НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ Олександр КОВАЛЬ

“___”_____2020р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 122 Комп’ютерні науки та інформаційні технології

На тему Автоматизація бізнес процесів з використанням CRM системи Salesforce

Виконав (-ла): студент (-ка) 4 курсу, групи ТМ-62

_____ _____

(прізвище, ім’я, по батькові) (підпис)

Керівник доцент кафедри, к.в.н. доцент Андрій ОНИСЬКО _____

(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант _____ _____

(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент_____ _____

(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

(підпис)

Київ – 2020 року

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 Комп’ютерні науки та інформаційні технології

Спеціалізація Інформаційні технології моніторингу довкілля

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр КОВАЛЬ
(підпис)
”___”_____2020р.

ЗАВДАННЯ

на дипломну роботу студенту

Новака Михайла Сергійовича
(прізвище, ім’я, по батькові)

1. Тема роботи Автоматизація бізнес процесів з використанням CRM системи Salesforce.
керівник роботи _____ доцент кафедри, к.в.н. доцент Андрій ОНИСЬКО
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”___ 2020 р. № _
2. Строк подання студентом роботи _____
3. Вихідні дані до роботи Вихідні дані системи – це інтерфейс, для студентів – це перегляд особистих даних і успішності в навчанні, для викладачів – це можливість контролю записів успішності студентів.
4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Створена система дозволяє автоматизувати роботу кафедри, позбавити від паперової роботи викладачів, надати студентам доступ до інформації і прогресу навчання в режимі онлайн завдякі хмаровій системі Salesforce, і через це дані доступні в будь-якому місці, в будь-який час.
5. Перелік ілюстративного матеріалу
В дипломній записці зображенно ілюстрації роботи для трьох типів користувачів адміністрації, викладачів та студентів, для адміністрації окремий інтерфейс для маніпулювання даними в системі, для викладачів та студентів сайт який буде доступний за посиланням де є необхідність авторизації, і для студентів та викладачів вкладці персональна інформація за типом користувача визначається його особистий інтерфейс.
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”___”_____202___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи		
2.	Вивчення рекомендованої літератури	25.03.2020	
3.	Аналіз існуючих методів розв’язання задачі.	10.04.2020	
4.	Постановка та формалізація задачі	15.04.2020	
5.	Розробка алгоритму призначення робіт та створення програмного продукту	25.04.2020	
6.	Захист програмного продукту	20.05.2020	
7.	Оформлення пояснювальної записки	25.05.2020	
8.	Оформлення дипломної роботи		

Студент _____ Михайло НОВАК
(підпис) (прізвище та ініціали)

Керівник роботи _____ Андрій ОНИСЬКО
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (57 с., 25 рис., 4 додатки).

Дипломна робота присвячена розробці системи (календарного планування навчального процесу) з метою підвищення гнучкості та оперативності планування, та зменшення помилок. Метою цієї роботи було розробити проект на базі CRM системи Salesforce та показати способи автоматизації бізнес процесів. Результуючий продукт дозволяє автоматизувати бізнес процес роботи кафедр університету.

У розділі 1 розглянуто ряд вхідних і вихідних даних, які необхідні для роботи системи. У розділі 2 обґрунтований розроблений евристичний підхід складання кращого розкладу безлічі робіт на безлічі робочих центрів з урахуванням обмежень. У розділі 3 описані основні засоби розробки комплексу завдань і визначені вимоги до технічного забезпечення, обґрунтована архітектура програмного забезпечення.

ABSTRACT

Qualification work includes an explanatory note (57 pp., 25 figs., 4 appendices).

Thesis is devoted to the development of a system (calendar planning of the educational process) in order to increase the flexibility and efficiency of planning, and reduce errors. The purpose of this work was to develop a project based on the CRM system Salesforce and show ways to automate business processes. The resulting product allows you to automate the business process of the departments of the university.

Section 1 discusses a number of input and output data that are required for the system to work. Section 2 substantiates the developed heuristic approach to compiling a better schedule of many jobs at many work centers, taking into account the limitations. Section 3 describes the main tools for developing a set of tasks and defines the requirements for hardware, sound software architecture.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

SOP (Sales and Operations Planning) – перспективне планування продажів і виробництва.

Org – аккаунт організації в Salesforce, який надає можливість користувачам працювати з системою.

Production org – аккаунт організації, де присутній готовий функціонал. Дозволяє налаштування, але розробка не можлива.

Sandbox org – аккаунт організації, призначений для розробки і тестування нового функціоналу.

Scratch org – тимчасовий екзмепляр аккаунта на Salesforce, який використовується лише для розробки. Необхідний для Source-Driven розробки.

Developer org – екзмлпяр аккаунта Salesforce для розробки додатків під платформу.

API (Application Programming Interface) – прикладний програмний інтерфейс.

БД – база даних.

Фреймворк – заготовки, шаблони для програмної платформи, що визначають архітектуру програмної системи; програмне забезпечення, що полегшує розробку і об'єднання різних модулів програмного проекту.

Зміст

ВСТУП

Бізнес процеси з часом ростуть все більше і більше, залучають до себе все більше людей. Разом з таким ростом збільшується кількість помилок, а також архівних даних, а шукати певний запис в горі паперів не раціонально. В нас є вихід – програмне забезпечення. Однак є один мінус, розробка програмного забезпечення для самих простих процесів може займати місяці, а після розробки його необхідно підтримувати (виправляти помилки, які виникають в процесі роботи, оновлювати). Можна купити ліцензію на готове програмне забезпечення, однак якщо для одного бізнес процесу вам необхідно декілька різних програм і чи будуть вони задовольняти всі ваші потреби? Скоріше ви купуєте програмне забезпечення і підстроюєте свою роботу під нього, а має бути навпаки. Враховуючи всі проблеми, зараз пік популярності CRM систем.

Коли люди говорять про CRM, вони зазвичай посилаються на систему CRM, інструмент, який допомагає в управлінні контактами, управлінні продажами, продуктивності тощо.

Використання CRM допомагає вам зосередитись на стосунках вашої організації з окремими людьми - включаючи клієнтів, користувачів послуг, колег чи постачальників - протягом усього циклу взаємодії з ними, включаючи пошук нових клієнтів, надання підтримки та додаткових послуг. Завдяки видимості та легкому доступу до даних простіше співпрацювати та підвищувати продуктивність.

CRM і революція хмарних обчислень змінили все. Мабуть, найбільш вагомою останньою розробкою в CRM-системах стало переміщення в хмару з локального CRM-програмного забезпечення. Організації по всьому світу, звільнені від необхідності встановлення програмного забезпечення на сотні чи тисячі настільних комп'ютерів та мобільних пристроїв, проявляються переваги переміщення даних, програмного забезпечення та послуг у безпечне онлайн-середовище.

Хмарні CRM-системи, такі як Salesforce означають, що кожен користувач постійно має однакову інформацію. Інформація доступна для всіх, хто її потребує.

CRM може бути швидким та простим у застосуванні. Хмарній системі не потрібна спеціальна інсталяція, і немає обладнання для встановлення, що забезпечує низькі витрати на IT та знімає головний біль від контролю над версіями та графіків оновлення.

Взагалі, CRM-системи на основі хмарних технологій оцінюються за кількістю користувачів, які отримують доступ до системи, та типом необхідних функцій. Це може бути дуже економічно вигідним з точки зору витрат капіталу, а також надзвичайно гнучко - дозволяє вам збільшувати масштаби та додавати

більше людей у міру зростання вашої організації. Salesforce є гнучким і з точки зору функціональності - ви не платите за корисні для вас функції. Тому було розроблено програмний продукт який показує всю силу CRM систем і як їх можна застосувати до управління університетом.

1. Постановка задачі

1.1. Нинішній бізнес процес і його недоліки

Пройшло майже четверть 21-го століття. Через декілька десятиліть технології розвинуться настільки, що при перегляді старих фантастичних фільмів, ми не будемо відносити їх до цього жанру. Можливо технології розвиваються швидше, ніж ми освоюємо їх використання. Приклад цьому CRM системи.

Основним їх призначенням є автоматизація бізнес процесу. Автоматизовувати в умовах сьогодення означає прискорення роботи з використанням різних механізмів. Самий популярний з них – комп'ютер або ж обчислювальна машина. На данний момент без цього винаходу не можу обійтись ніякий інший механізм. Всі розробки технологій включають в себе використання комп'ютера. Цей винахід настільки вкорінився в усі сфери життя, що і люди, як живі організми, теж не можуть без нього. Не зважаючи на це, ми ще не вміємо використовувати повний його потенціал.

У багатьох організаціях досить популярне фізичне ведення документації, тобто все пишеться на папері. Необхідно подавати звіти по певному пункту бізнес процесу і таких прикладів можна навести безліч. Давайте розглянемо бізнес процес роботи кафедри університету. Я постараюсь максимально точно описати даний бізнес процес та проаналізувати його.

Бізнес процес являє собою будь-яку діяльність, яка має певний вхідний продукт, покращує його і забезпечує вихідний продукт. Існують різні види таких процесів, проте вони все одно перетинаються. Розглянемо деякі з них і проведемо аналогії до роботи університету.

Процеси управління – це бізнес процеси, які забезпечують функціонування системи. Для цього визначимо структуру системи. У нашому випадку структура виглядає так:

- Фундамент – університет.
- До нього входять факультети і інститути.
- Вони в свою чергу включать в себе кафедри.

Кожен з цих пунктів має свої органи управління. Ними керують і вони кимось керують.

Основний продукт університетів студенти і наукова діяльність. Студентів перетворюють в прекрасних спеціалістів. Наукова діяльність дає свій вклад у розвиток технологій. Сфери, де це все застосовується і для чого нам не цікаві. Звідси випливає другий тип бізнес процесів — основні. Вони описують основну діяльність організації.

Для того, щоб кожен бізнес процес виконував свої завдання їм необхідна підтримка. Тому можна виділити третій вид бізнес процесу — забезпечувальний.

Отже, бізнес процес роботи університету можна поділити на три підпроцеси, які забезпечують функціонування один одного.

Розглянемо основний тип бізнес процесів, кінцевим продуктом якого являються студенти і наукові дослідження. Для забезпечення цих продуктів необхідні кадри — викладачі. Вони прив'язуються до кафедри. Навчають студентів, ведуть дослідження самостійно або ж разом із студентами. Необхідно вести облік всіх цих дій. Тому навчання поділено на сесії для відстежування якості вихідного продукту і ефективності роботи. Протягом сесії проводяться учбові заняття — пари. На них викладачі дають теоретичні знання, закріпляють їх на практичних роботах і відповідно ведеться звітність цього всього — виставляються бали студентам. Рейтингова система надає змогу оцінити якість знань, які отримали студенти.

На жаль в епоху цифрових технологій оцінювання виконується на папері. Очевидний недолік, бо спочатку викладач робить замітки на папері, через певний період часу йому необхідно надати підсумок набраних студентами балів і він сідає або вручну вираховує і записує знову на лист паперу або переносить всі дані з паперу в електронний вигляд і за допомогою технологій подає необхідну звітність. В будь-якому разі цей алгоритм все одно повторюється, на будь яких етапах бізнес процесу і при його виконанні збільшується відсоток людських помилок. Тобто такий підхід впливає не тільки на ефективність роботи, але й на її якість.

Студентам також така ситуація додає незручностей. Щоб дізнатись певну оцінку своєї роботи потрібно постійно підходити до викладача і просити подивитись в його замітки. Так він може втратити всю свою перерву, бо якщо відразу підійде 100 людей, яким не терпиться дізнатись оцінку. Або ж якщо викладач втратить свої нотатки, як тоді бути студентам? Перездавати все заново?

