МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра Програмної інженерії

ЗВІТ

З НАУКОВО-ДОСЛІДНОЇ ПРАКТИКИ

Місце проходження практики «ІРЕ НАНУ»

у період з "22" січня 2018 р. по "14" квітня 2018 р.

Тема індивідуального завдання:

«\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

|  |  |
| --- | --- |
| Студент ПЗСм-16-1  Шпетний Д.В.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис) | Керівник практики  ст. викл. Трощило О.С.  Рекомендована оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2018р. |

Керівник атестаційної роботи: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

доц. Турута О.П. (оцінка)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

Комісія з прийому практики: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ проф. Четвериков Г.Г.

(підпис)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ доц. Турута О.П.

(підпис)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ доц. Ревенчук І.А.

(підпис)

Харків 2018

РЕФЕРАТ

Звіт з науково-дослідної практики: 21 сторінки, 3 розділів, 3 рисунків, 10 джерел.

Метою проходження практики є аналіз предметної області, аналіз можливостей сучасних технологій автономного керування транспортним засобом, постановка задачі та аналіз технологій розробки програмного комплексу для моделювання і дослідження дорожньо-транспортного середовища у контексті дослідження можливостей автономного керування.

Обрані методи розробки базуються на технологіях C#, Unity, Python.

У результат проходження практики було проведено аналіз предметної області та існуючих систем, порівняні та обрані технології розробки програмного забезпечення.

МОДЕЛЮВАННЯ СЕРЕДОВИЩА, АВТОНОМНИЙ ТРАНСПОРТ, МАШИННЕ НАВЧАННЯ, МАШИННИЙ ЗІР.

Report on research practice: 21 pages, 3 sections, 3 figures, 10 sources.

The aim of the practice is to analyze the subject area, analyze autonomous vehicle systems, formulate the problem and analyze the technologies of developing a software system for modeling and analysis of a urban road environment for research of autonomous driving.

The development methods are based on C#, Unity, Python.

As a result of the course of practice, the subject area and autonomous driving systems were analyzed, the software development technologies were compared and selected.

ENVIRONMENT MODELING, AUTONOMOUS VEHICHLE, MACHINE LEARNING, COMPUTER VISION.

ЗМІСТ

|  |  |
| --- | --- |
| Вступ | 4 |
| 1 Аналіз предметної області та постановка задачі | 6 |
| 1.1 Аналіз предметної області розробки | 6 |
| 1.2 Аналіз можливостей існуючих систем | 8 |
| 1.3 Постановка задачі |  |
| 2 Перелік вимог до програмної системи |  |
| 2.1 Призначення розробки |  |
| 2.2 Вимоги до програмного продукту |  |
| 2.3 Вимоги до клієнта |  |
| 3 UML моделювання та вибір технологій розробки |  |
| 3.1 UML – моделювання програмної системи |  |
| 3.2 Аналіз та вибір технологій розробки |  |
| Висновки |  |
| Перелік посилань |  |

ВСТУП

Людство завжди прагне до автоматизації повсякденних завдань. Користування індивідуальними засобами транспорту для багатьох є необхідністю. Розробники автомобілів ставлять за одну з найважливіших цілей підвищення комфорту та безпеки керування. Для цього постійно ведуться роботи над покращенням систем контролю та створення автоматичних допоміжних систем для водія (адаптивне керування, режими круїзної подорожі та інше).

Наступним етапом автоматизації стає усунення водія як основного центру виконування механічних дій для керування автомобілем і перетворення його ролі на контролюючу та спрямовуючу – задання маршруту, обрання типу пересування (пріоритет швидкості, комфорту, безпеки тощо) та вибір інших параметрів вищого рівня абстракції. Над проектом автономного керування працюють усі визначні автомобільні компанії (Toyota, Ford, Volkswagen, Tesla Motors) та гіганти розробки програмного забезпечення (Google, Samsung, NVidia). Однак результати їх роботи не можуть бути використаними у серійних автомобілях зараз у зв’язку з юридичними складностями та неповноцінністю систем.

Основні проблеми у дослідженні автономного пересування у мережах міста є безпосередньо неможливість навчати і тестувати систему фізично у самому місті. Підготовка та операція навіть над єдиним екземпляром автомобілю потребує значних кошт та кваліфікацій у суміжних областях діяльності, що є неможливим для невеликих груп дослідників. Один екземпляр автомобілю надає недостатньо даних для збору та обробки необхідних для покриття усіх крайових сценаріїв. Це стає очевидним навіть при типово алгоритмічному підході до задачі, не кажучи про більш вимогливу до кількості даних концепцію машинного навчання.

На кафедрі програмної інженерії університету постійно ведуться дослідження у галузях машинного зору, штучного інтелекту та моделювання складних мультиагентних систем, що дає достатню теоретичну базу для виконання даного дослідження.

Метою роботи є дослідження існуючих систем автономного пересування, транспорту та прийняття рішень задля покращення існуючих способів аналізу дорожньо-транспортної ситуації у системах автономного автомобілю завдяки використання додаткових груп сенсорів та новаторських підходів у аналізі сукупності даних. Метою практики є дослідження оптимального середовища та технологій для подальшої роботи над обраною метою дослідження.

Інтеграція систем автоматичного керування є небезпечною у першу чергу завдяки тому що неможливо передбачити та опрацювати усі сценарії поведінки звичним алгоритмічним шляхом, тому є раціональним використовувати деякі практики машинного навчання та впровадження базових концепцій штучного інтелекту, які сформують модель поведінки, що емулює дії професійного водія.

