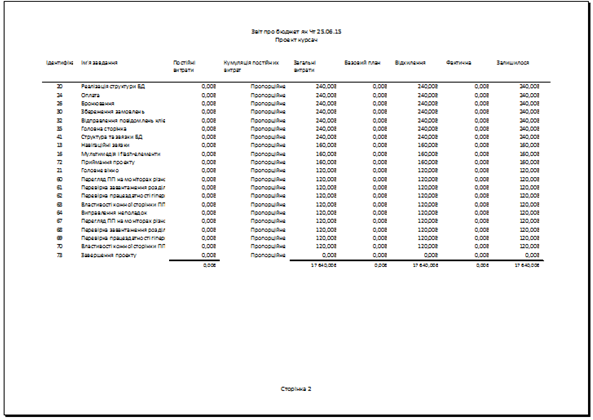




**Рис. 12 Календар проекту**

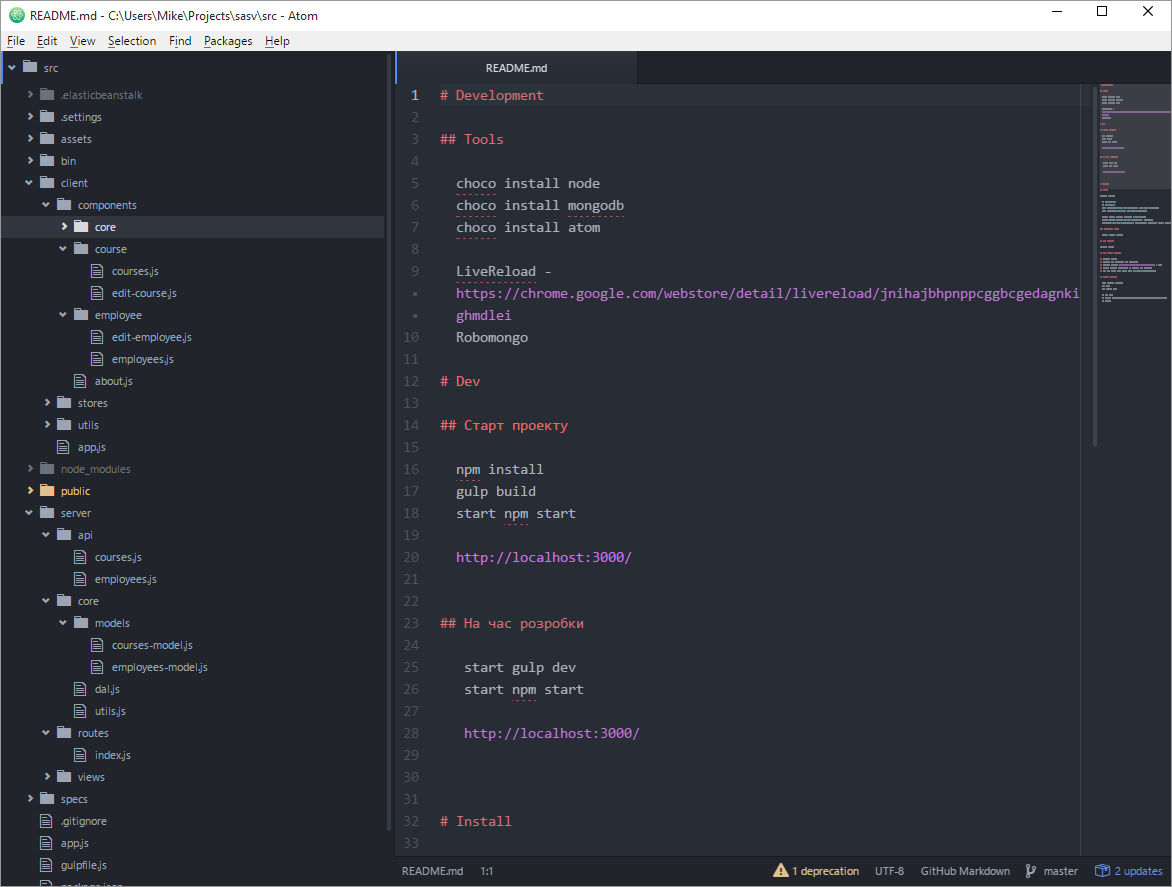


**Рис. 13.1 Звіт про бюджет**



**Рис. 13.2 Звіт про бюджет (продовження)**

# Код програмного забезпечення



### Лістінг dal.js

var assert = require('assert');

var assign = Object.assign || require('object-assign');

var mongo = require('mongodb');

var MongoClient = mongo.MongoClient;

var URL = process.env.DB || 'mongodb://localhost:27017/sasv';

var \_connect = function(cb){

MongoClient.connect(URL, function(err, db) {

assert.equal(err, null);

cb(null, db);

});

};

var \_markUpdated = function(item){

return assign(item, {

updatedAt: new Date()

});

};

var dal = function(n){

return {

collection: function(cb){

\_connect(function(err, con){

assert.equal(err, null);

cb(null, con.collection(n), function(){

con.close();

});

});

},

all: function(filter, cb){

this.collection(function(err, collection, close){

assert.equal(err, null);

var c = collection.find(filter || {});

c.toArray(function(err, items){

assert.equal(err, null);

close();

cb(null, items);

});

});

},

/\*

\* Додати запис в базу данних

\* - Відкривається підключення в БД

\* - Оновлюється поле updatedAt з поточним ервеним часом

\* - Зберігається забис

\*/

add: function(item, cb){

this.collection(function(err, collection, close){

assert.equal(err, null);

var updatedItem = \_markUpdated(item);

collection.insert(updatedItem, function(err, result){

assert.equal(err, null);

close();

cb(null, result);