Тобто обом сторонам необхідний інструмент, який надає змогу в будь-який момент часу дізнатись всю необхідну інформацію і не втратити її. На допомогу приходяться хмарні технології. Якщо перенести всю роботу в хмару, це надасть змогу бачити в будь-який момент часу всі необхідні дані.

Розглянемо забезпечувальний тип бізнес процесу, який можливо теж потребує оптимізації. Саме він забезпечує підтримку основного процесу. До нього можна віднести бухгалтерський облік, кадрова і інформаційне забезпечення.

Бухгалтерський облік дуже відповідальна робота. Найменша помилка може спричинити великі неприємності, як наприклад не правильно вирахуваний бюджет через помилку, спричинену людським фактором. В результаті співробітникам можливо затримають зарплату, цікаві проекти в середині університету не отримають фінансової підтримки. У разі фатальних помилок

можливе скорочення як зарплат, так і співробітників, закриття цілих факультетів. Звичайно вони трапляються рідко, проте все ж трапляються. В історії відомі випадки, коли компанії втрачали мільярди доларів через помилку в заповненні таблиці Excel. Для уникнення таких ситуацій необхідно використовувати програмне забезпечення, яке б перевіряло введені дані і обробляло їх.

Для кадрового і інформаційного забезпечення важливу роль грає комунікація. Співробітники постійно взаємодіють між собою. Іноді це досягає величезного масштабу. Коли одній людині необхідно повідомити важливу інформацію великій кількості людей. Що в таких випадках робити і де шукати інформацію про всіх цих людей? Хоча знайти інформацію зараз не є проблемою, але проблемою є швидкість її пошуку. Тому ідеальний варіант буде використання певної хмарної технології для доступу до всіх необхідних даних де завгодно і коли завгодно.

Враховуючи всі ці фактори, необхідний комплексний інструмент, який може задовільнити відразу всі потреби, для оптимізації бізнес процесу. Таким інструментом може стати CRM система.

1.2. Опис оптимізованого бізнес процесу та результат його автоматизації

CRM система буде в основному зосереджуватись на кафедрі. Вона зможе надати змогу полегшити комунікацію між адміністрацією, викладачами і студентами, автоматизувати рутинні задачі і зменшити кількість помилок. Всі ці люди будуть поєднані єдиною платформою, якою будуть користуватись кожен день.

Я вважаю, що задовільнити всі перечисленні критерії здатна технологія Salesforce. Дану платформу розробили компанія з таким же іменем. Дана розробка представляє з себе платформу, з вбудованими методами налаштування і розробки, яку можна настроїти під будь-які потреби, швидко і надійно. Хмарна технологія забезпечить вам доступ до будь-яких даних і з іншої сторони гарантує їх конфіденційність завдяки своїй моделі безпеки. На цій платформі можна налаштувати параметри доступу до записів. Тобто навіть в середині вашої організації дані все одно залишають свою приватність.

Оптимізований бізнес процес буде мати такий вигляд:

- Викладачі будуть вести всі свої записи в середині платформи. Це можуть бути бали за контрольні роботи студентів, практичні роботи, пропуски студентів. Окрім цього викладачі будуть мати змогу бачити актуальну персональну і учбову інформацію своїх студентів. Так як Salesforce хмарна технологія, дані занесені в неї відразу будуть доступні всім іншим хто має право доступу до них. Це дасть змогу студентам моніторити в реальному

часі їх прогрес у навчанні. Окрім цього в систему викладачі можуть мати змогу завантажувати необхідні для студентів дані, такі як програмні продукти, методичні вказівки, лабораторні роботи і т.д. Все буде структуровано і швидко доступне.

- Викладачі будуть значно менше витрачати часу на створення звітів своєї роботи. Так як, вони зберігають всі свої дані в системі, можна автоматично генерувати необхідні звіти по шаблону.
- Будуть створені правила валідації важливої документації, такі як бухгалтерський облік.
- Платформу можна синхронізувати з будь-яким іншим програмним забезпеченням.

2. Існуючі методи рішення

2.1 Що таке Salesforce?

Можливо, ви вважаєте, що Salesforce – це лише CRM (Client Relationship Managment). Вона зберігає ваші дані, надає вам процеси взаємодії з клієнтами та забезпечує спосіб співпраці з колегами. Є набагато більше, ніж це. Salesforce – це ціла платформа. Вона постачається з великою кількістю стандартних функціональних можливостей та нестандартних продуктів та функцій, які ви можете використовувати для швидшого створення необхідного вам функціоналу.

Платформа Salesforce - це корпоративна платформа номер один у світі, створена для того, щоб допомогти вам ефективніше працювати. За допомогою набору технологій ви можете налаштувати та розширити потужність CRM , щоб вона краще відповідала вашим потребам.

Salesforce надає повний спектр інструментів, які полегшують побудову майже всього. Наприклад, адміністратори можуть створити простий додаток для роботи з програмою Lightning App Builder, перетягнувши стандартні компоненти. Також розробники можуть створювати власні компоненти, якими адміністратор може користуватися під час створення програм.

Salesforce хмарна система і це означає, що обчислювальні ресурси розподіляються між багатьма користувачами. Незалежно від розміру вашої організації, ви отримуєте доступ до тієї ж обчислювальної потужності, зберігання даних та основних функцій.

Розподіл обчислювальних ресурсів реалізовується за певним принципом:

1. Для вашої організації виділяється певний простір, якому виділяються певні ресурси.
2. Ви налаштовуєте ваш простір під необхідний вам бізнес процес. Їх може бути скільки забажаєте, проте зазвичай вони побудовані так, щоб допомагати один одному.
3. Визначаєте параметри авторизації користувачів і надаєте їм доступ до створеного функціоналу.

Незважаючи на те, що ви ділитесь простором з іншими організаціями, ви можете бути впевнені в безпеці ваших данних. Також ви отримуєте найновіші та найкращі функції за допомогою автоматичних, безшовних оновлень тричі на рік. Оскільки Salesforce - це хмарний сервіс, вам ніколи не доведеться встановлювати нові функції або турбуватися про своє обладнання.

Інформація про ваш простір зберігається у вигляді метаданих. Метадані — це інформація про інформацію, стандартні та спеціальні конфігурації, функціональність та код у вашому просторі.

Починаючи з 1999 року, Salesforce був зосереджений на побудові технологій для бізнесу, що постачаються через Інтернет, витісняючи традиційне програмне забезпечення для підприємств.

2.2 Архітектура платформи

2.2.1 Термінологія

Instance - це повний екземпляр системи, мережевої та пам'яткової інфраструктури, як спільних, так і нерозподілених. Наприклад, na14.salesforce.com - це екземпляр, де na14 - це унікальне ім'я конкретного екземпляру.

Superpod - сукупність зовнішніх систем, мережевої та пам'яткової інфраструктури, включаючи проксі-сервери, балансири завантаження, поштові сервери та інша інфраструктура, що підтримує кілька екземплярів. Superpod забезпечує службову ізоляцію в центрі обробки даних, щоб проблеми із спільними або складними компонентами не могли впливати на кожен примірник цього центру обробки даних.

Org (організація) - єдиний клієнт програми Salesforce. Початок використання послуг на веб-сайті www.salesforce.com або developer.force.com, породжує нову організацію. Org налаштовується і може мати різні параметри безпеки, налаштування видимості запису та обміну, зовнішній вигляд інтерфейсу користувача, робочі процеси, тригери, спеціальні об'єкти, спеціальні поля на стандартних CRM-об'єктах salesforce.com і навіть власні API REST. Орган може підтримувати від будь-якого до мільйонів ліцензованих індивідуальних облікових записів користувачів, облікових записів користувачів порталу та облікових записів користувачів Force.com Sites.

Sandbox - екземпляр сервісу salesforce.com, який є повною копією основної Org. Це тестові середовища для клієнтів, щоб впровадити новий функціонал та протестувати коректність його роботи перед завантаженням на основний Org.

Використані програмні технології

- Linux для розробок та систем первинного виробництва
- Solaris 10 w / ZFS
- Jetty
- Solr
- Memcache
- QPID Apache
- QFS
- Puppet, Razor

- Perl, Python
- Nagios
- Perforce, Git, Subversion

2.2.2 Вхід у службу Salesforce.com

Ми підтримуємо пул серверів для обробки трафіку для входу для всіх екземплярів. Кілька серверів з багатьох (але не всіх) екземплярів приймають запити на вхід та перенаправляють сеанс на домашній екземпляр користувача. Це те, що відбувається під час входу через login.salesforce.com.

Клієнтський трафік починається із зовнішньої DNS. Після того, як пошук успішно повернув IP-адресу для екземпляра, стандартна маршрутизація в Інтернеті спрямовує його до відповідного центру обробки даних.

Як тільки трафік потрапляє в нашу мережу в цьому центрі обробки даних, він спрямовується на пару балансира навантаження, на якій живе цей IP. Всі наші IP-адреси з інтернетом - це VIP-персони, налаштовані на активну / резервну пару балансирів навантаження.

2.2.3 Всередині org

Балансир навантаження направляє трафік до рівня додатку даного екземпляра. На цьому рівні ми обслуговуємо як стандартний трафік веб-сторінок, так і наш трафік API. Трафік API складає понад 60% трафіку, який обслуговується загальним рівнем нашого додатка. Залежно від потреб запиту клієнта, він буде спрямований на додаткові рівні серверів для різних типів резервної обробки.

Основний рівень додатків містить від десяти до 40 серверів додатків, залежно від примірника. Кожен сервер запускає один JVM Hotspot, сконфігурований із різними потребами доступних ресурсів, залежно від конфігурації серверного обладнання.

Пакетний сервер відповідає за виконання запланованих автоматизованих процесів на рівні баз даних. Salesforce.com пропонує ряд послуг, включаючи базове та розширене управління вмістом. Існує сервер пошуку вмісту та пакетний сервер для управління асинхронними процесами на рівні додатка екземпляра.

2.2.4. База даних

Первинний потік даних відбувається між базовим рівнем сервера додатків і рівнем бази даних. З точки зору програмного забезпечення, все проходить через базу даних, тому продуктивність бази даних є критичною. Кожен первинний

екземпляр використовує 8-вузловий кластерний рівень бази даних. Клієнтський Sandbox має 4 вузлові кластеризовані рівні бази даних.

Оскільки Salesforce є системою, керованою базою даних, зменшення навантаження на базу даних є критично важливим. Для зменшення навантаження на рівень бази даних був розроблений ACS - API Cursor Server. Це вирішило дві проблеми, які дозволили нам значно покращити роботу основної бази даних. По-перше, це дало використання можливість зберігання курсорів у базі даних, але операції видалення впливали на продуктивність. По-друге, після переходу до використання таблиць бази даних для утримування курсорів накладні витрати DDL стали негативними. Так народився OCH. ACS - це кеш-пам'ять курсора, що працює на парі серверів, що забезпечує метод для завантаження обробки курсору з ярусу бази даних.

2.2.5. Пошук

Пошуковий рівень працює на товарних хостах Linux. Ці хости отримують свої дані із спільного масиву SAN через файлову систему NFS. Пошукові індекси зберігаються на флеш-накопичувачі, щоб забезпечити більшу продуктивність для пропускної здатності пошуку.

Індексація пошуку в даний час відбувається на серверах трансляції, які монтують LUN з масивів зберігання через Fiber Channel SAN. Ці LUN складають файлову систему QFS, яка дозволяє одноразовому доступу запису даних, але мультимедійному доступу до читання цих даних. Трансляція відбувається, коли ці ж LUN встановлюються лише для читання з групи чотирьох серверів NFS, що працюють під Solaris 10 на SPARC. Ці файлові системи, встановлені SAN, потім передаються через NFS до попереднього описаного рівня пошуку.