Об’єктом дослідження є автономний автомобіль та створена програмна симуляція обраних транспортних ситуацій. У ході дослідження були використані емпіричні, експериментальні методи дослідження у сукупності з абстрактно-формальним моделюванням середовища. Отримані результати удосконалюють роботу аналогічних систем контролю та аналізу середовища згідно обраним формальним критеріям. Дані наукові інновації можуть бути використаними для поліпшення існуючих систем аналізу та прийняття рішень у автоматизації керування, або інтегровані у автомобільні допоміжні підсистеми контролю та аналізу руху.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області розробки

Автономний автомобіль складається з безпосередньо транспортного засобу, який виконує роль основної платформи, комплексу сенсорів різного типу та програмної системи, що виконує роль агрегатору даних та приймає рішення на базі наявних даних [1]. Приклад такого транспортного комплексу наведений у схематичному вигляді на рисунку 1.1.



Рисунок 1.1 – Схематичний приклад автономного автомобілю

Традиційним підходом до вибору сенсорів є наявність камер та візуальних сенсорів різного формату, що надають недостовірні дані по окремості, але їх сукупність дозволяє відтворити більш дійсний склад речей. Розглянемо наявні сенсори та їх можливості у контексті аналізу дорожньо-транспортної ситуації.

Предметна область розробки знаходиться на перетині галузей робототехніки, прийняття рішення, комп’ютерного зору, машинного навчання. У даному дослідженні зроблений акцент на програмних системах, тому деякі деталі з інших галузей подані у спрощеному вигляді або не розглядаються взагалі.

Типовий склад сенсорів у автономному транспортному засобі [1] поданий на рисунку 1.2 у схематичному форматі.

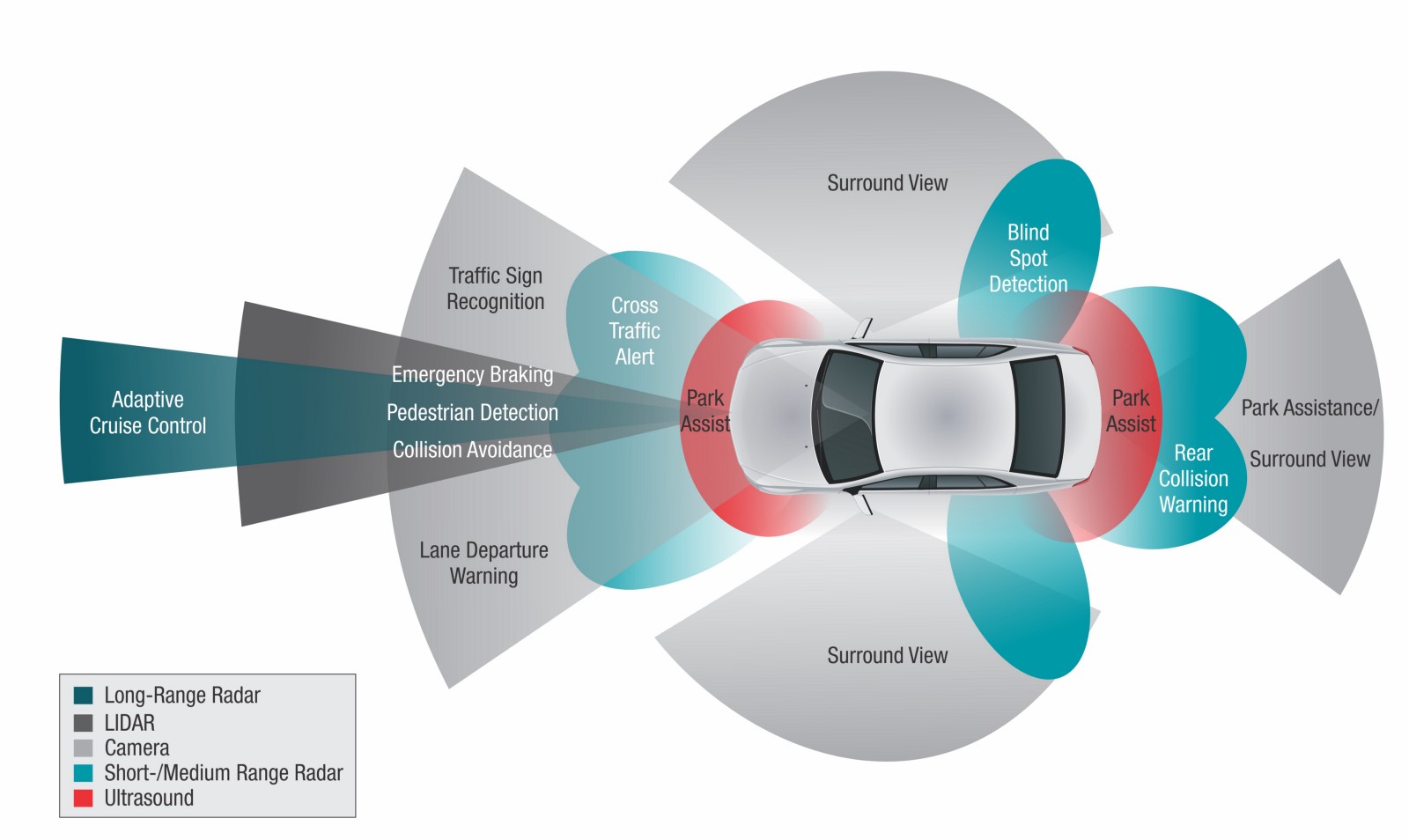


Рисунок 1.2 – Детектори автономного автомобіля

Загалом, можна виділити основні класи проблем, що вирішує кожен с датчиків:

* критичні ситуації – LIDAR, камера, допоміжні радари короткого радіусу дії. До таких ситуацій переважно входять виявлення аномальної поведінки пішоходів або інших динамічних агентів на дорозі таких як автомобілів, тварин, падаючих гілок чи каменів.
* допоміжні інформаційні дані – камери, радари різного радіусу дії, ультразвукові радари. До цього класу можна віднести аналіз типових сліпих зон водія, допомога у аналізі дорожньої ситуації (знаки, світлофори, дорожня розмітка), допомога у парковці.
* системи покращення життя водія – камера, радар дальнього радіусу дії. Переважно це такі системи як круїз контроль, моделювання навколишнього середовища.