});

});

},

update: function(id, patch, cb){

this.collection(function(err, collection, close){

assert.equal(err, null);

var updatedItem = \_markUpdated(patch);

collection.update({\_id: mongo.ObjectID(id)}, {$set: updatedItem}, function(err, result){

assert.equal(err, null);

close();

cb(null, result);

});

});

},

remove: function(id, cb){

this.collection(function(err, collection, close){

assert.equal(err, null);

collection.remove({\_id: mongo.ObjectID(id)}, function(err, result){

assert.equal(err, null);

close();

cb(err, result);

});

});

},

merge: function(filter, patch, cb){

this.collection(function(err, collection, close){

assert.equal(err, null);

var updatedItem = \_markUpdated(patch);

collection.update(filter, {$set: updatedItem},

function(err, result){

assert.equal(err, null);

if (result.result.n === 1){

// Запис оновився отже існує, повертаємось

close();

cb(null, result);

} else {

// Запису по фільтру немає, отже додаємо

collection.insert(assign(updatedItem, filter), function(err, result){

assert.equal(err, null);

close();

cb(null, result);

});

}

});

});

},

};

};

module.exports.EmployeesDataAccess = dal('employees');

module.exports.CoursesDataAccess = dal('courses');

module.exports.ObjectID = mongo.ObjectID;

module.exports.ISODate = mongo.ISODate;

### Лістінг employees-model.js

var assign = Object.assign || require('object-assign');

var assert = require('assert');

var utils = require('../utils');

var dal = require('../dal').EmployeesDataAccess;

var \_predefinedEmployees = {

'Паша Чеховський': {position: 'Супер Директор'},

};

var \_getEmployeeNames = function(){

return Object.keys(\_predefinedEmployees);

};

/\*\*

\* Обробка запису збереженного користувачем

\* - Валідація стану сутності

\* - Бізнес правила

\* - Нотифікація зацікавлених сторін

\*\*/

var \_process = function(employee, cb){

cb(null, employee);

};

var EmployeesModel = assign({}, {

/\*\*

\* Список всіх employee

\*\*/

all: function(filter, cb){

dal.all(filter, cb);

},

/\*\*

\* Зберегти зміни в employee

\*\*/

update: function(id, patch, cb){

\_process(patch, function(err, employee){

dal.update(id, employee, cb);

});

},

/\*\*

\* Додати employee

\*\*/

add: function(employee, cb){

\_process(employee, function(err, employee){

dal.add(employee, cb);

});

},

/\*\*

\* Видалити employee

\*\*/

remove: function(id, cb){

dal.remove(id, cb);

},

loadPredefined: function(cb){

utils.forAll(\_getEmployeeNames(),

function(fullName, cb2){

var predefined = \_predefinedEmployees[fullName];

dal.merge({fullName: fullName}, predefined, function(err, result){

cb(err, fullName);

});

},

function(err, fullNames){

assert.equal(err, null);

close();

cb(null, fullNames);

}

);

},

});

exports.EmployeesModel = EmployeesModel;

**Лістінг courses-model.js**

var assign = Object.assign || require('object-assign');

var assert = require('assert');

var utils = require('../utils');

var dal = require('../dal').CoursesDataAccess;

var CoursesModel = assign({}, {

/\*\*

\* Список всіх курсів

\*\*/

all: function(filter, cb){

dal.all(filter, cb);

},

/\*\*

\* Зберегти зміни в курси

\*\*/

update: function(id, course, cb){

dal.update(id, course, cb);

},

/\*\*

\* Додати курс

\*\*/

add: function(course, cb){

dal.add(course, cb);

},

/\*\*

\* Видалити курс

\*\*/

remove: function(id, cb){

dal.remove(id, cb);

},

});

exports.CoursesModel = CoursesModel;

**Лістінг employees.js**

var express = require('express');

var assert = require('assert');

var router = express.Router();

var EmployeesModel = require('../core/models/employees-model').EmployeesModel;

var utils = require('../core/utils');

router.post('/load/', function(req, res) {

var loaders = {

'predefined': EmployeesModel.loadPredefined,

};

var loader = loaders[req.body.type];

if (loader){

loader(function(err, result){

res.status(201).json(result);

});

} else {

res.status(400).json("Loader not found!");

}

});

router.post('', function(req, res) {

EmployeesModel.add(req.body, function(err, result){

res.status(201).json(result);

});

});

router.put('/:id', function(req, res) {

EmployeesModel.update(req.params.id, req.body, function(err, result){

res.status(202).json(result);

});

});

router.delete('/:id', function(req, res) {

EmployeesModel.remove(req.params.id, function(err, result){

res.json({});

});

});

router.get('', function(req, res) {

EmployeesModel.all({}, function(err, employees){

assert.equal(err, null);

res.json(employees);

});

});

module.exports = router;

**Лістінг employees.js**

var express = require('express');

var assert = require('assert');

var router = express.Router();

var CoursesModel = require('../core/models/courses-model.js').CoursesModel;

router.post('', function(req, res) {

CoursesModel.add(req.body, function(err, result){

assert.equal(err, null);

res.status(201).json(result);

});

});

router.put('/:id', function(req, res) {

CoursesModel.update(req.params.id, req.body, function(err, result){

res.status(202).json(result);

});

});

router.delete('/:id', function(req, res) {

CoursesModel.remove(req.params.id, function(err, result){

res.json({});

});

});

router.get('', function(req, res) {

CoursesModel.all({}, function(err, employees){

assert.equal(err, null);

res.json(employees);

});

});

module.exports = router;