2.2.6. Fileforce

Salesforce підтримує рівень серверів, які забезпечують зберігання об'єктів, схожих за концепцією проекту S3 або OpenStacks 'Swift Amazon. Ця система, Fileforce, була розроблена внутрішньо для зменшення навантаження на наш рівень бази даних. До впровадження Fileforce всі бінарні великі об'єкти (BLOB) зберігалися безпосередньо в базі даних. Як тільки Fileforce з’явився в Інтернеті, до нього перенесли всі BLOB, розміром більше 32 Кб. BLOB розміром менше 32 КБ продовжують жити в базі даних. Усі BLOB в Fileforce мають посилання на базу даних, тому для відновлення даних Fileforce з резервних копій нам потрібно запустити екземпляр бази даних на основі резервного копіювання бази даних з тієї ж точки відновлення.

Fileforce включає функцію bundler, розроблену для зменшення завантаження диска на сервери Fileforce. Якщо в базі даних зберігаються 100+ об'єктів розміром менше 32 КБ, на серверах додатків запускається процес для

об'єднання цих об'єктів в один файл. Посилання на пакетний файл залишається в базі даних разом із зміщенням пошуку в комплекті. Це схоже на систему зберігання зображень Haystack у Facebook, але вбудована в систему зберігання об'єктів.

2.3 Порівняння Salesforce з мовою програмування Java.

Компанія Salesforce створила свою вузько-спеціалізовану мову програмування – Apex. Вона була створена на основі мови програмування Java. Як інженер або розробник програмного забезпечення, вам не складе труднощів вибрати платформу розробки або мову, на якій слід зосередити увагу, між Java та Salesforce. Давайте ознайомимося з обома прикладами, а потім підемо глибше. Платформа Java призначена для людей, які люблять кодування та мають певну впевненість у створенні власного програмного забезпечення. З іншого боку, Salesforce вже розроблений в CRM, де розробнику потрібно його вдосконалити, а не будувати. Ця ж причина підтверджує, що Salesforce краще стосується консультацій, але не стосується розробників. В даний час Salesforce користується більшим попитом там, де грамотних розробників менше. Це дає змогу організаціям обрати офшорні IT-сервіси, оскільки їм потрібно мати функцію Salesforce, яку не можна замінити на меншу. Коли розробник перекваліфіковується з Java на Salesforce, він ніколи не озирається назад, оскільки вони мають різні утиліти та основні функції.

Відповідальність розробника Java полягає в тому, щоб залишатися протягом усього циклу розробки та аналізувати або визначати, чи є якісь проблеми для забезпечення ефективних рішень. Крім цього, від них також очікується документування вимог та аналіз даних для кращого забезпечення якості та тестування. Компанія з розробки Java завжди шукає розробника для вирішення вищезазначених питань або завдань кваліфіковано:

- розробити, реалізувати та підтримувати фази додатку Java;
- легко брати участь у архітектурній діяльності та розробці програмного забезпечення;
- провести велику кількість програмного аналізу, тестування програм та налагодження;
- ефективно розробляти коди програм для програм Java;
- розробити технічні проекти, що підтримують розробку додатків.

Щоб подолати перешкоди технології, розробнику важливо вміти вирішувати проблеми, як розробник Java.

З просуванням в IT-секторі велика кількість можливостей відкрила свої двері для розробника Java, оскільки він має таку ж корисність як в приватному, так і в державному секторі. Графік розвитку кар'єри Java-розробника, безумовно,

вищий і визнаний відомою компанією з розробки Java. Якщо ви не знаєте їх значущості, перегляньте наведені нижче пункти:

- Це найпоширеніша мова програмування - близько 9 мільйонів розробників з більш ніж 7 мільярдами пристроїв у всьому світі використовують цю мову програмування. Звідси немає заперечень у його ефективності та технологічній корисності. Це також збільшило перспективи кар'єри у кваліфікованих фахівців на Java, доступних у багатьох секторах.
- Java не просто кодування - для розробника Java робота або завдання сильно відрізняються, оскільки вони мають набагато більше, ніж просто кодування. Вони також повинні доглядати за розробкою інтерфейсу, створенням або тестуванням динаміки додатків. Під час розробки програми та впродовж усього процесу розробники Java активно беруть участь у кожній фазі.
- Ця платформа є спільною - одна з головних переваг, пов'язаних із співпрацею до моменту подання заявки, полягає в тому, що розробник може взаємодіяти з низкою професіоналів з різних секторів. В основному це стосується веб-дизайнерів, веб-розробників, інженерів програмного забезпечення тощо, це допоможе їм розширити свої комунікаційні та передавальні навички.
- Його корисність є в реальних програмах - Java не обмежується технологічним сектором, а однаково важлива у фінансовій формі для послуг охорони здоров'я. Велика кількість бізнес-платформ використовує його для своїх процесів розвитку. Більшість інших платформ та мов розвитку вимагають від людини технологічної ефективності, особливо з програмуванням. Будучи розробником Java, вам не доведеться доводити ступінь, а досить складені відмінні навички, і ви все зробили.

Salesforce - це продукт із чітко визначеними циклами випуску. Кожен розробник повинен бути в курсі нових можливостей про реліз Salesforce. Salesforce поставляється з пакетом, наповненим корисними побічними продуктами, такими як Desk.com, Work.com, Data.com, Communities тощо. Знання про них дуже потрібні. Слід також знати про внутрішню модель даних, до якої можна отримати пряме посилання на об'єкт, як у поєднанні Java-Spring-Hibernate.

При створені програмного забезпечення на платформі Salesforce, воно функції автоматично асоціюються з:

- підтримкою API;
- Мобільний доступ до програмного продукту за допомогою Salesforce;
- Інтерфейс користувача для функцій перегляду, створення, оновлення або видалення записів.

При розробці на платформі Salesforce розробники зосереджуються на задачі - при створенні додатків не потрібно багато хвилюватися з питань

апаратних чи програмних засобів. Навіть рішення щодо вибору із найскладнішими або нерівними операційними системами або декількома серверами додатків також може бути полегшене.

В даний час платформа Salesforce зростає в експоненціальному масштабі зі зростаючою швидкістю, отже, є більш високі шанси, що незабаром вона стане лідером на ринку декількох спеціальних служб веб-розробки.

Програма Java запускається на JVM і може розміщуватися на серверах або працювати локально, але Salesforce - це чиста хмарна технологія. Для розробників Java потрібно розуміти, що це багатостороння архітектура. Дизайн хмарного сервісу, який має багато орендарів, може мати драматичний вплив на доставку додатків та продуктивність IT-організації, але більшість керівників технічних фірм, CTO, системних архітекторів та розробників, які використовують хмари, не замислюються над цим, тому що це все магія що прозоро відбувається за лаштунками.

Salesforce як CRM потребує великої інтеграції з іншими сторонніми інструментами. Це один із найпоширеніших сценаріїв, коли розробник Java дізнається про Salesforce та піде на шлях просвітлення. Salesforce також надає готові API для інтеграції. Java-розробники мають декілька варіантів інтеграції Java-систем із Force.com та Database.com.

- API RESTful, такі як API FOREST REST, Bulk API та Streaming API.
- Роз'єм для веб-служб (WSC)

Мова програмування Salesforce Apex та Java досить схожі. Код Apex використовує синтаксис, схожий на Java, і, як і Java, Apex Code строготипізований, що полягає в тому, що код збирається розробником до його виконання, і що змінні повинні бути пов'язані з певними типами об'єктів під час цього процесу компіляції. Структури управління також схожі на Java, петлі for / while і ітератори точно такі ж. Коде Apex значно обмежений за обсягом, тоді як мова та платформа на зразок Java можуть використовуватися для створення майже будь-якого типу додатків. Apex призначений виключно для створення бізнес-додатків для управління даними та процесами в рамках платформи Force.com. Бібліотека функцій і системних команд Salesforce необхідна розробнику Java, щоб якомога швидше навчитися та зрозуміти. Salesforce та Java також мають деякі загальні моделі дизайну.

Розроблені Java класи потребують упаковки перед розгортанням, і кожен розробник повинен виконати власне тестування блоку та виправлення помилок. Створення Jar - це стандартний процес, який виконується вручну або IDE, в той чай як Force.com дає деякі обмеження, такі як обмеження використання ресурсів платформи та процес розробки коду Apex, тестування, виправлення помилок та розгортання. Salesforce має дуже чіткі кроки перевірки та розгортання для Org. Розробка особливостей у Salesforce CRM завжди характерна для певної версії системи. Усі компоненти розробки, такі як класи Apex, сторінки Visualforce,

правила перевірки тощо, можуть бути додані до наборів змін, керованої та некерованої упаковки та метаданих. Усі ці методи мають різні цілі. Керувати пакетом призначено для розробки додатків у Appexchange, Changeset, а метадані групи - для розгортання коду.

2.4 Порівняння Salesforce із CRM системою Zoho

Salesforce очолює всі категорії CRM своїм комплексним рішенням, заснованим на хмарних технологіях. І незважаючи на те, що Salesforce пропонує базові пакети послуг, які включають такі функції, як акаунти та управління контактами, відстеження завдань і подій, синхронізація Outlook, мобільний додаток Salesforce1, бібліотека вмісту, настроювані звіти та соціальна мережа компанії. Може бути випадок, коли, наприклад, організація не знає, чи дійсно CRM їм піде на користь і чи потрібен їй весь цей функціонал. Або, можливо, у них є підстави сумніватися, що це буде здійснено повною мірою та надається повна підтримка зверху вниз. Можливо, у них є необхідність створити історію роботи з CRM, щоб представити доповіді вищому керівництву звіт про тематичний випадок, сподіваючись запропонувати більш рішучі вкладення часу та ресурсів.

У будь-якому з цих випадків хороша ідея може здатися CRM, що пропонує безкоштовну версію. Влітку 2015 року Business News Daily переглянув вичерпний список програмного забезпечення CRM. Після цього вони опублікували посібник для покупців на основі своїх висновків. Кожен запис у посібнику покупця давав основний конспект функцій та переваг для кожного програмного забезпечення CRM, що є чудовим ресурсом для всіх на ринку. Але справжнім центром посібника покупця була колекція вичерпних оглядів CRM, які оцінили найкращі в одній з трьох категорій. Ці категорії були найкращим CRM-програмним забезпеченням для малого бізнесу, найкращим CRM-програмним забезпеченням для дуже малого бізнесу та найкращим безкоштовним програмним забезпеченням CRM. Zoho виграв категорію найкращого безкоштовного CRM-програмного забезпечення.

Безкоштовна версія Zoho CRM має всі основні функції, які потребує малий бізнес, включаючи збір клієнтів, управління контактами, автоматизацію робочого процесу, аналітику, соціальну співпрацю. Це також веб-платформа, до якої можна отримати доступ через мобільний пристрій. Безкоштовна версія Zoho дозволяє до 10 користувачів, що є приємною функцією для малого бізнесу. Іншими функціями Zoho були інтеграція сторонніх додатків, безпека та 360-градусний огляд існуючого бізнес процесу, який відображає всю критичну інформацію, від контактів до аналітики, що бізнесу потрібно для прийняття рішень.

Те, що Zoho дозволяє отримати до 10 користувачів у безкоштовній CRM - це користь для малих організацій, але лише для них. Крім того, це лише перевага для тих, хто не прагне розвиватися. Це означає, що безкоштовна версія Zoho займає досить вузький ринковий сегмент. Він також менш придатний до налаштування під певні задачі, ніж преміальний CRM, як Salesforce. Наприклад, якщо організація хоче додати спеціальні звіти або додаткові модулі, що є дуже реальною ймовірністю для підприємств будь-якого розміру, їм потрібно купувати платну підписку. Функції налаштування Zoho також отримали критику. Хоча навігація у верхній частині панелі інструментів має вкладку для кожної функції, деякі користувачі скаржаться, що для виконання простих завдань та навігації по кожному розділу потрібно занадто багато кроків.