LIDAR (Light Identification, Detection and Ranging) – це технологія отримання та обробки інформації про віддалені об'єкти за допомогою активних оптичних систем, що використовують явища відбиття світла і його розсіювання в прозорих і напівпрозорих середовищах. Скануючі лідари в системах машинного зору формують двовимірну або тривимірну картину навколишнього простору [2].

Ключовими з точки зору проектування автономного автомобілю є питання використання даних з сенсорів у реальному-часі, що накладає певні обмеження до складності та оптимізації обчислень. Одними з систем такого роду є комп’ютерні ігри, які можуть складати базу оточення для дослідницьких робот з візуальним зображенням [3].

Окрім безпосередньо потоку інформації також необхідно обробляти його для прийняття рішень у керуванні транспортним засобом, чим займається підсистема аналізу та прийняття рішень.

На сьогодні існує дві основні парадигми для автономних систем водіння на основі комп'ютерного зору: опосередкований підхід до сприйняття (mediated perception approach) щоб розібрати всю дорожню сцену, і підхід рефлекторного аналізу поведінки для прийняти рішення про водіння (behavior reflex approach) яка складає пряму залежність між зображенням та безпосередньою дією завдяки регрессору.

Mediated perception approach складається з багатьох підкомпонентів для розпізнавання об’єктів що відносяться до водіння таких як дорожня розмітка, знаки, лінії руху, світлофори, автомобілі, пішоходи тощо. Результати аналізу складаються у загальну картину безпосереднього оточення автомобілю. Для контролю транспортного засобу система штучного інтелекту бере до уваги всю інформацію перед прийняттям рішення. Оскільки тільки невелика кількість об’єктів має значення у виборі рішення [4], то рівень повного аналізу оточення може додати непотрібний шар складності до і так важкої задачі. На відміну від багатьох задач робототехніки автономний автомобіль контролює лише два параметри – швидкість та напрямок. Остаточний простір рішень має невеликий ступінь свободи на порівняння зі складністю розмірності моделі оточуючого світу. Повнота інформації також надає необхідність використовувати велику кількість сенсорів, що в свою чергу підвищує загальну ціну системи.

Behavior reflex approach складається з прямої залежності між сенсорними вхідними даними та безпосередньо дією автомобіля. Ідея полягає у використанні нейронної мережі для створення прямого мапінгу від зображення до дії. Для навчання використовується реальний водій. Система зберігає зображення та дії водія під час конкретного зображення. Така концепція елегантна у своїй простоті, але вона може зазнавати нижче приведених складностей під час пересування у реальному трафіку та під час виконання складних водійських маневрів. По-перше, водії приймають різні рішення під час однакових ситуацій, що може спричинити розбіжності під час навчання регрессора. Наприклад, якщо автомобіль знаходиться прямо на шляху, то один водій може продовжувати рухатися за машиною, інший – обігнати зліва або справа. Коли всі сценарії присутні у даних, то модель зазнає складностей у прийнятті рішення хоча сенсорні початкові дані однакові. По-друге, система прийняття рішень на такій базі є занадто низькорівневою. Прямий мапінг не може розібратися з повною картиною ситуації. Для прикладу, з точки зору такої моделі обгін автомобілю та пересування назад на полосу є низкою дуже низькорівневих рішень як поворот руля на певний час, а потім поворот у інший бік на певний час. Система не може правильно реагувати у критичних ситуаціях та не може відслідковувати реальний стан речей, які підходять для водіння – необхідний рівень абстракції – це маневр обгону або пересування за автомобілем, а не ступінь повороту руля.

У даній роботі розглядається така модель, що поєднує у собі ці дві концепції. Ця парадигма займає місце між першими двома. Ключовою ідеєю концепції є існування певних станів та індикаторів доступності (affordance indicator) у контексті дорожньої ситуації [5]. Індикатори можуть обиратися в залежності від обраної задачі. Наприклад, дистанція до інших автомобілей, дистанція до дорожньої розмітки тощо.

Важливим компонентом у автономному автомобілі є підхід до штучного інтелекту, який приймає рішення на основі даних. Концептуально можна поділити системи штучного інтелекту на ті що навчаються з учителем (supervised learning) та на самонавчаємі (reinforcement learning).

Традиційним підходом для систем автономного транспорту є використання даних отриманих під контролем людського водія. Це надає змогу отримувати лише коректні дані та спростити кількість необхідного навчання у силу оптимальності контролю автомобіля досвідченим водієм. Недоліками концепції є неможливість моделювання екстремальних ситуацій та можливі протиріччя на великих даних, оскільки кожен водій унікально реагує на дорожню ситуацію. Також процес збору даних накладає потенційні обмеження, оскільки людина вирішує що саме є релевантним у процесі збору вхідних даних.

Навчання з підкріпленням на відміну від першого підходу не потребує людського водія та початкових даних. Основою концепції є встановлення цілей для досягнення та критерії оптимальності їх виконання. Наприклад, для автомобіля це може бути подорож із точки А в точку Б з мінімальною кількістю збитих пішоходів, або з найменшим рівнем витрати пального. Часто використовують комбінацію критеріїв для оптимізації сценаріїв поведінки. Недоліками такого підходу є значно більший час на навчання, візуальні не

1.2 Аналіз можливостей існуючих систем

У даний час існує велика кількість досліджень автономних автомобілей та пов’язаних суміжних галузей, але більшість з них не є комплексними системами або є частиною комерційної таємниці, оскільки перспективи досліджень та потрібні економічні інвестиції недоступні невеликим дослідницьким організаціям. Далі детально розглянемо та проаналізуємо системи, обрані для дослідження.