**Лістінг app.js**

'use strict';

var React = require('react');

var es6 = require('es6-shim');

var Router = require('react-router');

var RouteHandler = Router.RouteHandler;

var Route = Router.Route;

var ReactBootstrap = require('react-bootstrap');

var Nav = ReactBootstrap.Nav;

var Navbar = ReactBootstrap.Navbar;

var ReactRouterBootstrap = require('react-router-bootstrap');

var NavItemLink = ReactRouterBootstrap.NavItemLink;

var ButtonLink = ReactRouterBootstrap.ButtonLink;

var About = require('./components/about').About;

var Employees = require('./components/employee/employees').Employees;

var Courses = require('./components/course/courses').Courses;

require('./stores/employees-store').EmployeesStore.init();

require('./stores/courses-store').CoursesStore.init();

var App = React.createClass({

render: function() {

return (

<div>

<Navbar brand='Студія Візажу'>

<Nav>

<NavItemLink to="employees" >Співробітники</NavItemLink>

<NavItemLink to="courses" >Курси</NavItemLink>

<NavItemLink to="about">Про программу...</NavItemLink>

</Nav>

</Navbar>

<div className='container'>

<RouteHandler />

</div>

</div>

);

}

});

var routes = (

<Route handler={App} path="/">

<Route name="employees" path="employees" handler={Employees} />

<Route name="courses" path="courses" handler={Courses} />

<Route name="about" path="about" handler={About} />

</Route>

);

Router.run(routes, function (Handler) {

React.render(<Handler/>, document.body);

});

**Лістінг employees-store.js**

var EventEmitter = require('events').EventEmitter;

var assign = require('object-assign');

var server = require('../utils/server');

var CHANGE\_EVENT = 'change';

var \_employees = [];

var EmployeesStore = assign({}, EventEmitter.prototype, {

getEmployees: function(){

return \_employees;

},

init: function(){

this.reload();

},

reload: function(){

var self = this;

server.get('/api/employees/').then(function(result){

\_employees = result.body;

self.emitChange();

});

},

add: function(employee){

var self = this;

return server

.post('/api/employees/', employee)

.then(function(result){

self.reload();

});

},

update: function(id, patch){

var self = this;

return server

.put('/api/employees/' + id + '/', patch)

.then(function(result){

self.reload();

});

},

remove: function(id){

var self = this;

return server

.del('/api/employees/' + id + '/')

.then(function(result){

self.reload();

});

},

emitChange: function() {

this.emit(CHANGE\_EVENT);

},

addChangeListener: function(callback) {

this.on(CHANGE\_EVENT, callback);

},

removeChangeListener: function(callback) {

this.removeListener(CHANGE\_EVENT, callback);

}

});

exports.EmployeesStore = EmployeesStore;

**Лістінг courses-store.js**

var EventEmitter = require('events').EventEmitter;

var assign = require('object-assign');

var server = require('../utils/server');

var CHANGE\_EVENT = 'change';

var \_courses = [];

var CoursesStore = assign({}, EventEmitter.prototype, {

getCourses: function(){

return \_courses;

},

init: function(){

this.reload();

},

reload: function(){

var self = this;

server.get('/api/courses/').then(function(result){

\_courses = result.body;

self.emitChange();

});

},

add: function(course){

var self = this;

return server

.post('/api/courses/', course)

.then(function(result){

self.reload();

});

},

update: function(id, patch){

var self = this;

return server

.put('/api/courses/' + id + '/', patch)

.then(function(result){

self.reload();

});

},

remove: function(id){

var self = this;

return server

.del('/api/courses/' + id + '/')

.then(function(result){

self.reload();

});

},

emitChange: function() {

this.emit(CHANGE\_EVENT);

},

addChangeListener: function(callback) {

this.on(CHANGE\_EVENT, callback);

},

removeChangeListener: function(callback) {

this.removeListener(CHANGE\_EVENT, callback);

}

});

exports.CoursesStore = CoursesStore;

**Лістінг about.js**

var React = require('react');

var About = React.createClass({

render: function() {

return <div className='jumbotron'>

<h1>КНз-31</h1>

<p>Павло Чеховський, Петро Вонс, Михайло Чалий</p>

</div>;

}

});

exports.About = About;

**Лістінг employees.js**

var assign = Object.assign || require('object-assign');

var React = require('react');

var ReactBootstrap = require('react-bootstrap');

var Button = ReactBootstrap.Button;

var ButtonToolbar = ReactBootstrap.ButtonToolbar;

var ButtonGroup = ReactBootstrap.ButtonGroup;

var ModalTrigger = ReactBootstrap.ModalTrigger;

var DropdownButton = ReactBootstrap.DropdownButton;

var MenuItem = ReactBootstrap.MenuItem;

var Panel = ReactBootstrap.Panel;

var Glyphicon = ReactBootstrap.Glyphicon;

var Griddle = require('griddle-react');

var EmployeesStore = require('../../stores/employees-store').EmployeesStore;

var EditEmployee = require('./edit-employee').EditEmployee;

var Employees = React.createClass({

getInitialState: function(){

return {

employees: EmployeesStore.getEmployees()

};

},

componentDidMount: function() {

EmployeesStore.addChangeListener(this.\_onStoreChange);

},

componentWillUnmount: function() {

EmployeesStore.removeChangeListener(this.\_onStoreChange);

},

render: function() {

var owner = this;

var columnMetadata = [

{

columnName: 'fullName',

displayName: 'Ім\'я'