До цього моменту в нашому огляді Zoho ми помітили, що малі організації з обмеженими бюджетами, які прагнуть впорядкувати управління та контроль відносин з клієнтами, свого бізнес процесу та працівників, може набути значної цінності, інтегруючи безкоштовну CRM Zoho в свою організацію.

Той самий посібник покупця CRM, який увінчав Zoho найкращим безкоштовним CRM, також вінчав Salesforce, що найкращий загальний CRM для малого бізнесу, і ось чому:

1. Промисловий стандарт CRM: Salesforce просто має репутацію за те, що забезпечує найкращу платформу. Так, це може коштувати дорожче. Однак Salesforce відомий тим, що встановлює стандарт, коли справа стосується простоти використання, налаштування та інновацій у просторі CRM.
2. Налаштування бізнес-потреб: Salesforce пропонує доступ до AppExchange, що дозволяє користувачам легко налаштувати свої CRM з тисячами додатків, вбудованих на платформу Salesforce. Фактично, у 91% компаній, які використовують Salesforce, встановлено принаймні один додаток AppExchange.
3. Масштабність: Мабуть, найголовніше з усіх, Salesforce не обмежує своєю функціональністю бізнес процес, яким би складним він в майбутньому не став.

2.5 Порівння Salesforce із Microsoft Dynamics

Хоча їх основна функціональність схожа, обидва продукти набирають додаткових балів у певних областях. Давайте розглянемо переваги та недоліки кожного рішення.

Плюси Salesforce:

- Salesforce керує своєю орієнтацією на споживача, і добре відома своїм орієнтованим на клієнта ставленням. Trailhead, навчальна платформа для

користувачів Salesforce, часто вважається одним із найважливіших активів компанії.

- Компанія AppExchange є одним з найбільших B2B-магазинів у цій галузі, де представлені тисячі сторонніх інтеграцій, щоб допомогти бізнесу отримати більше можливостей від платформи.
- Служби маркетингової хмарності вже зрілі та готові до використання. Dynamics 365 немає таких служб взагалі.
- Salesforce має надійні засоби електронної комерції, які тільки покращаться тепер, коли Salesforce придбав платформу електронної комерції Demandware.

Плюси Microsoft Dynamics:

- Dynamics 365 має конкурентоспроможні ціни та гнучкість у своїй ліцензійній моделі, що робить її більш економічною та доступною для менших підприємств.
- Dynamics 365 має вбудовану інтеграцію з LinkedIn. Dynamics 365 для продажів та LinkedIn Sales Navigator об'єднуються, щоб створити Microsoft Relationship Sales, який використовує дані LinkedIn для визначення потенційних клієнтів та сприяння створенню стосунків за допомогою персоналізованого залучення.
- Dynamics 365 також виграє від безперебійної інтеграції з іншими продуктами Microsoft, такими як Office 365, Outlook та потужним інструментом бізнес-аналітики Microsoft Power BI. Синхронізація між платформами не тільки сприяє продуктивності, але і знайомство з інтерфейсом Microsoft може забезпечити більш позитивну роботу користувачів.

В плані гнучкості налаштування і розробки кастомних рішень Microsoft Dynamics терпить повний провал в порівнянні з Salesforce, який в свою чергу надає безліч інструментів для розробки та налаштувань, всі ці можливості Salesforce будуть розглянуті пізніше. Microsoft Dynamics має зовсім іншу концептуальну модель. В ній немає можливості створювати функціонал за допомогою написання кода, що є величезним недоліком. Створення персоналізованого функціоналу реалізовано декларативними методами розробки, що надзвичайно обмежує гнучкість системи. В свою чергу, Salesforce це ціла екосистема, більш схожа до операційних систем.

3. Методи розробки вибраного типу розробки програмного забезпечення та їх аналіз

3.1 Життєвий цикл продукту та моделі розробки

Salesforce пропонує різні інструменти та процеси розробки для задоволення потреб клієнтів. Розглянемо процес управління життєвим циклом додатків (ALM) та три моделі розвитку.

- Change Set розробка
- Org розробка
- Package розробка

На високому рівні всі три моделі розвитку дотримуються одного і того ж процесу ALM. Але моделі відрізняються тим, що вони дозволяють вам керувати змінами у вашій Org. Контроль змін - це велика справа в розробці програмного забезпечення, і ви можете вибрати модель розробки, яка найкраще відповідає вашій ситуації, якщо ви розумієте свої потреби і переваги кожної з моделей розробки.

Для безпечної зміни даних у вашій Production Org вам необхідно використовувати Sandbox Org - тестові організації, які є точними копіями вашої основної Production Org. На них можна сміливо розробляти нові види функціональності у виробничих органах. Налаштування, які не впливають на дані, безпечно створювати у вашій Production Org, таких як розробка нових інформаційних панелей, звітів та шаблонів електронної пошти. Однак певні налаштування, зроблені безпосередньо на Production Org, можуть створити безлад, видаливши дані або ще гірше.

Якщо ви не протестуєте зміни, перш ніж зробити їх доступними у Production Org, то можливі такі наслідки:

1. Методи обробки внесених даних випадково створюють нескінченний цикл обробки.
2. Зміна типу поля змінює дані способами, які ви не можете скасувати.
3. Логічна помилка в правилі перевірки коректності даних перешкоджає збереженню запису.
4. Зміни в макеті сторінки збивають з пантелику людей замість покращення їх досвіду.

Найбезпечніший спосіб налаштувати ваш Org - це вносити та перевіряти зміни, використовуючи спеціальне середовище для розвитку. Насправді деякі зміни повинні бути здійснені в тестовому середовищі, тому існують певні обмеження розробки, наприклад розробники не можуть писати код Apex в Production Org.

Для перенесення даних з тестового середовища в основне використовуються Change Set, це декларативні інструменти переносу функціональності між організаціями. Їм не потрібно використовувати інтерфейс командного рядка або систему контролю версій для задоволення зростаючих потреб у налаштуванні та переносу функціональності.

Отже, процес управління життєвим циклом додатків (ALM) має такий вигляд:

1. План випуску. Планування проекту починається із налаштування чи розробки з плану. Зберіть вимоги та проаналізуйте їх. Запропонуйте менеджеру продуктів (або його аналогу) створити технічні характеристики та поділитися ними з командою розробників для впровадження. Визначте різні середовища розробки та тестування, які потрібні команді в міру прогресу проекту через цикл ALM.
2. Розробка. Завершіть роботу, дотримуючись технічних характеристик. Виконуйте роботу в середовищі, що містить копію метаданих Production Org, але без даних, що містяться в ній. Розробіть на платформі Lightning, використовуючи відповідну комбінацію засобів декларування (декларативні засоби розробки процесів, майстер користувальницьких об'єктів та інші в інтерфейсі) та програмні засоби (консоль розробника, редактор вихідного коду або код Visual Studio).
3. Тестування. Виконайте зміни, які ви вносите, щоб перевірити, чи працюють вони за призначенням, перш ніж інтегрувати їх у роботу інших людей. Виконайте тестування в тому ж типі середовища, що використовували на етапі розробки. На цьому етапі зосередьтеся на тестуванні самих змін, а не на розумінні того, як ваші зміни впливають на інші частини випуску чи додаток у цілому.
4. Побудова релізу(випуску). Об'єднайте всю функціональність, яку ви створили або змінили на етапі розробки, в екземпляр єдиного релізу: логічний набір налаштувань, який ви розгортаєте для Production Org.
5. Тестування релізу. Перевіряйте, що виправляєте, що потрібно розробити, але перевірити безпечно в тестовому середовищі, що максимально копіює Production Org. Використовуйте реальну кількість репрезентативних даних про виробництво. Підключіть тестові дії до всіх необхідних зовнішніх систем, щоб протестувати точки інтеграції вашої виробничої системи.
6. Реліз. Завершивши тестування та дотримавшись показників якості, ви можете розгорнути нову функціональність на Production Org. Ознайомте з релізом своїх співробітників та партнерів, щоб вони зрозуміли зміни. Якщо випуск має значний вплив на користувачів, створіть окреме середовище з реалістичними даними для навчання користувачів.

3.2. Розробка з використанням системи контролю версій

Коли ваша організація зростає, ви можете зіткнутися з проблемами щодо координації та тестування змін у різних проектах, використовуючи Change Set. Розробка пакету функціональності безпосередньо вирішує ці проблеми. Ваші нововведення впорядковані в пакети на основі групи функцій або налаштувань, які ви хочете доставити разом. Проект Salesforce DX відображає цей пакетний підхід до організації вашого джерела.

Проект Salesforce DX (далі Salesforce DX) - це локальна структура каталогів ваших метаданих у вихідному форматі. Це дозволяє розробляти і тестувати інструменти Salesforce DX. Він містить файли конфігурації для створення scratch orgs(тимчасові екзмепляри Org).

Коли ви використовуєте CLI (command line interface) для створення нового проекту Salesforce DX, він створює структуру каталогів проекту для вас. Створюючи проект з нуля, для вас створюється багато речей. Платформа створює базовий файл конфігураційного проекту, зразки файлів та каталогів для ваших тестів та зразки наборів даних. Ми також створюємо каталог «пакет» для джерела вашого пакету.

Пакет - це група пов'язаних кодів та налаштувань. Розробники можуть протестувати пакет незалежно від інших компонентів у вашій Org. Компоненти метаданих всередині пакета можуть жити лише в одному пакеті одночасно.

Як мінімум, проект управляє джерелом для одного пакету. Однак, якщо кілька пакунків збираються та випускаються разом, ви можете організувати ці пакети в один проект DX. Кожен із ваших пакетів вирівнюється до каталогу пакетів, визначеного у файлі конфігурації проекту.

Scratch org створені з вашого джерела та метаданих, полегшують послідовно створювати додаток. Розробники працюють лише з джерелом та метаданими для конкретного проекту. Не потрібно копіювати речі, які вам не потрібні. А тому що scratch org - це тимчасове середовище Salesforce, ви можете швидко створити нову для кожного пакету чи проекту розвитку. Після створення scratch org, залишаються деякі завдання налаштування.

Перш ніж інтегрувати систему управління версіями, обов'язково проведіть тести. Розробники можуть або використовувати один і той самий scratch org, щоб запустити тести, або запустити інший спеціально для тестування, перш ніж внести зміни в систему контролю версій. Ця ж схема розкручування скретч-тестів для тестування є ключовою частиною автоматизованої системи безперервної інтеграції.

Ключовою особливістю Salesforce DX є те, що ви можете легко тримати проект в цілому і різні частини функціональності, що розробляються різними людьми асинхронно в синхронізації. Salesforce DX відстежує будь-які зміни, внесені на місцевому рівні в проект, та будь-які зміни, внесені вами в scratch org.

Файли вашого проекту може містити величезні файли. Формат Salesforce DX розбиває великі файли вихідних файлів, щоб зробити їх більш засвоюваними та простішими в управлінні системою контролю версій. Наприклад, Salesforce DX перетворює власні об'єкти та власні переклади об'єктів у декілька файлів та каталогів. Ця структура даних значно полегшує пошук того, що ви хочете змінити або оновити. Менші файли в керуванні даними означають меншу кількість конфліктів злиття під час розробки командою.

3.3 Мова програмування Apex

Apex - це строго-типізована, об'єктно-орієнтована мова програмування, яка дозволяє розробникам виконувати операції управління потоком та транзакціями на серверах Salesforce спільно з викликами до API. Використовуючи синтаксис, схожий на Java та діє як процедури, що зберігаються в базі даних, Apex дозволяє розробникам додавати бізнес-логіку до більшості системних подій, включаючи натискання кнопок, відповідні оновлення записів. Код Apex може бути ініційований запитами веб-служб та триггерами на об'єктах.