CRM система (Customer Ralationship Management) – це прикладне програмне забезпечення для організацій, призначене для автоматизації стратегій взаємодії з замовниками (клієнтами), зокрема, для підвищення рівня продажів, оптимізації маркетингу і поліпшення обслуговування клієнтів шляхом збереження інформації про клієнтів та історії взаємин з ними, встановлення і поліпшення бізнес-процесів, подальшого аналізу результатів.

CRM - модель взаємодії, що визначає: центром всієї філософії бізнесу є клієнт, а основними напрямками діяльності є заходи з підтримки ефективного маркетингу, продажу та обслуговування клієнтів. Підтримка цих бізнес-цілей включає збір, збереження та аналіз інформації про споживачів, постачальників, партнерів, а також про внутрішні процеси компанії. Функції для підтримки цих бізнес-цілей включають продажі, маркетинг, підтримку споживачів.

CRM системи класифікуються за функціональними можливостями та за рівнями обробки інформації. За функціональними можливостями вони поділяються на системи управляння продажами, управляння маркетингом, управління клієнтським обслуговуванням. За рівнем обробки інформації йде наступний поділ:

* операційний CRM - реєстрація та оперативний доступ до первинної інформації щодо подій, компаній, проектів, контактів;
* аналітичний CRM - звітність і аналіз інформації в різних розрізах (воронки продажів, аналіз результатів маркетингових заходів, аналіз ефективності продажів в розрізі продуктів, сегментів клієнтів, регіонів і інші можливі варіанти);
* коллаборативний CRM - рівень організації тісної взаємодії з кінцевими споживачами, клієнтами, аж до впливу клієнта на внутрішні процеси компанії (опитування, для зміни якостей продукту або порядку обслуговування, веб-сторінки для відстеження клієнтами стану замовлення, повідомлення по SMS про події, пов'язані із замовленням або особовим рахунком, можливість для клієнта самостійно конфігурувати і замовляти в режимі реального часу продукти та послуги).

Вибір CRM-систему є важкою задачею, зважаючи на їхню різноманітність на ринку. Під час вибору CRM системи треба враховувати усі поточні потреби бізнесу. Детально розглянувши ринку, було виявлено, що найбільш популярними на даний час CRM системами є нижче проаналізовані додатки.

AmoCRM - це хмарний сервіс, формату SFA (Sales Force Automation - система автоматизації продажів). Простий і зрозумілий інтерфейс дозволяє без особливих зусиль налаштовувати і користуватися нею через веб-додаток і з мобільних пристроїв. Сервіс пропонує: створення клієнтської бази, збереження історії взаємодії з кожним клієнтом; контроль потоку заявок, інформація по кожній угоді; аналітика: аналіз продажів, зведений звіт, звіт по співробітниках; API - готові рішення (веб-форми, віджети для сайту і т.д; тайм-менеджмент, планувальник завдань, нагадування, звіти; зручний імпорт і експорт важливих даних.

ZohoCRM - супермен в світі управління взаємовідносинами з клієнтами на 3-х ключових рівнях: маркетинг, продажі, support. Сервіс дозволяє контролювати і автоматизувати основні бізнес-процеси. Функціонал: кроссплатформеність; Інтеграція з Facebook, Twitter, Gmail, WordPress; перевірка відвідувачів сайту і сегментація; відстеження джерел трафіку; історія взаємодій, звіти, повідомлення; управління клієнтською базою; клієнтський сервіс; зручний імпорт / експорт даних; зручний розклад завдань, календар; планування, прогноз, аналіз воронки. ZohoCRM дозволяє зібрати всі важливі дані в одному місці: контакти покупців, їх історію, інформацію про джерела трафіку, кількості продажів і т.д.

Бітрікс24 - зручна, багатофункціональна, а також безкоштовна система, яка вже кілька років не здає позиції в рейтингу CRM-систем. Основний упор у сервісі зроблений саме на управління проектами, завданнями, документацією, звітами, планами. Разом з цим Бітрікс24 непогано справляється з оптимізацією відносин з клієнтами. На відміну від попередніх 2-х систем, Бітрікс розрахований на впровадження у всю компанію, а не тільки в відділ продажів. Основні функції: ведення та управління базою даних клієнтів; фіксація різних взаємодій (дзвінки, листи); робота з документами та фінансами; відстеження замовлень, оплат; кроссплатформеність; внутрішній чат, розклад, календар, постановник завдань, стрічка змін по проектам, база знань, бейджи; управління проектами; email-трекер - збереження історії листування всередині системи; інтеграція з соцмережами; технічна підтримка. Цей сервіс оптимізує взаємодію між різними відділами, автоматизує робочі. В огляді CRM-систем для малого бізнесу, а також для малобюджетних стартапів, Бітрікс24 - безперечний лідер як мінімум по одній важливій причині - сервісом можна користуватися безкоштовно, якщо в ньому задіяно не більше 12 осіб.

Salesforce Sales Cloud - хмарна система планування, управління і контролю бізнес-процесів (відносини з клієнтами, маркетинг, бюджет, продажу, аналітика). Функції: таргетинг відвідувачів; внутрішній онлайн-чат для менеджерів; облік / робота з угодами, оплатами; хмарне сховище даних; управління клієнтською базою; звіти про взаємодію з покупцями; автоматичний розподіл заявок між менеджерами; інтеграція з соцмережами, поштовими сервісами; доступ до сервісу з мобільних пристроїв; призначення пріоритетності заявками; формування робочих завдань, планування.

BaseCRM - система, за допомогою якої можна відслідковувати продажу, угоди, прогнозувати майбутні доходи, складати вичерпні звіти. З іншого боку - це потужна машина для підвищення продуктивності працівників компанії (планинг, KPI, звіти по ефективності, постановка цілей / завдань). А також багаторівневий контроль даних по клієнтській базі (контакти, угоди, листи, документи, історія торкань). Функціонал сервісу: наскрізні звіти про розвиток конкретної угоди; аналіз ефективності співробітників; аналітика воронки продажів; прогнозування динаміки продажів; інформаційні картки клієнтів; вбудована телефонія; робота з документами, фінансами; постановка завдань, календар.