},

{

columnName: '\_id',

displayName: '',

customComponent: React.createClass({

render: function(){

this.modal = <EditEmployee

\_id={this.props.rowData.\_id}

fullName={this.props.rowData.fullName}

onSubmit={this.\_handleSubmitClick}

onRequestHide={owner.\_handleEditCancel}

/>;

return <ButtonGroup>

<ModalTrigger modal={this.modal}>

<Button bsStyle='primary'><Glyphicon glyph='edit' /></Button>

</ModalTrigger>

<Button onClick={this.\_handleRemoveClick}><Glyphicon glyph='remove' /></Button>

</ButtonGroup>

},

\_handleRemoveClick: function(e){

owner.\_handleRemoveSubmit(this.props.rowData.\_id);

},

\_handleSubmitClick: function(data){

owner.\_handleUpdateSubmit(this.props.rowData.\_id, data);

},

})

}

];

var columnsToShow = ['fullName', '\_id'];

return <div>

<h2>Співробітники</h2>

<Panel>

<ButtonGroup>

<ModalTrigger modal={<EditEmployee onSubmit={this.\_handleAddSubmit} onRequestHide={this.\_handleEditCancel} />}>

<Button bsStyle='primary'>Add</Button>

</ModalTrigger>

</ButtonGroup>

</Panel>

<Griddle results={this.state.employees} columnMetadata={columnMetadata} columns={columnsToShow} resultsPerPage={60} useFixedLayout={false} />

</div>;

},

\_handleEditCancel: function(e){

},

\_handleAddSubmit: function(data){

EmployeesStore.add(data);

},

\_handleUpdateSubmit: function(id, data){

EmployeesStore.update(id, data);

},

\_handleRemoveSubmit: function(id){

EmployeesStore.remove(id);

},

\_onStoreChange: function(){

this.setState({

employees: EmployeesStore.getEmployees()

});

}

});

exports.Employees = Employees;

**Лістінг edit-employees.js**

var React = require('react');

var ReactBootstrap = require('react-bootstrap');

var Input = ReactBootstrap.Input;

var ButtonToolbar = ReactBootstrap.ButtonToolbar;

var Button = ReactBootstrap.Button;

var Modal = ReactBootstrap.Modal;

var FormData = require('react-form-data');

var assign = Object.assign || require('object-assign');

var EditEmployee = React.createClass({

mixins: [ FormData ],

getInitialState: function(){

return {

};

},

render: function() {

return <Modal title='Редагувати співробітника'

bsStyle='primary'

onRequestHide={this.props.onRequestHide}

animation={true}>

<div className='modal-body'>

<form onChange={this.updateFormData}>

<Input name='fullName' type='text' label='Ім`я' placeholder='Введіть Ім`я' defaultValue={this.props.fullName} />

</form>

</div>

<div className='modal-footer'>

<Button onClick={this.props.onRequestHide}>Відмінити</Button>

<Button bsStyle='primary' onClick={this.\_handleSubmit}>Записати</Button>

</div>

</Modal>;

},

\_handleSubmit: function(e) {

if (this.props.onSubmit) {

this.props.onSubmit(this.formData);

}

this.props.onRequestHide();

}

});

exports.EditEmployee = EditEmployee;

**Лістінг edit-course.js**

var React = require('react');

var ReactBootstrap = require('react-bootstrap');

var Input = ReactBootstrap.Input;

var ButtonToolbar = ReactBootstrap.ButtonToolbar;

var Button = ReactBootstrap.Button;

var Modal = ReactBootstrap.Modal;

var FormData = require('react-form-data');

var assign = Object.assign || require('object-assign');

var EditCourse = React.createClass({

mixins: [ FormData ],

getInitialState: function(){

return {

};

},

render: function() {

return <Modal title='Редагувати курс'

bsStyle='primary'

onRequestHide={this.props.onRequestHide}

animation={true}>

<div className='modal-body'>

<form onChange={this.updateFormData}>

<Input name='title' type='text' label='Назва курсу' placeholder='Введіть назву курсу' defaultValue={this.props.title} />

</form>

</div>

<div className='modal-footer'>

<Button onClick={this.props.onRequestHide}>Відмінити</Button>

<Button bsStyle='primary' onClick={this.\_handleSubmit}>Записати</Button>

</div>

</Modal>;

},

\_handleSubmit: function(e) {

if (this.props.onSubmit) {

this.props.onSubmit(this.formData);

}

this.props.onRequestHide();

}

});

exports.EditCourse = EditCourse;

**Лістінг courses.js**

var assign = Object.assign || require('object-assign');

var React = require('react');

var ReactBootstrap = require('react-bootstrap');

var Button = ReactBootstrap.Button;

var ButtonToolbar = ReactBootstrap.ButtonToolbar;

var ButtonGroup = ReactBootstrap.ButtonGroup;

var ModalTrigger = ReactBootstrap.ModalTrigger;

var DropdownButton = ReactBootstrap.DropdownButton;

var MenuItem = ReactBootstrap.MenuItem;

var Panel = ReactBootstrap.Panel;

var Glyphicon = ReactBootstrap.Glyphicon;

var Griddle = require('griddle-react');

var CoursesStore = require('../../stores/courses-store').CoursesStore;

var EditCourse = require('./edit-course').EditCourse;

var Courses = React.createClass({

getInitialState: function(){

return {

courses: CoursesStore.getCourses()

};

},

componentDidMount: function() {

CoursesStore.addChangeListener(this.\_onStoreChange);

},

componentWillUnmount: function() {

CoursesStore.removeChangeListener(this.\_onStoreChange);

},

render: function() {

var owner = this;

var columnMetadata = [

{

columnName: 'title',

displayName: 'Назва курсу'