Apex забезпечує вбудовану підтримку для загальних ідіом платформи Lightning Platform, включаючи:

- Виклики мови маніпулювання даними (DML), такі як INSERT, UPDATE та DELETE, що включають вбудовану обробку DmlException
- Вбудована мова запитів об’єктів Salesforce (SOQL) та мова пошуку об'єктів Salesforce (SOSL), що повертають списки записів sObject
- Цикл, що дозволяє проводити обробку кількох записів одночасно
- Синтаксис блокування, який запобігає конфліктам оновлення записів
- Користувацькі загальнодоступні виклики API, які можна побудувати із збережених методів Apex
- Попередження та помилки, видані, коли користувач намагається редагувати або видаляти користувацький об’єкт або поле, на яке посилається Apex

Apex заснований на знайомих ідіомах Java, таких як синтаксис змінних та виразів, синтаксис блоку та умовного висловлювання, синтаксис циклу, позначення об'єкта та масиву. Там, де Apex вводить нові елементи, він використовує синтаксис та семантику, які легко зрозуміти. Тому на Apex код одночасно стислий і простий для запису.

В Apex можна багаторазово діставати дані з бази даних та багаторазово маніпулювати даними на протязі однієї транзакції, яка виконується на сервері Salesforce. Розробники використовують процедури, що зберігаються в базі

даних, щоб виконувати необхідну логіку. Як і інші процедури, що зберігаються в базі даних, Apex не намагається надати загальну підтримку для роботи в інтерфейсі користувача. Його основна ціль back-логіка.

Apex - це строго-типізована мова, яка використовує прямі посилання на об'єкти схеми, такі як імена об'єктів і полів. Він швидко виходить з ладу під час компіляції, якщо будь-які посилання недійсні. Він зберігає всі власні залежності поля, об’єктів і класів у метаданих, щоб гарантувати їх видалення, викликаючи активний код Apex.

Apex інтерпретується, виконується та повністю контролюється платформою і працює у багатосторонньому середовищі. Отже, двигун виконання Apex розроблений для того, щоб тісно захищати від монополізації спільних ресурсів. Будь-який код, який порушує обмеження, не компілюється.

Apex забезпечує вбудовану підтримку для створення та виконання тестових підрозділів. Він включає результати тестів, які вказують, наскільки охоплений код та які частини вашого коду можуть бути ефективнішими. Salesforce гарантує, що всі користувацькі коди Apex працюють так, як очікувалося, виконуючи всі тести блоку перед будь-яким оновленням платформи.

Ви можете зберегти свій код Apex на різних версіях API. Це дає змогу підтримувати коректність роботи коду незважаючи на постійні оновлення платформи.

Код Apex, як правило, містить багато речей, які вам можуть бути знайомі з інших мов програмування (рис 3.1).

Рисунок 3.1. Приклад коду Apex

У користувальницькому інтерфейсі Salesforce ви можете вказати версію API Salesforce, на якій можна зберегти клас Apex або тригер. Цей параметр вказує не тільки версію SOAP API, яку слід використовувати, але й версію Apex. Ви можете змінити версію після збереження. Кожен клас або ім'я тригера має бути унікальним. Ви не можете зберегти один і той же клас або тригер проти різних версій.

Ви також можете скористатися налаштуваннями версій для асоціації класу або тригера з певною версією керованого пакета, встановленого у вашій організації з AppExchange. AppExchange — вбудований магазин різних застосунків спеціально для вашої організації в Salesforce, щось схоже на Play Market від Google.

Попередньо вбудовані програми Salesforce забезпечують потужний функціонал CRM. Окрім цього, Salesforce надає можливість налаштувати попередньо вбудовані програми, що відповідають вашій організації. Однак у вашої організації можуть бути складні бізнес-процеси, які не підтримуються наявними функціоналами. У цьому випадку Lightning Platform (назва інтерфейсу користувача на платформі) надає різні шляхи для адміністраторів та розробників для створення спеціальних функцій.

Використовуйте Apex, якщо вам необхідно реалізувати:

- веб-служби;
- сервіси електронної пошти;
- Виконувати складну логіку, яка включає в себе роботу над декількома об'єктами.
- Складні бізнес-процеси, які не можна реалізувати вбудованими засобами платформи.
- Додавати логіку до іншої операції

Весь Apex працює повністю на вимогу на платформі Lightning. Розробники записують і зберігають код Apex на платформі, а кінцеві користувачі запускають виконання коду Apex через інтерфейс користувача. Apex збирається, зберігається та працює повністю на платформі Lightning (рис. 3.2).

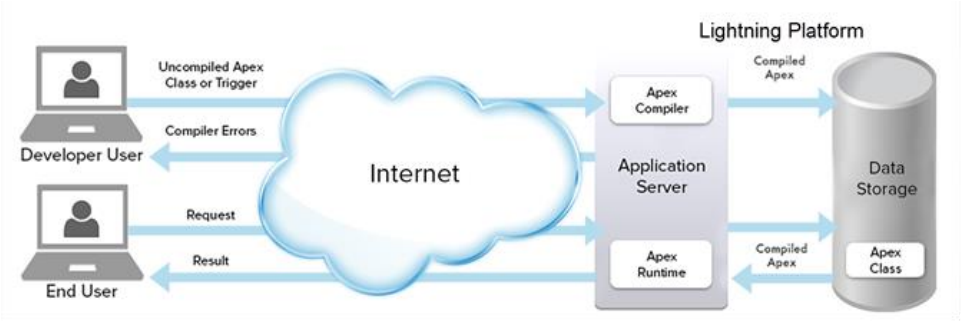

Рисунок 3.2 Схема компіляції та виконання Apex коду

Коли розробник пише і зберігає код Apex на платформі, сервер додатків платформи спочатку компілює код у абстрактний набір інструкцій, який може бути зрозумілий інтерпретатором виконання Apex, а потім зберігає ці інструкції як метадані.

Коли кінцевий користувач запускає виконання Apex, можливо, натиснувши кнопку в користувацькому інтерфейсі, сервер додатків платформи витягує складені інструкції з метаданих та надсилає їх через інтерпретатор виконання перед поверненням результату. Кінцевий користувач не помічає відмінностей у часі виконання від стандартних запитів на платформу.

3.4 Використання фреймворка Aura Components на Lightning

Aura Components – фреймворк для створення інтерфейсу користувача на платформі Salesforce. Lightning включає в себе Lightning Component Framework та деякі захоплюючі інструменти для розробників. Lightning полегшує побудову програм для будь-якого пристрою.

Lightning включає ці технології:

- Компоненти Lightning прискорюють розвиток та продуктивність додатків. Розробіть спеціальні компоненти, які інші розробники та адміністратори можуть використовувати як багаторазові будівельні блоки для налаштування мобільного додатка Salesforce та Lightning Experience.
- Lightning App Builder дає змогу адміністраторам створювати сторінки Lightning візуально, без коду, використовуючи нестандартні та вбудовані на замовлення компоненти Lightning. Зробіть доступними компоненти Lightning у програмі Lightning App Builder, щоб адміністратори могли створювати власні користувальницькі інтерфейси без коду.
- Experience Builder дає можливість адміністраторам створювати спільноти візуально, без коду, використовуючи шаблони та компоненти Lightning. Зробіть доступними компоненти Lightning у програмі Experience Builder, щоб адміністратори могли створювати сторінки спільноти без коду.

Використовуючи ці технології, ви можете безпроблемно налаштувати та легко розгорнути нові додатки на мобільних пристроях під управлінням Salesforce. Насправді, мобільний додаток Salesforce та Salesforce Lightning Experience побудовані з компонентів Lightning.

До переваг можна віднести нестандартний набір компонентів, архітектуру, керовану подіями, та рамку, оптимізовану для роботи.

Aura components надає набір готових компонентів, щоб розпочати створення програм. Вам не доведеться витрачати свій час на оптимізацію своїх додатків для різних пристроїв, оскільки компоненти дбають про це за вас.

Фреймворк надає можливість командам працювати швидше з стандартними компонентами, які безперебійно функціонують на настільних та мобільних пристроях. Створення програми з компонентами полегшує паралельний дизайн, підвищуючи загальну ефективність розробки.

Компоненти інкапсульовані, а внутрішні органи залишаються приватними, тоді як їх публічна форма видно споживачам компонента. Це сильне розмежування дає авторам компонентів свободу змінювати внутрішні деталі реалізації та ізолює споживачів компонентів від цих змін. Програми використовують чуйний дизайн та підтримують найновіші технології браузера, такі як HTML5, CSS3 та сенсорні події.

Компоненти Aura – це автономні та багаторазові одиниці програми. Вони представляють розділ користувальницького інтерфейсу для багаторазового використання і може відрізнятися деталізацією від одного рядка тексту до всього додатка.

Фреймворк включає набір попередньо вбудованих компонентів. Наприклад, компоненти, що поставляються зі Lightning Design System - бібліотека стилів Salesforce, доступні у просторі імен Lightning. Ці компоненти також відомі як основні компоненти Lightning. Ви можете збирати та налаштовувати компоненти для формування нових компонентів у додатку. Компоненти надаються для створення елементів HTML DOM в браузері.

Компонент може містити інші компоненти, а також HTML, CSS, JavaScript або будь-який інший веб-код. Це дає змогу створювати програми із складними інтерфейсами користувача.

Деталі реалізації компонента укладені в опис. Це дозволяє споживачеві компонента зосередитись на створенні програми, тоді як автор компонента може впроваджувати інновації та вносити зміни, не переривaючи роботу споживачів. Ви налаштовуєте компоненти, встановлюючи названі атрибути, які вони викривають у своєму визначенні. Компоненти взаємодіють зі своїм оточенням, слухаючи або публікуючи події.

Програмування на основі подій використовується у багатьох мовах та структурах, таких як JavaScript та Java Swing. Ідея полягає в тому, щоб ви писали обробники, які реагують на події інтерфейсу в міру їх виникнення.

Компонент реєструє, що він може викликати подію під час розмітки. Події запускаються з дій контролера JavaScript, які зазвичай ініціюються користувачем, що взаємодіє з користувальницьким інтерфейсом. Ви записуєте обробники в дії контролера JavaScript.

У фреймворку є два типи подій:

- Component events обробляються самим компонентом або компонентом, який інстанціює або містить компонент.
- Application events обробляються всіма компонентами, які слухають подію. Ці події, по суті, є традиційною моделлю публікації та підписки.

3.5 Звіти Salesforce та інформаційні панелі

Звіти Salesforce та інформаційні панелі можна підсумувати лише у двох пропозиціях: типи звітів – це шаблони, з яких користувачі створюють звіти, які організовують дані та передаються в папки. Звіти також є джерелом для компонентів інформаційної панелі, які візуально відображають дані на інформаційних панелях, які також передаються через папки.

Звіти та інформаційні панелі Salesforce складаються з декількох інтегрованих частин. Тип звіту визначає набір записів і полів, доступних для звіту, на основі зв’язків між первинним об’єктом та пов'язаними з ним об’єктами. Звіти відображають лише записи, які відповідають критеріям, визначеним у типі звіту. Salesforce пропонує набір попередньо визначених стандартних типів звітів; адміністратори також можуть створювати власні типи звітів.

Наприклад, адміністратор може створити тип звіту, який відображає лише контакти, які мають пов’язане з ними публікації, контакти без публікацій не відображатимуться у звітах такого типу.

Сам звіт повертає набір записів, який відповідає певним критеріям, і відображає їх в організованих рядках і стовпцях. Дані звітів можна фільтрувати, групувати та відображати графічно у вигляді діаграми. Звіти зберігаються у папках, які контролюють, хто має доступ.