Отже, основні модулі СRM систем можна об’єднати у наступні категорії:

* Customer Service & Support (CSS), або автоматизація служби підтримки та обслуговування клієнтів
* Sales Force Automation (SFA), або автоматизація діяльності продавців;
* Marketing Automation (MA), або автоматизація маркетингу.

Згідно специфіки роботи навчального центру, що з одного боку займається навчальною діяльністю на базі університету, а з іншого діяльність за межами ВУЗу (проведення курсів, трененгів, семінарів, пошук студентів, рекламна діяльність і т.і.), особливу уваг при виборі CRM системи слід звернути на CSS модуль. При аналізі даного модулю у різних вищеперерахованих та інших присутніх на ринку CRM системах, особливу вагу було приділено задачі складання розкладу. Майже усі системи дають таку функцію. Але не було знайдено можливості інтеграції за сторонніми сервісами для розширення цієї саме задачі.

Навчальний центр

Враховуючи специфіку потреб навчального центру OSTPC інтегруватися та розширювати систему розкладу ХРУРЕ CIST та відсутність таких можливостей у готових рішеннях, постає питання про реалізацію власного програмного продукту.

1.3 Постановка задачі

Метою проходження науково-дослідної практики є проведення аналіз предметної області, аналіз підходів та можливостей сучасних CRM систем, UML моделювання та вибір технологій для розробки програмної системи для автоматизації формування розкладу у навчальному центрі, призначеної для додавання пар у розклад, управління завантаженістю лабораторії, спілкуванню зі студентами. Дана система буде реалізована у якості атестаційної роботи магістра.

Після проведення відповідних досліджень, створення програмної системи складається з реалізації наступних задач:

– реєстрація користувачів у системі (враховуючи різні ролі);

– перегляд розкладу занять;

– інтеграція системи з розкладом CIST для відповідних аудиторії, груп та викладачів;

– надання можливостей студенту: переглядати розклад, надсилати повідомлення викладачу ;

– надання можливостей викладачу: переглядати розклад занять аудиторії, надсилати запити на додання пар до розкладу аудиторії (з рядом критеріїв, таких як час проведення, пари, кількість студентів, зайнятість комп’ютерів та ін.), спілкування зі студентами та адміністратором, отримувати сповіщення про накладання пар;

– надання можливості адміністратору: приймати/відхиляти запити на додання пар до розкладу, додавати викладачів до системи, отримувати сповіщення про накладання розкладу;

– можливість отримувати інформацію про стан завантаженості аудиторії для оптимального використання ресурсів.

Для реалізації системи [2] було обрано середовище розробки WebStrom. Так як засіб буде розроблено у вигляді веб-системи це забезпечить кросплатформеність та можливість роботи у будь якій системі – з сімейства Windows або Unix-подібних систем. Мовою написання серверної частини був обраний Node.js з використанням веб-фреймворку express. В якості сховища даних була обрана СУБД PostgreSQL. Для реалізації клієнтської частини обрані технології HTML5/CCS3, TypeScript, з використанням фреймворку Angular5 та бібліотеки AngularMaterial2.

У ході виконання роботи необхідно зробити такі завдання:

– провести аналіз, концептуальне та UML-моделювання предметної галузі;

– провести дослідження існуючих підходів та можливостей систем формування розкладу;

– розробити схему бази даних для зберігання усіх даних користувача.

2 ПЕРЕЛІК ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Призначення розробки

Перед початком роботи над програмною системою необхідно визначитися з вимогами до неї. У загальному випадку під вимогами розуміють сукупність властивостей, які повинна мати система, що реалізується. У ході роботи треба реалізувати програмну систему для полегшення формування розкладу занять у навчальному центрі.

Даний додаток призначений для викладачів навчального центру (на даний час саме для Учбового науково-виробничого центру аутсорсингу). Дана система має сучасний інтерфейс, тому не потребує багато часу для навчання користування, і , як наслідок, навпаки зменшує час, що потрібен користувачу (викладачу) для внесення пар до розкладу навчального центру. Інтегрування розкладу CIST у додаток, слідкування за завантаженістю аудиторії, да можливість максимально оптимізувати роботу у навчальному залі. Сповіщення про накладання пар – допомагає запобігти конфліктам в розкладах. Таким чином зникає необхідність постійного перегляду розкладів аудиторії занять у вузі та ручного порівняння з заняттями у навчальному центрі, що значно полегшує життя викладачеві, зменшуючи кількість зайвої роботи.

2.2 Вимоги до програмного продукту

Загальні відомості про розробку системи полягають у наступному: система формування розкладу занять у навчальному центрі має бути розроблено за допомогою мови програмування Node.js, фреймворку express. та технологій HTML5/CCS3, ТуреScript, з використанням фреймворку AngularJS та бібліотеки Bootstrap. В якості сховища даних була обрана СУБД PostgreSQL [9]. Засіб було вирішено створити у вигляді веб-додатку (сайту), що підтримує роботу з чотирма типами користувачів: викладачем ХНУРЕ, зовнішнім викладачем, студентом, адміністратором. При перегляді сайту, користувач має змогу переглянути розклади аудиторії. Після вводу свого логіну і паролю вчителям та адміністратору надається доступ до повного функціоналу системи.

Основна функціональність даної системи полягає у наступному:

– перегляд розкладу занять в аудиторіях;

– зіставлення розкладу занять університету та навчального центру;

– комунікація зі студентами.

При створенні підсистеми повинно бути взято до уваги:

– зручність використання сайту – інтерфейс має бути зручним та інтуїтивно зрозумілим, не вимагати додаткової підготовки користувачів;

– довідкова інформації щодо використання системи повинна бути розташована у відповідному розділі на сайті.