},

{

columnName: '\_id',

displayName: '',

customComponent: React.createClass({

render: function(){

this.modal = <EditCourse

\_id={this.props.rowData.\_id}

title={this.props.rowData.title}

onSubmit={this.\_handleSubmitClick}

onRequestHide={owner.\_handleEditCancel}

/>;

return <ButtonGroup>

<ModalTrigger modal={this.modal}>

<Button bsStyle='primary'><Glyphicon glyph='edit' /></Button>

</ModalTrigger>

<Button onClick={this.\_handleRemoveClick}><Glyphicon glyph='remove' /></Button>

</ButtonGroup>

},

\_handleRemoveClick: function(e){

owner.\_handleRemoveSubmit(this.props.rowData.\_id);

},

\_handleSubmitClick: function(data){

owner.\_handleUpdateSubmit(this.props.rowData.\_id, data);

},

})

}

];

var columnsToShow = ['title', '\_id'];

return <div>

<h2>Перелік всіх курсів</h2>

<Panel>

<ButtonGroup>

<ModalTrigger modal={<EditCourse onSubmit={this.\_handleAddSubmit} onRequestHide={this.\_handleEditCancel} />}>

<Button bsStyle='primary'>Add</Button>

</ModalTrigger>

</ButtonGroup>

</Panel>

<Griddle results={this.state.courses} columnMetadata={columnMetadata} columns={columnsToShow} resultsPerPage={60} useFixedLayout={false} />

</div>;

},

\_handleEditCancel: function(e){

},

\_handleAddSubmit: function(data){

CoursesStore.add(data);

},

\_handleUpdateSubmit: function(id, data){

CoursesStore.update(id, data);

},

\_handleRemoveSubmit: function(id){

CoursesStore.remove(id);

},

\_onStoreChange: function(){

this.setState({

courses: CoursesStore.getCourses()

});

}

});

exports.Courses = Courses;

Повну версію кода можна знайти на Github - <https://github.com/chaliy/sasv>

# Тестування

*Тестування* – процес направлений на виявлення дефектів і помилок в програмному продукті шляхом пошуку невідповідностей між очікуваним та отриманим результатом. Процес тестування не передбачає аналіз отриманих проблем.

Планується три етапи проведення процесу тестування. **Перший етап** полягає в аналізі ТЗ, складанні критичного звіту та тест плану, а також часткового прогону функціональних тестів. **Другий етап** буде присвячений деталізації функціонального звіту і детальному прогону функціональних тестів з виявленням і описом дефектів. На **третьому етапі** буде вироблено стрес-тестування з описом знайдених дефектів. Таким чином, досягається максимальна деталізація глибини тестування, що, в свою чергу, дозволяє більш точно визначити витрачаються ресурси, а так само дозволяє розробникам проекту почати виправляти дефекти на самих ранніх етапах.

Зважаючи на відмову від ведення дефектів, всі виявлені дефекти будуть передаватися менеджерам проекту в письмовому вигляді по електронній пошті.

На першому етапі буде застосований Smoke Testing при якому будуть уточнюватися вимоги, визначаться тестові середовища. До початку другого етапу будуть сформований критичний звіт, а так само звіт за функціональним тестування та історія користувача. На другому етапі проводиться детальне тестування функціоналу проекту, збираються і описуються дефекти. Третій етап завершує роботу з тестування . У ньому проводиться встановлений набір тестів для виявлення вразливих місць.

Операційні системи затвердженні при тестуванні:

1. Windows Vista

2. Windows XP

3. Windows 7

4. Windows 8

# Скріншоти робочої програми

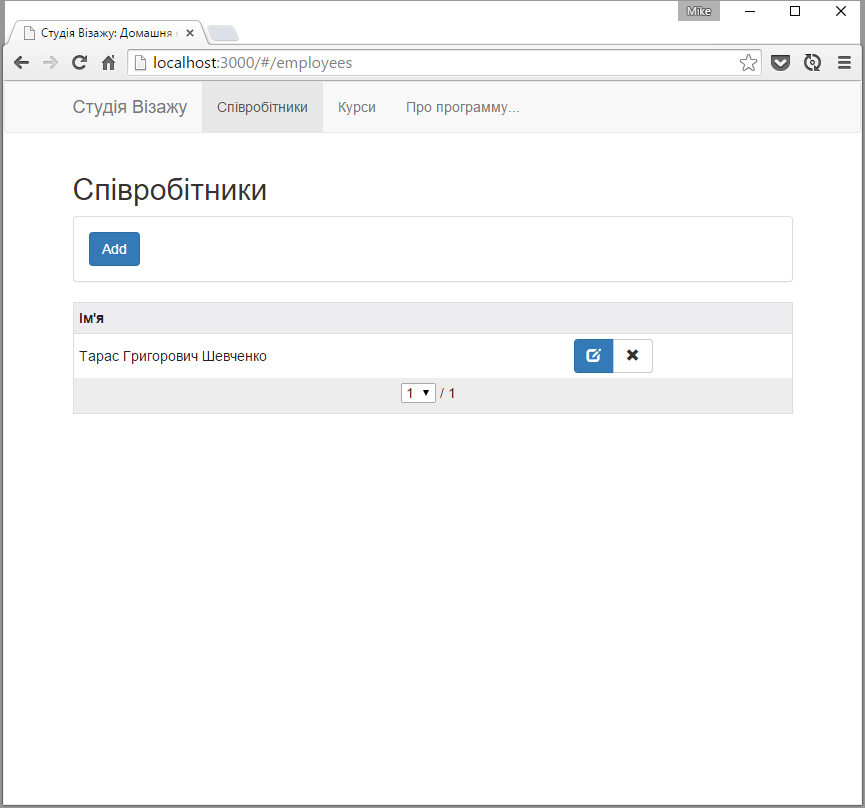


Рис 1. Список співробітників.