Адміністратори контролюють доступ до інформаційних панелей, зберігаючи їх у папках із певними налаштуваннями видимості. Папки на інформаційній панелі можуть бути загальнодоступними, прихованими або обмеженими групами, ролями чи територіями. Якщо у вас є доступ до папки, ви можете переглянути її інформаційні панелі.

На кожній інформаційній панелі працює користувач, налаштування безпеки якого визначають, які дані відображати на інформаційній панелі. Якщо поточний користувач - це конкретний користувач, усі глядачі інформаційної панелі бачать дані, що базуються на налаштуваннях безпеки цього користувача, незалежно від їх особистих налаштувань безпеки. Для динамічних інформаційних панелей ви можете встановити працюючого користувача таким, що входить у систему, щоб кожен користувач бачив інформаційну панель відповідно до свого власного рівня доступу.

Папка - це місце, де можна зберігати звіти, інформаційні панелі, документи чи шаблони електронної пошти. Папки можуть бути загальнодоступними, прихованими або спільними, і їх можна встановити лише для читання або для читання / запису. Ви визначаєте, хто має доступ до його вмісту на основі ролей, дозволів, публічних груп та типів ліцензій. Ви можете зробити папку доступною для всієї організації або зробити її приватною, щоб мав доступ лише власник.

Знімок звітності дозволяє звітувати про історичні дані. Авторизовані користувачі можуть зберігати табличні або зведені результати звітів у полях на власному об’єкті, а потім зіставляти ці поля у відповідні поля цільового об’єкта. Потім вони можуть планувати, коли запустити звіт, щоб завантажити поля користувальницького об’єкта з даними звіту. Знімки звітів дозволяють вам працювати з даними звітів аналогічно тому, як ви працюєте з іншими записами у Salesforce.

3.6 Використання Salesforce Object Query Language (SOQL)

Для маніпуляції збережених даних в системі використовуються API-адреси мови запитів об’єктів Salesforce (SOQL) та API пошуку мови об’єктів Salesforce (SOSL) для пошуку даних даних Salesforce вашої організації.

Запит SOQL є еквівалентом оператора SELECT SQL і здійснює пошук у базі даних організації. SOSL - це програмний спосіб здійснення текстового пошуку щодо пошукового індексу.

Використовуєте ви SOQL або SOSL, залежить від того, чи знаєте ви, які об’єкти чи поля ви хочете шукати, а також інші міркування. Використовуйте SOQL, коли ви знаєте, в яких об’єктах знаходяться дані, і вам потрібно:

- Отримати дані з одного об'єкта або з декількох об'єктів, які пов'язані один з одним.
- Порахувати кількість записів, які відповідають заданим критеріям.
- Сортувати результати як частину запиту.
- Отримайте дані з полів.

SOSL необхідно використовувати в таких випадках:

- Отримати дані про певний термін, який, на вашу думку, існує в полі. Оскільки SOSL може токенізувати декілька термінів у полі та створити з цього індекс пошуку, пошук SOSL відбувається швидше і може повернути більш відповідні результати.
- Ефективно отримуйте декілька об'єктів і полів там, де об'єкти можуть бути не пов’язані один з одним.
- Отримайте дані для певного підрозділу в організації за допомогою функції підрозділів.
- Отримайте дані китайською, японською, корейською або тайською мовами. Морфологічна токенізація для термінів допомагає забезпечити точні результати.

SOSL може токенізувати декілька термінів у полі (наприклад, кілька слів, розділених пробілами) та будувати з цього індекс пошуку. Якщо ви шукаєте конкретний окремий термін, який ви знаєте, існує в полі, ви можете виявити, що

для цих пошуків SOSL швидше, ніж SOQL. Зменшіть до мінімуму кількість полів, які потрібно шукати чи запитувати. Використання великої кількості полів призводить до великої кількості перестановок, які важко налаштувати.

3.7 Тригери

Виконання Apex - коду можна викликати, використовуючи тригери. Тригери Apex дозволяють виконувати власні дії до або після змін у записах Salesforce, таких як вставки, оновлення чи видалення.

Тригер - код Apex, який виконується до або після наступних типів операцій:

- insert
- update
- delete
- merge
- upsert
- undelete

Наприклад, можна виконати тригер перед тим, як записи об’єкта вставлятимуться в базу даних, після того, як записи були видалені або навіть після відновлення запису з кошика.

Можна визначити тригери для стандартних об'єктів верхнього рівня, що підтримують тригери, такі як Контакт або Обліковий запис.

Існує два типи тригерів:

- Перед тим, як тригери використовуються для оновлення або перевірки значень записів до їх збереження в базі даних.
- Після тригерів використовуються для доступу до значень полів, встановлених системою (наприклад, поле Id або LastModifiedDate), а також для впливу на зміни в інших записах, такі як вхід у таблицю аудиту або запуск асинхронних подій з черги. Записи, які запускаються після тригера, є лише для читання.

Тригери можуть також змінювати інші записи того ж типу, що і записи, які спочатку спрацювали тригер. Наприклад, якщо тригер спрацьовує після оновлення контакту A, тригер також може змінювати контакти B, C та D. Оскільки тригери можуть спричиняти зміни інших записів, а тому, що ці зміни, у свою чергу, можуть спрацьовувати більше тригерів, Apex компілятор вважає всі такі операції єдиною одиницею роботи і встановлює обмеження на кількість операцій, які можна виконати для запобігання нескінченної рекурсії.

Крім того, якщо оновити або видалити запис у ньому перед тригером, або видалити запис у ньому після тригера, ви отримаєте помилку виконання. Сюди

входять як прямі, так і непрямі операції. Тригери, які виконуються після відміни запису, працюють лише з певними об'єктами. Значення поля не записується до кінця тригера. Якщо ви запитуєте історію полів у тригері, ви не бачите жодної історії для поточної транзакції.

Відстеження історії поля враховує дозволи поточного користувача. Якщо поточний користувач не має дозволу безпосередньо редагувати об’єкт або поле, але користувач активує тригер, який змінює об'єкт або поле з увімкненим відстеженням історії, історія змін не записується.

3.8 Використання Javascript

JavaScript був представлений у 1996 році, але лише до 1997 року був створений стандарт ECMAScript (ES), на якому базується JavaScript. Це була версія 1. Наступні дві версії з'явилися швидко і запропонували лише незначні вдосконалення. Потім з 1999 по 2009 рік JavaScript вступив у повільний період, коли не було нових випусків. У 2009 році було опубліковано ECMAScript 5 (або ES5, як його прийнято називати), і це версія JavaScript, з якою більшість розробників знайома. Однак лише до випуску ECMAScript 2015 (або ES6, як відомо) JavaScript вступив у те, що зараз відомо як період «Сучасного розвитку JavaScript».

JavaScript протягом майже 20 років мало змінювався. Але випуск ES6, який включав такі речі, як класи та модулі, викликав новий інтерес до розробки JavaScript. Раптом розробники зрозуміли, що JavaScript - це більше, ніж просто скриптова мова, з якою потрібно гратись. Розробники почали сприймати це як щось, що вони можуть використовувати для складання складних веб-додатків.

Із усім цим відновився інтерес до численних навчальних посібників, публікацій в блогах та бібліотек JavaScript з відкритим кодом. Як видно на часовій шкалі, випуск ES6 був лише початком. Релізи зараз приходять щороку, і вони називаються відповідно до року, коли вони виходять. І з кожним новим випуском з'являються нові захоплюючі функції, що сприяють наступній межі веб-розробки.

Для роботи JavaScript потрібен спеціальний двигун, схожий на JVM мови розробки Java. Усі основні веб-та мобільні браузери мають один, але існує безліч різних, з різними версіями. І всі вони не підтримують останню версію ECMAScript.

Це сповільнило прийняття багатьох нових функцій ECMAScript і є основною причиною, чому більшість розробників знайомі лише з синтаксисом ES5 (який зараз вважається JavaScript старої школи). Незважаючи на те, що існують способи обмеження браузера за допомогою транспілерів та поліфілів, вони вкладають більшу складність і можуть спричинити проблеми з

продуктивністю. Щоб дізнатися, які функції підтримуються типом двигуна, перейдіть на сторінку http://kangax.github.io/compat-table. В ній міститься інформація про всі доступні можливості Javascript, які підтримують різні браузери.

Перед ES6 єдиним способом можна було оголосити змінну чи функцію в JavaScript було використання ключового слова var. Тепер у вас є інші варіанти.

Для початку розглянемо трохи ближче, як визначаються змінні. Кажуть, що змінні, оголошені за допомогою ключового слова var, знаходяться в області функцій. Це означає, що змінна існувала б лише в межах функції, в якій вона була оголошена. Або найближчої батьківської функції, якщо це вкладена функція. Що має сенс. І це все ще має сенс, коли ви враховуєте глобальну область, в якій змінна оголошується поза функцією.

Відсутність обстеження блоків спричинило багато головної болі для розробників JavaScript, особливо якщо мова йде про змінні, оголошені для циклів. ES6 мав на меті усунути всі ці непотрібні труднощі, ввівши ключове слово let. Змінні, присвоєні let, завжди охоплюють блок. Але це не єдина користь від використання ключового слова let. Змінні, призначені таким чином, також не можуть бути усунені.

Піднімання відбувається, коли інтерпретатор JavaScript робить два проходи на ваш код. По-перше, оголошення змінної та функції "піднімаються" вгорі коду. А у другому проході вони оцінюються і виконуються завдання. Якби у нас був нікель кожного разу, то була б погано зрозуміла поведінка, яка викликала помилку.

З цієї причини багато розробників JavaScript зараз використовують let постійно. Використання змінних, що регулюються блоком, не тільки менш схильні до помилок, але полегшує іншим розробникам знати, як мається на увазі розмір змінної.

ES6 також ввів ще одне ключове слово, const. Цей тип змінної визначається як константа. Це може бути корисно, коли вам потрібно оголосити змінну, яку неможливо змінити або визначити заново. По суті, це лише для читання якщо ви декларуєте змінну як постійну, а потім спробуєте перепризначити її пізніше в коді, вона видає помилку типу. Змінні, задекларовані за допомогою ключового слова const, також мають блок-діапазон і не можуть бути замінені, що ви вже дізналися - це добре.

Однак слід пам’ятати про кілька речей при використанні ключового слова const. Оскільки значення const не можна перепризначити, вони повинні бути ініціалізовані під час їх декларування.

ES6 представив буквений текст разом із символом backtick (`). Літерали шаблонів дозволяють легко вставляти вирази в рядок, використовуючи позначення виразів, які повинні виглядати досить звично. Літерали шаблону пропонують безліч переваг, таких як:

1. Можливість вставляти одинарні та подвійні лапки без використання символів втечі.
2. Багаторядкові повідомлення, які чудово підходять, коли вам потрібно створити рядок, що містить розмітку HTML, що охоплює кілька рядків.
3. Шаблони з тегами, які дозволяють запускати шаблон через створену функцію. Це дає вам більший контроль над тим, як виглядає отриманий рядок.

4. Опис моделі даних програмного продукту

4.1 Створення моделі даних

Основне призначення платформи Salesforce це робота з даними. В архітектуру системи вбудована реляційна база даних. В якості таблиць тут використовується поняття об’єкта. Salesforce надає графічний інтерфейс для керування базою даних, налаштування системи та можливість створення бізнес логікі декларативними способами розробки. Тому вся робота з Salesforce починається з побудови моделі даних (схеми БД).

При створені вашої організації на платформі автоматично створюються стандартні об’єкти. Вони мають більше функціоналу в порівнянні зі створеними кастомними об’єктами. Це пов’язано з тим, що реалізація цих об’єктів зашита в самій системі. Використання цих об’єктів не є обов’язковим, але рекомендується так і робити. Поля стандартного об’єкта, можна налаштувати так, що вони не будуть використовуватись. Видалити стандартний об’єкт і його стандартні поля не можна.