2.3 Вимоги до клієнта

Система розробляється у вигляді веб-додатку, тож вона не потребує специфічних умов для того, щоб користувач зміг працювати з нею. Серед необхідних вимог для початку роботи можна назвати лише:

– комп’ютер з підключенням до мережі Інтернет;

– наявність програми для перегляду веб-сторінок – браузеру.

3 UML МОЕЛЮВАННЯ ТА ВІБІР ТЕХНОЛОГІЙ РОЗРОБКИ

3.1 UML – моделювання програмної системи

Моделювання розроблюваної системи проводиться з використанням мови UML для побудови діаграм, що допоможуть відобразити функціональність та внутрішню структуру системи. UML - мова графічного опису для об'єктного моделювання в області розробки програмного забезпечення. UML [3] є мовою широкого профілю, це відкритий стандарт, який використовує графічні позначення для створення абстрактної моделі системи, що називається UML-моделлю.

Можна виділити наступні переваги UML:

– UML об'єктно-орієнтований, в результаті чого методи опису результатів аналізу і проектування семантично близькі до методів програмування на сучасних об'єктно-орієнтованих мовах;

– діаграми UML порівняно прості для читання після досить швидкого ознайомлення з його синтаксисом;

– UML розширює і дозволяє вводити власні текстові та графічні стереотипи.

При проектуванні системи були розроблені діаграми наступних типів: діаграма прецедентів та діаграма розгортання.

Діаграма прецедентів описує функціональне призначення системи або те, що система повинна робити. Розробка діаграми має такі цілі:

– визначити контекст предметної області, яка моделюється;

– сформулювати загальні вимоги до функціонального поведінки проектованої системи;

– розробити вихідну концептуальну модель системи для її подальшої деталізації у формі логічних і фізичних моделей.

Суть діаграми прецедентів полягає в наступному. Проектована система представляється у вигляді безлічі сутностей або акторів, що взаємодіють з системою за допомогою варіантів використання. При цьому актором або дійовою особою називається будь-яка сутність, що взаємодіє з системою ззовні

Діаграма прецедентів для даної системи відображена на рисунку 3.1.

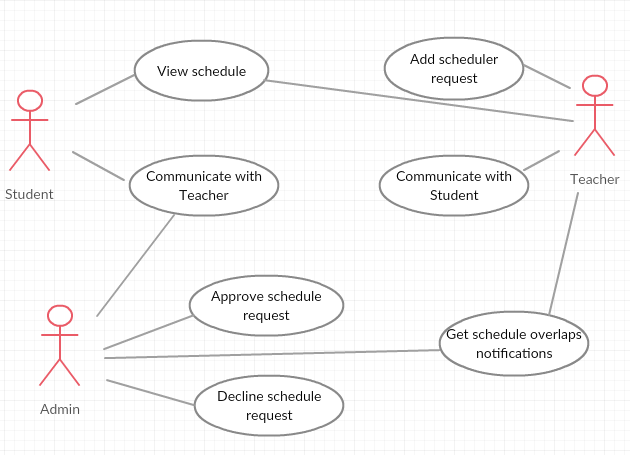


Рисунок 3.1 – Діаграма прецедентів

Діаграма прецедентів моделює систему з точки зору користувача і відображає доступні йому можливості. Діаграми цього типу допомагають при аналізі вимог до системи та проектуванні інтерфейсу користувача.

На рисунку 3.1 зображені основні функції системи, що доступні користувачам. При переході на сайт користувач потрапляє на головну сторінку, де представлена інформація про розклад роботи аудиторій, юзеру також надається можливість увійти у систему для доступу к роле-залежним функціям.

Основний функціонал, доступний користувачу системи включає в себе наступні можливості:

– реєстрація користувачів у системі;

– перегляд розкладу занять в аудиторіях учбового центру;

– інтеграція системи з розкладом CIST для відповідних аудиторії, груп та викладачів;

– надсилати запити на додання пар до розкладу аудиторії (з рядом критеріїв, таких як час проведення, пари, кількість студентів, зайнятість комп’ютерів та ін.);

– отримувати сповіщення про зміни в розкладі та про накладання пар;

– приймати/відхиляти запити на додання пар до розкладу, додавати викладачів до системи;

– надання можливостей комунікацій між студентами та викладачами;

– надання можливостей комунікацій між викладачами та адміністраторами;

– можливість отримувати інформацію про стан завантаженості аудиторії для оптимального використання ресурсів.

Фізичне представлення програмної системи разом з інформацією про те, на якій платформі вона реалізована, відображається на діаграмі розгортання.

Система для формування розкладу занять у навчальному центрі складається з таких основних частин:

– серверу (Node.js);

– веб–кліенту (chromium–based браузери);

– бази даних (PostgreSQL).

Діаграма розгортання – діаграма в UML [4], на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент. Вона застосовується для представлення загальної конфігурації і топології розподіленої програмної системи і містить розподіл компонентів по окремих вузлах системи. Крім того, діаграма розгортання показує наявність фізичних сполук-маршрутів передачі інформації між апаратними пристроями, задіяними в реалізації системи. Діаграма компонентів допомагає виявити вузькі місця системи і реконфігурувати її топологію для досягнення необхідної продуктивності.

Взаємодія компонентів системи представлена на діаграмі розгортання, що зображена на рисунку 3.2. На ній показані всі зв’язки та деякі нефункціональні вимоги, такі як: chromium–based браузери.