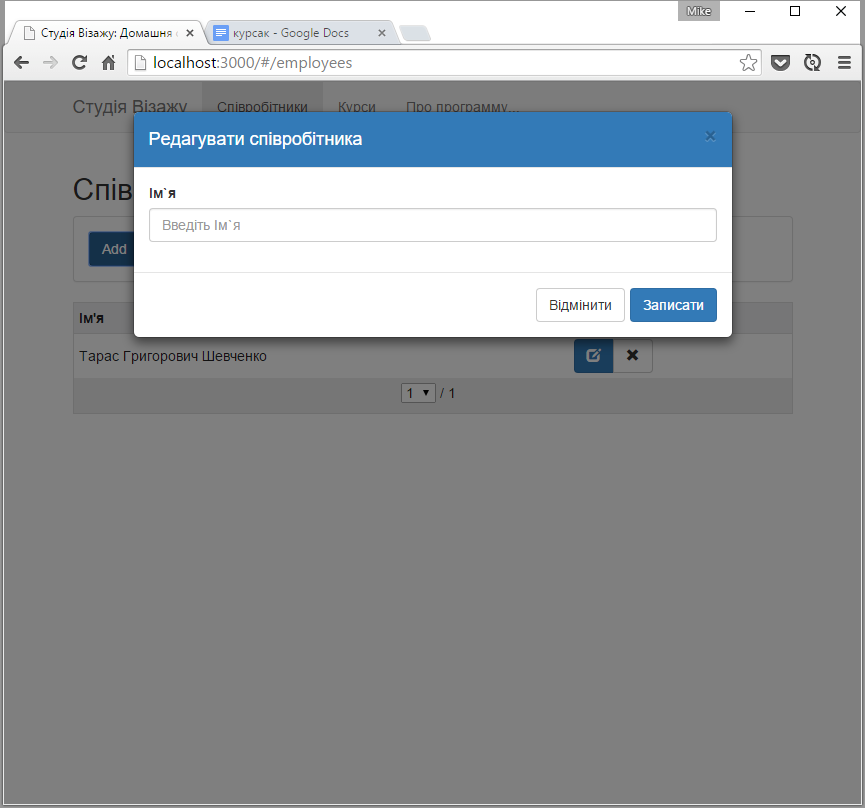


Рис 2. Додавання співробітника

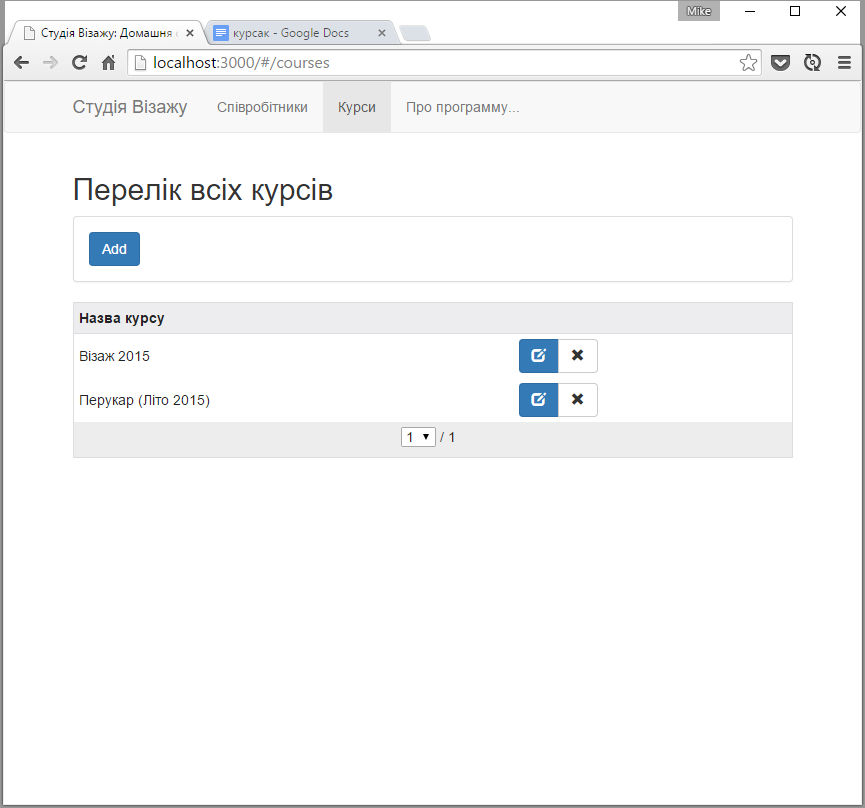


Рис 3. Списко курсів

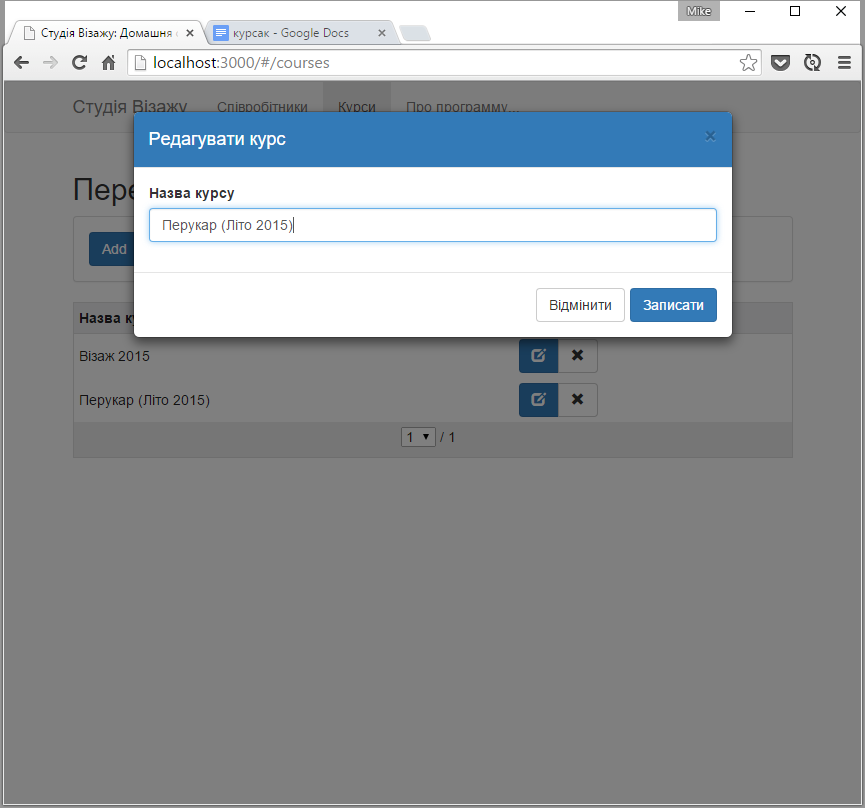


Рис 4. Редагування курсу

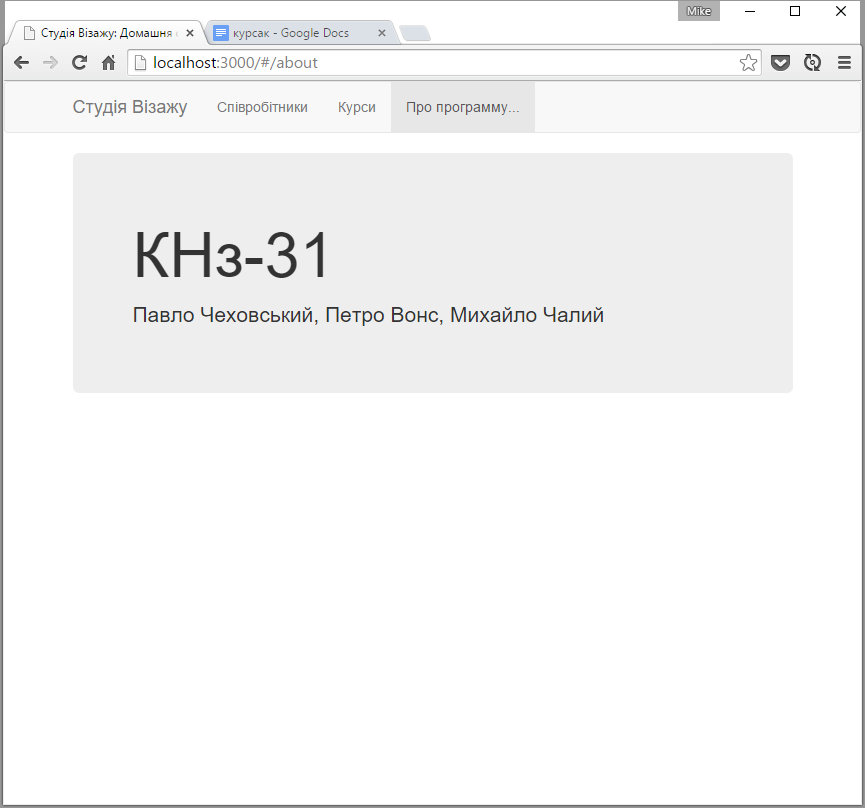


Рис 5. Про программу

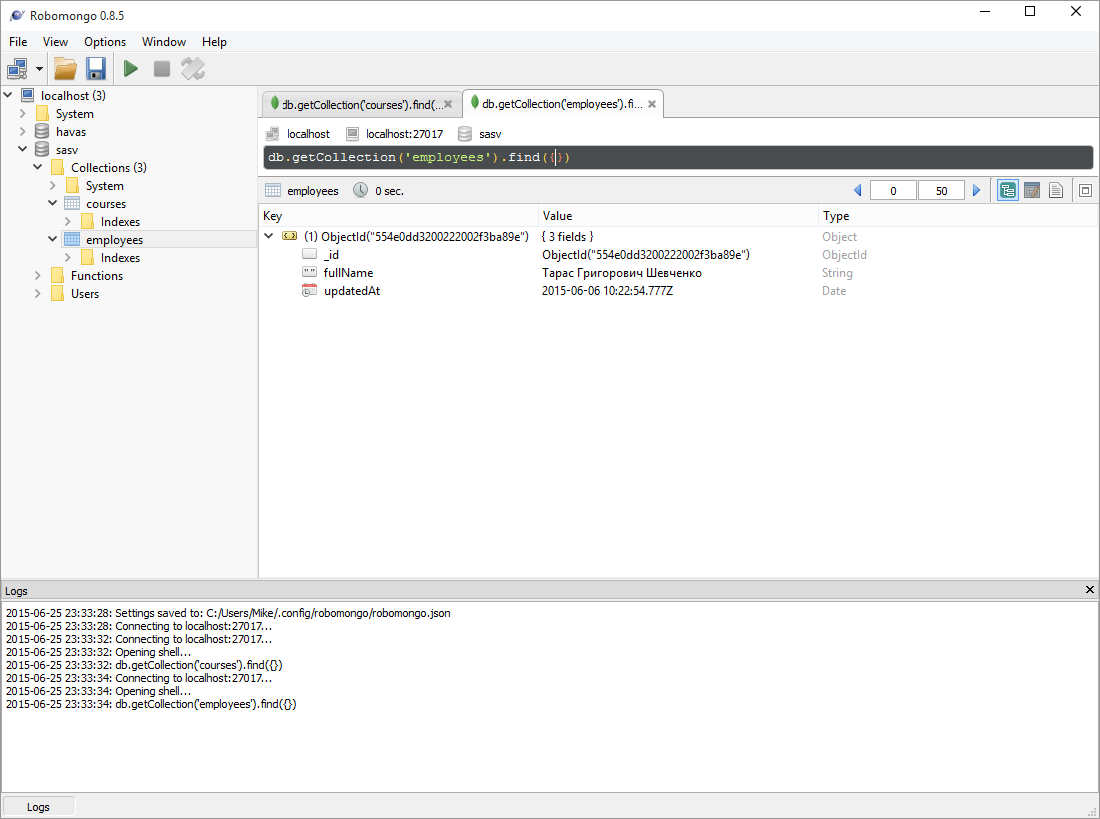


Рис 6. Менеджер бази данних

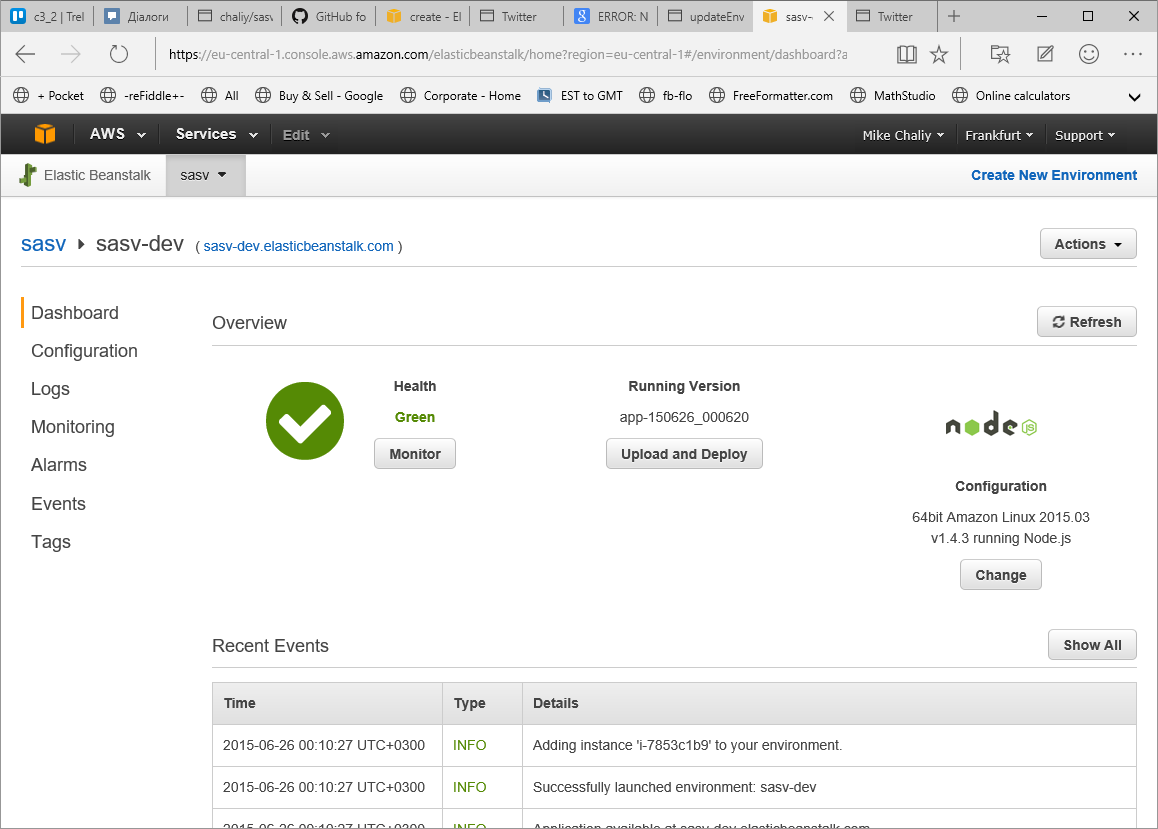


Рис 7. Дашбоард розгорнутого додатку на Amazon Beanstalk

**Висновок**

Виконуючи цей курсовий проект ми, командою з 3 чоловік, працювали над розробкою програмного продукту – Створення сайту (веб-додатку). Навчилися працювати в команді, прислуховуватись до порад, приймати спільні рішення, ділити обов’язки, бути відповідальними за свою частину роботи. Програма, яку ми створили, не є ідеальною (завжди є до чого прагнути і куди розвиватись), знаходиться на стадії розробки, але є досить зручною у користуванні, розрахована на будь-якого клієнта і працівника студії візажу..

Робота над програмним продуктом була доволі цікавою та пізнавальною. Сподіваємося, скоро він стане у пригоді багатьом людям.

**Використана література:**

1. <http://www.intelsd.com/?tc=175&sc=197&lvl=2>

2. <http://bridge.tneu.edu.ua/web_info/lectures/part8/08-05.html>

3. Т.С. Карпова. – "Базы данных: Модели, разработка, реализация" СПб.: Питер, 2002. – 304 с.: ил.

4. Петров В.Н. – "Информационные системы" СПб.: Питер, 2003. – 688 с.: ил.

5.<http://pidruchniki.ws/1701120547727/informatika/modeli_zhittyevogo_tsiklu>

6. <http://www.ex.ua/10311109>

7. <http://lib.mdpu.org.ua/e-book/vstup/L8.htm>

8. <https://ru.wikipedia.org/wiki/Microsoft_Solutions_Framework>

9. http://krasaschool.lviv.ua/school.htm