Основна ідея бізнес процесу — автоматизація роботи кафедри. Основні юзери системи це:

- Адміністрація
- Викладачі
- Студенти

Функціонал для адміністрації:

- Перегляд всіх даних системи, за допомогою наданого платформою Salesforce графічного дизайну і можливість вносити в них зміни
- Генерація звітів за допомогою засобів Salesforce. Через те, що платформа надає можливість вести всі свої записи в ній, користувачу потрібно лише створити шаблон звіту і вибрати необхідні для нього дані і їх параметри. Система сама перевірить всі дані, які включає в себе і згенерує звіт по них.
- Зворотній зв’язок з іншими користувачами.

Функціонал для викладачів:

- Особистий кабінет
- Виставлення оцінок студентам в середині системи
- Виставлення пропущених студентом занять
- Прикріплення домашнього завдання для студентів до конкретної пари
- Перегляд груп, в яких викладач проводить пари
- Перегляд всіх предметів, які викладач веде
- Викладач, який займається дослідницькою діяльністю пов’язує свій особистий кабінет з дослідженнями, які він проводить.

Функціонал для студентів:

- Перегляд опису кожного предмету
- Доступ до оцінок, пропущених пар, домашнього завдання
- Перегляд повідомлень від адміністрації в середині системи

Модель бази даних, яка буде забезпечувати весь цей функціонал наведена нижче.

На рис. 4.1 зображено частину моделі бази даних, яка відповідає за забезпечення навчального процесу.

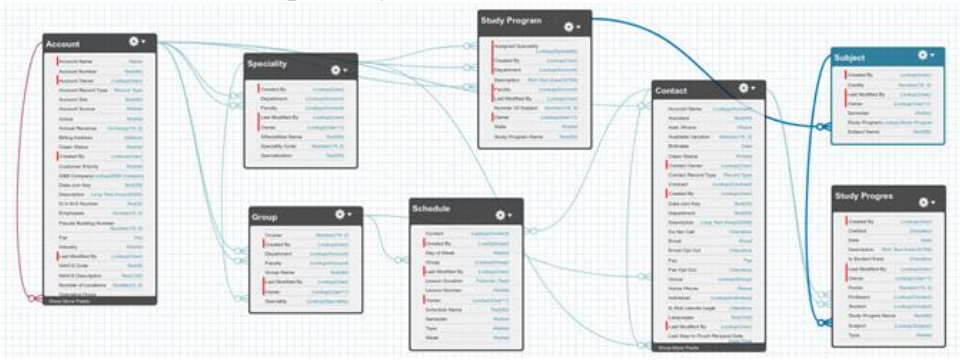

Рисунок 4.1 Опис схеми БД для навчального процесу

На рис. 4.2 зображено частину моделі даних, яка допомагає вести внутрішню документацію університету, моніторинг зовнішній зв’язків університету.

Рисунок 4.2 Опис схеми БД для контролю зовнішніх зв’язків і внутрішньої документації

4.2 Детальний опис об’єктів бази даних

Об’єкт Account — описує кафедри, факультети, компанії з якими університет має зовнішні зв’язки або користується їхніми послугами. Має три типи записів:

- faculty - факультет
- department - кафедри на факультетах
- Partner - опис всіх партнерів університету

Поля даного об’єкта:

- Account Name - назва факультету
- BillingAddress - адреса факультету
- Faculty_Building_Number__c - номер корпусу
- Description - опис факультету
- ParentId - зв'язок між факультетом і кафедрою(це поле доступне для заповнення тільки для аккаунтів з типом запису department)
- Phone - контактний телефон
- Employees- кількість співробітників
- Students - кількість студентів на факультеті (та сама логіка, що і для Employees)
- Type - тип факультету faculty or institute
- YearStarted - рік заснування факультету (тільки для записів типу faculty і department)
- Website - сайт аккаунта

Об’єкт Contact - опис контактів університету. Загальні поля об’єкта:

- name
- phone
- email
- аватар
- дата народження

Даний об’єкт має три типи записів, для кожного з яких відкриваються різні поля для заповнення.

1. Тип запису — адміністрація. Спеціальні поля:
   - доступні дні відпустки
   - Кількість днів проведені на лікарняному
   - посада
   - зарплата
2. Тип запису — Викладачі. Спеціальні поля:
   - доступні дні відпустки
   - дні проведені на лікарняному
   - зарплата
   - лукап на контракт
3. Тип запису — Студенти. Спецільні поля:
   - стипендія

4. Тип запису — Партнери. Спеціальні поля:

- тип співпраці

Об’єкт Speciality - описує спеціальність на якій навчається студент. Поля:

- Код спеціальності
- Назва спеціальності
- Назва спеціалізації
- дочірній об’єкт факультету
- дочірній об’єкт кафедри
- батьківський об’єкт учбової програми
- батьківський об’єкт учбових предметів
- батьківський об’єкт для студентів

Об'єкт Група — описує інформацію про групу студентів. Поля:

- код групи
- курс групи

Є дочірнім об’єктом таких об’єктів:

- спеціальність
- кафедра
- факультет

Об’єкт Дисципліна - опис учбового предмету

- Ім'я дисципліни
- Кредити
- Номер семестру (picklist 1 or 2)
- є парентом для записів успішності студентів
- дочірній об’єкт Навчальної програми

Студенти і викладачі будуть діставати ці записи через учбовий план або спеціальність, також цей об’єкт прив'язаний до розкладу.

Об'єкт Навчальна програма описує навчальну програму для певної спеціальності. Прив'язаний до спеціальності, факультету і кафедри. Містить дочірні об'єкти з предметами, які входять до програми. Поля:

- Кількість дисциплін - кількість прив'язаних до навчальної програми предметів
- Описання - короткий опис навчальної програми
- Статус - пікліст який описує стан навчальної програми

Об'єкт Прогрес навчання зберігає дані про успішність студента. Поля :

- оцінка (бали)
- присутність
- дата
- дочірній об’єкт Дисципліни
- дочірній об’єкт Студента

- дочірній об’єкт викладача, який поставив оцінку
- зараховано — поле типу boolean
- тип запису (Лекція, Лабораторна, Практика, Курсова робота, Консультація, Диплом, Переддипломна практика, Сумарні результати, Домашнє завдання)

Об'єкт Schedule описує запис пари для групи і викладача. Поля:

- лукап на групу
- лукап на викладача
- номер пари(1-5)
- семестр (1-2)
- часовий проміжок пари
- номер тижня(перший тиждень, другий тиждень)
- день тижня
- тип (лекція, лабораторна, практика)

Об’єкт Контракт описує інформацію про контракти з різними підприємствами, студенти, що навчаються на контрактній формі навчання. Поля:

- дочірній об’єкт контакта
- опис
- дата початку
- дата кінця
- ціна
- ім'я

5.Опис створеного програмного продукту

5.1 Вимоги до комп’ютера і програмного забезпечення

Salesforce насправді має дуже мало технічних вимог. Оскільки це програмне забезпечення, засноване на хмарі, вам не потрібно встановлювати це програмне забезпечення на свій комп’ютер.

Вимоги Salesforce прості, все, що вам потрібно, це комп'ютер з підключенням до Інтернету, а також підтримуваний браузер:

- Google Chrome
- Mozilla Firefox
- Microsoft Edge
- Apple Safari

Очевидно, ви також хочете мати швидке підключення до Інтернету, щоб отримати максимальну користь від Salesforce.

5.2 Використання системи

Платформа Salesforce буде виконуватись двома різними способами:

- Робота на платформі адміністрації. Цей тип роботи буде використовувати стандартний користувацький інтерфейс Salesforce для роботи з даними, які зберігаються в Salesforce, а також використання певних вбудованих функцій(генерація звітів).
- Робота на платформі викладачів та студентів. Для цього створена спеціальна сторінка, яка буде доступною за URL-адресою. З першого погляду виглядє як звичайний сайт, проте він повністю інтегрований з Salesforce. Будь-які зміни додані через даний сайт автоматично будуть зберігатись на платформі.

Стандартний користувацький інтерфейс, який буде використовувати адміністрація зображено на рис. 5.1

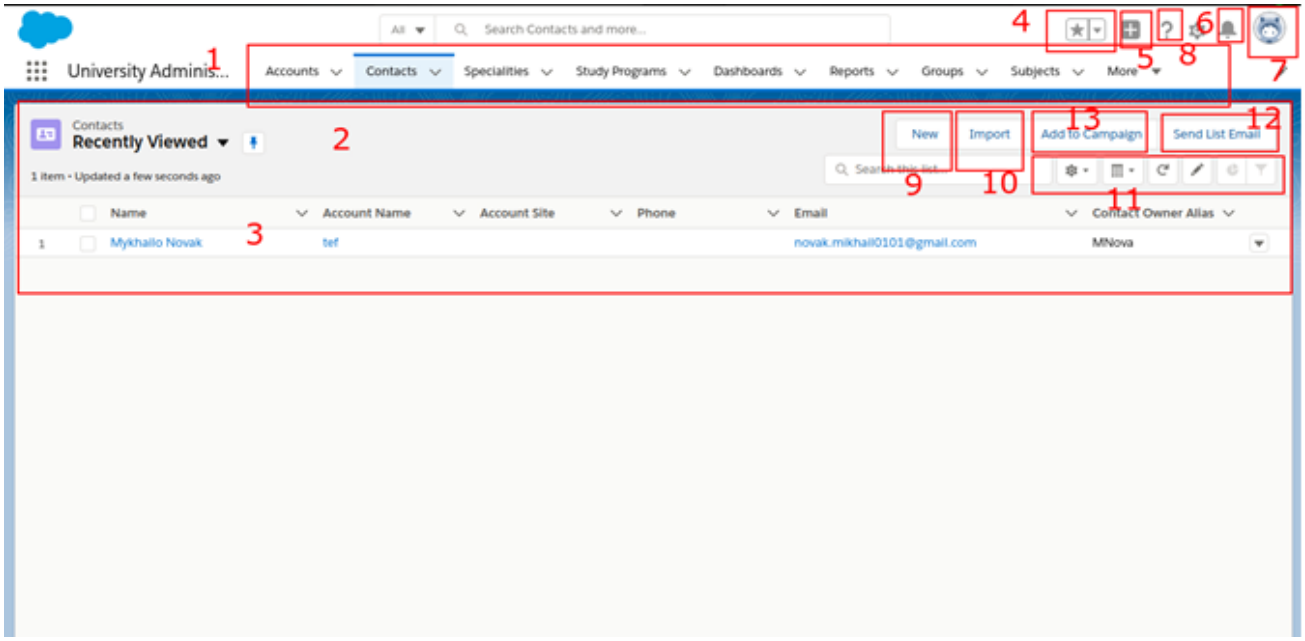

Рисунок 5.1 – Сторінка управління записами об’єктів

1. Вкладки вибору об’єкта
2. Вікно створених записів об’єкта
3. Запис об’єкта, при натисканні відкриє детальний опис запису
4. Кнопка Favorites, додає сторінку в список найбільш часто використовуваних сторінок, при натисканні стрілочки це список відкривається. Додавайтесторінки, які часто використовуєте для швидкого доступу до них
5. Глобальні дії. Перелік дій, які не стосуються конкретного запису чи об’єкту
6. Сповіщення користувача
7. Іконка аватара користувача і налаштування профілю
8. Центр допомоги
9. Створення нового запису відкритого об’єкта
10. Імпортування записів об’єкта з інших ресурів
11. Вікно налаштування і фільтрації доступних записів
12. Кнопка розсилки e-mail листів. Для цього необхідно вибрати бажані контакти і натиснути цю кнопку. Потім ввести бажаний текст або вибрати шаблон листа і надіслати його. Електронні адреси отримувачів зберігаються в записах об’єкта. Система автоматично дістане ці значення і відправить листи від імені користувача.
13. Додати вибрані записи до рекламної кампанії (в рамках призначення конкретно нашого бізнес процесу, цей функціонал не реалізований)