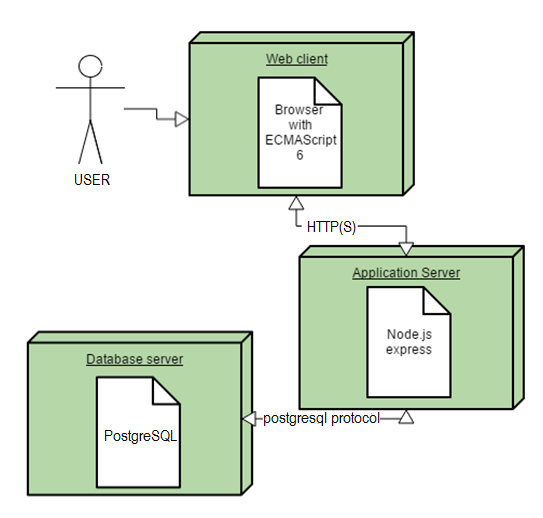


Рисунок 3.2 – Діаграма розгортання

Веб клієнт був побудований за допомогою сучасних технологій HTML5/CCS3, TypeScript, а також з використанням фреймворку Angular5 та бібліотеки AngularMaterial2 [10]. Серверна частина додатку була розроблена за допомогою Node.js з використанням фреймворку express. Node.js був обраний у зв’язку з тим, що розробка як backend частини, так і frontend ведеться однією мово програмування, та дані між частинами додатку передаються у JSON–форматі, що є нативним для JavaScript. В якості сховища даних була обрана СУБД PostgreSQL.

3.2 Аналіз та вибір технологій розробки

Веб клієнт був побудований за допомогою сучасних технологій HTML5/CCS3 [5], TypeScript, а також з використанням фреймворку Angular5 та бібліотеки AngularMaterial2. JavaScript – динамічна, об’єктно–орієнтовна мова програмування. Представляє собою стандарт 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. JavaScript [6] класифікують як прототипну скриптову мову програмування з динамічною типізацією. TypeScript – мова програмування, представлена Microsoft восени 2012; позиціонується як засіб розробки веб-застосунків, що розширює можливості JavaScript. TypeScript є зворотньо сумісним з JavaScript. Фактично, після компіляції програму на TypeScript можна виконувати в будь-якому сучасному браузері або використовувати спільно з серверною платформою Node.js. Нововведення у TypeScript в порівнянні з JavaScript підвищують швидкість розробки, читабельність коду, рефакторинг і повторне використання коду, здійснювати пошук помилок на етапі розробки та компіляції, а також швидкодію програм.

Angular [7] це JavaScript–фреймворк з відкритим програмним кодом, який розробляє Google. Призначений для розробки односторінкових застосунків, що складаються з одної HTML сторінки з CSS і JavaScript. Його мета –розширення браузерних застосунків на основі шаблону Модель–вид–контролер (MVC), а також спрощення їх тестування та розробки. Фреймворк працює зі сторінкою HTML, що містить додаткові атрибути і пов'язує області вводу або виводу сторінки з моделлю, яка являє собою звичайні змінні JavaScript. Значення цих змінних задаються вручну або отримуються зі статичних або динамічних JSON–даних. Двостороннє зв'язування даних в Angular є найбільш примітною особливістю і зменшує кількість коду, звільняючи сервер від роботи з шаблонами. Замість цього, шаблони відображаються як звичайний HTML, наповнений даними, що містяться в області видимості, визначеній у моделі. Сервіс $scope в Angular стежить за змінами в моделі і змінює розділ HTML–вирази в уявленні через контролер. Крім того, будь–які зміни в уявленні відображаються в моделі. Це дозволяє обійти необхідність маніпулювання DOM і полегшує ініціалізацію та прототипування веб–додатків, а також підвищує продуктивність і спрощує тестування.

AngularMaterial2 був обраний як фронтенд фреймворк. Це набір готових компонентів для Angular2+ проектів, а також набір кольорових схем та стилістичних рішень для підвищення швидкості розробки та дотримання єдиного дизайнерського рішення. Цей набір інстрементів створений для полегшення розробки веб-застосунків та сайтів. Він включає CSS та HTML для типографії, форм, кнопок, таблиць, сіток, навігації тощо, а також додаткові розширення. Основними перевагами є наступне: економія часу розробки; масштабованість на різноманітні пристрої та роздільну здатність екрану без жодних змін у розмітці сторінки; усі компоненти платформи використовують єдиний стиль та шаблон за допомогою центральної бібліотеки, що дозволяє узгоджувати дизайн сторінок; простота у використанні фреймворку; сумісність з усіма популярними браузерами.

Серверна частина додатку була розроблена за допомогою Node.js з використанням фреймворку express. Node.js був обраний у зв’язку з тим, що розробка як backend частини, так і frontend ведеться однією мово програмування, та дані між частинами додатку передаються у JSON–форматі, що є нативним для JavaScript.

Node.js – платформа з відкритим сирцевим кодом для написання серверної частини веб–реалізації на JavaScript, автором якої є Раян Дал. Node.js [8] характеризується такими властивостями, як асинхронна однопотокова модель виконання запитів, неблокуючий ввід/вивід, система модулів CommonJS, JavaScript Google V8. Для управління модулями використовується пакетний менеджер npm. Node.js призначений для відокремленого виконання високопродуктивних мережевих застосунків на мові JavaScript. Функції платформи не обмежені створенням серверних скриптів для веб, платформа може використовуватися і для створення звичайних клієнтських і серверних мережевих програм. Для забезпечення обробки великої кількості паралельних запитів Node.js задіює асинхронну модель запуску коду, засновану на обробці подій в неблокуючому режимі і визначенні обробників зворотніх викликів.

В якості сховища даних була обрана СКБД PostgreSQL. PostgreSQL - об'єктно-реляційна система керування базами даних (СКБД). Порівняно з іншими проектами з відкритим кодом, PostgreSQL не контролюється якоюсь однією компанією, її розробка можлива завдяки співпраці багатьох людей та компаній, які хочуть використовувати цю СКБД та впроваджувати у неї найновіші досягнення. Сервер PostgreSQL написаний на мові C. Зазвичай розповсюджується у вигляді набору текстових файлів із сирцевим кодом. Фундаментальна характеристика об'єктно-реляційної бази даних - це підтримка об'єктів і їх поведінки, включаючи типи даних, функції, операції, домени і індекси. Це робить PostgreSQL неймовірно гнучким і надійним. Серед іншого, він вміє створювати, зберігати та видавати складні структури даних. Він забезпечує розширену ємність даних і заслужив довіру дбайливим ставленням до цілісності даних.

Додаток був розміщений на Heroku. Heroku – хмарна PaaS–платформа, що підтримує ряд мов програмування (у тому числі Node.js). Програми, які працюють на Heroku, використовують також DNS–сервер Heroku (зазвичай додатки мають доменне ім'я виду «ім’я\_додатку.herokuapp.com»). Для кожної програми виділяється кілька незалежних віртуальних процесів, які називаються «dynos». Вони розподілені за спеціальною віртуальною сіткою («dynos grid»), яка складається з декількох серверів.

Завдяки використанню адаптивного веб–дизайну, забезпечується коректне відображення веб–сторінок на різних пристроях. Метою адаптивного веб–дизайну є універсальність веб–сайту для різних пристроїв. Для того, щоб веб–сайт було зручно переглядати з пристроїв різних роздільних здатностей та форматів, за технологією адаптивного веб–дизайну не потрібно створювати окремі версії веб–сайту для окремих видів пристроїв. Один сайт може працювати на смартфоні, планшеті, ноутбуці та телевізорі з виходом в інтернет, тобто на всьому спектрі пристроїв.

У результаті була отримана архітектура додатку, що зображена на рисунку 3.3.

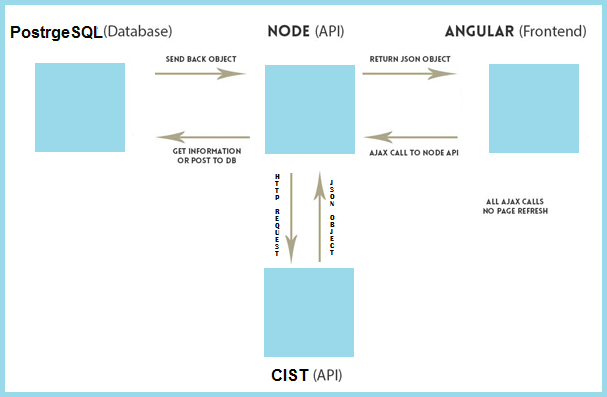


Рисунок 3.3 – Архітектура додатку

Git буде використаний для контрою версій під час розробки додатку. Git – розподілена система керування версіями файлів та спільної роботи. Git є однією з найефективніших систем керування версіями, що надає гнучкі засоби розробки.

ВИСНОВКИ

У ході проходження науково-дослідної практики було проведено аналіз предметної області, порівняні та обрані технології розробки програмного забезпечення для формування розкладу занять у навчальному центрі, якому необхідно базуючись на розкладі пар у певних аудиторіях університету оптимально розподілити ресурси навчальних лабораторій.

На початку роботи було проведено аналіз предметної області, в процесі якого були визначені основні проблеми, що існують в розглянутій галузі. На підставі аналізу предметної області була проведена постановка завдання.

У ході виконання поставленої задачі було:

* проведено аналіз предметної області;
* проведено UML-моделювання предметної галузі;
* проведено аналіз та обрані технології розробки програмної системи для формування розкладу занять навчального центру.

У результаті аналізу потреб було визначено, що кінцевий продукт має наступні функції:

– перегляд розкладу занять в аудиторіях;

– зіставлення розкладу занять університету та навчального центру;

– комунікація зі студентами.

При створенні підсистеми повинно бути взято до уваги:

– зручність використання сайту – інтерфейс має бути зручним та інтуїтивно зрозумілим, не вимагати додаткової підготовки користувачів;

– довідкова інформації щодо використання системи повинна бути розташована у відповідному розділі на сайті.

ПЕРЕЛІК ПОСИЛАНЬ

1. Вільна енциклопедія Википедия [Електронний ресурс] / Мережева енциклопедія Wikipedia, 2000. - Forefront TMG 2010: Режим доступу: : www/ URL : http://ru.wikipedia.org/ . – Загол. з екрану.

2. Фаулер, М., Дейвид, Р. Архитектура корпоративных программных приложений [Текст] / М. Фаулер, Д. Райс; – М.: Вильямс, 2008. – 544 с.

3. Фаулер, М. UML. Основи, 3-є видання: пер. з англ. [Текст] / М. Фаулер. – СПб: Символ-Плюс., 2004. – 192 с. – ISBN 5-93286-060-Х.

4. Діаграма варіантів використання (use case diagram) [Електронний ресурс] / UML Теорія – Режим доступу: http://www.info-system.ru/designing/methodology /uml/theory/use\_case\_diagram\_theory.html/ . – Загол. з екрану.

5. Duckett, J. HTML and CSS: Design and Build Websites. 1st Edition. [Текст] / J. Duckett. – John Wiley & Sons, 2011. – 490 c. – ISBN-13: 978-1118008188.

6. Флэнаган, Д. JavaScript. Подробное руководство [Текст] / Д. Флэнаган. – СПб. Символ-Плюс, 2013. – 1080 с. ISBN 978-5-93286-215-5.

7. Wahling, D. Angular in 60 Minutes [Текст ]/ D. Wahling. – Wahlin Consulting, 2014. – 102c.

8. Документація Nodejs [Електронний ресурс] / Офіційний сайт проекту Nodejs. – Режим доступу: www/ URL: https://wwwnodejs.org/ . – Загол. з екрану.

9. Уорсли, Дж., Дрейк, Дж. PostgreSQL. Для профессионалов [Текст] / Дж. Уорсли, Дж. Дрейк. – СПб: Питер, 2003. – 496 с.

10. Colborne, G. Simple and Usable Web, Mobile, and Interaction Design (Voices That Matter) 1st Edition. [Текст] / G. Colborne. – New Riders, 2010. – 208 c.