Після вибору конкретного запису, відкривається нове вікно(рис. 5.2) з детальною інформацією

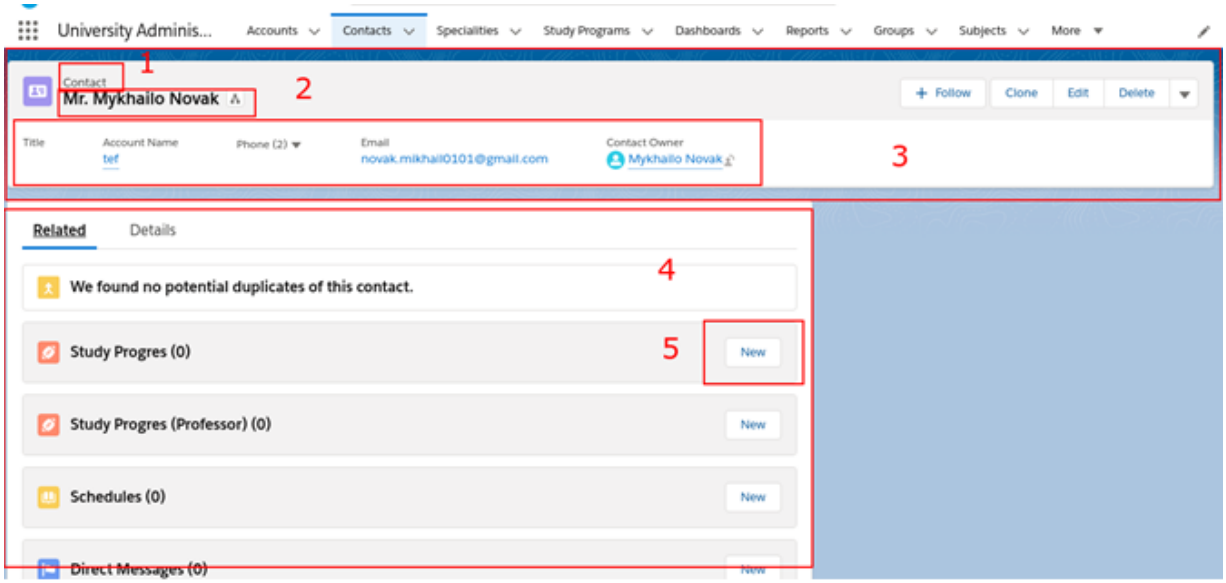

Рисунок 5.2 – Детальний опис запису об’єкта

1. Тип об’єкта
2. Ім’я запису
3. Коротка інформація про запис (налаштовується)
4. Вкладка дочірніх записів, асоційованих з даним
5. Кнопка створення нового дочірнього запису, який буде пов’язаний з даним.

При натисканні вкладки Details вигляд зони 4 зміниться і буде відкрита інформація зі значеннями інших полів запису (рис. 5.3).

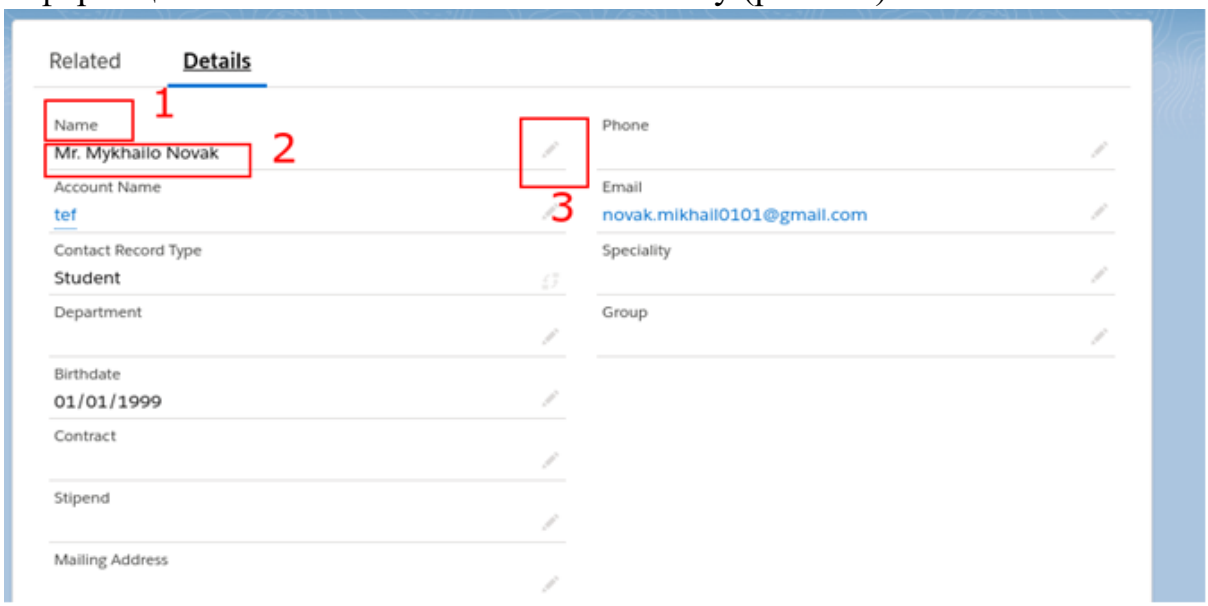

Рисунок 5.3

1. Назва поля
2. Значення поля

3. Кнопка редагування. Після її натискання значення полів можуть бути змінені і збережені

Стандартний функціонал Salesforce також надає потужну функцію — генерацію звітів. Вона надає можливість сумувати всю інформацію в один файл. Інтерфейс створення звітів показаний на рис. 5.4

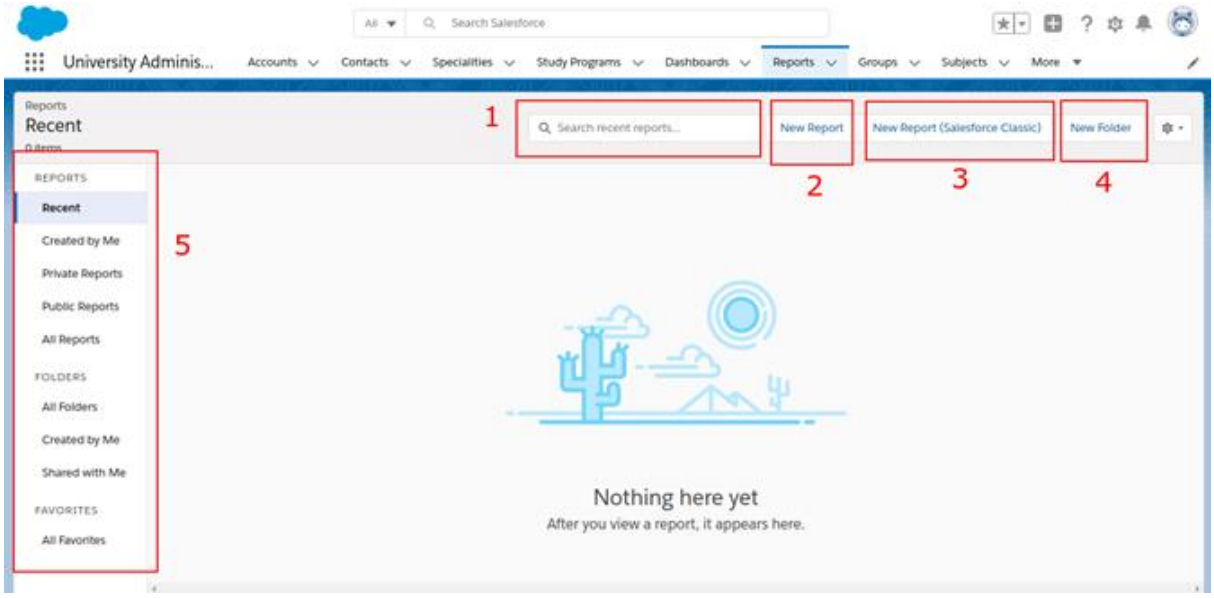

Рисунок 5.4

1. Пошук створених шаблонів звіту за назвою
2. Створення нового шаблону
3. Створення шаблону звіту для застарілого користувацького інтерфейсу. Ця функція збережена для підтримки користувачів із старими версіями платформи.
4. Створення нової папки для шаблонів звіту
5. Інструменти фільтрування звіту

Далі коли шаблон звіту створений, можна в будь-який момент часу вибрати його і згенерувати звіт за вибраним шаблоном який буде містити актуальні на момент генерування дані.

Для роботи викладачів і студентів був створений кастомний дизайн сайту, який буде доступний за посиланням.

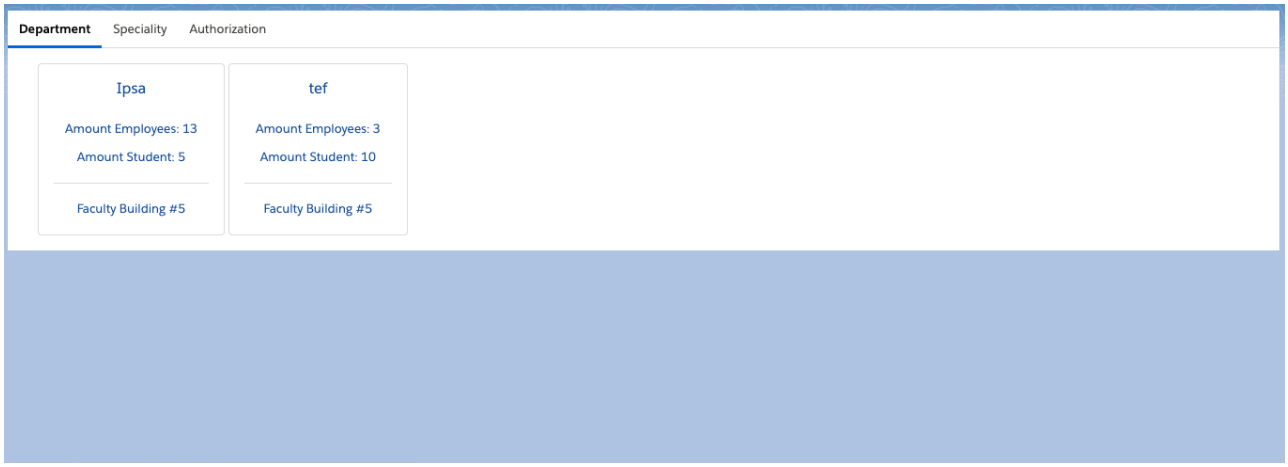

Рисунок 5.5 Сторінка вибору факультету та кафедри

На рисунку 5.5 відкривається список всіх факультетів університету, після вибору якого, відкривається таке ж вікно тільки з вибором кафедри. Після вибору кафедри відкривається, її сторінка (рис 5.8). Тобто користувачу потрібно в кінцевому результаті вибрати необхідну йому кафедру. Вибір кафедри можна здійснити за допомогою таби Спеціальність, де після вибору спеціальностей стають доступні кафедри, які навчають за даним напрямом.

Користувачу на вибір дається можливість трьох варіантів вибору кафедри.

1. Вибір факультету, після чого надається вибрати кафедру факультету (рис. 5.5).
2. Вибір спеціальності, після чого вибирається факультет, який надає підготовку за вибраним напрямом і після вибору факультету відкривається можливість вибору кафедри (рис. 5.6).
3. Користувач може відразу залогінитись в систему і кафедра, до якої він відноситься, автоматично вибереться (рис 5.7).

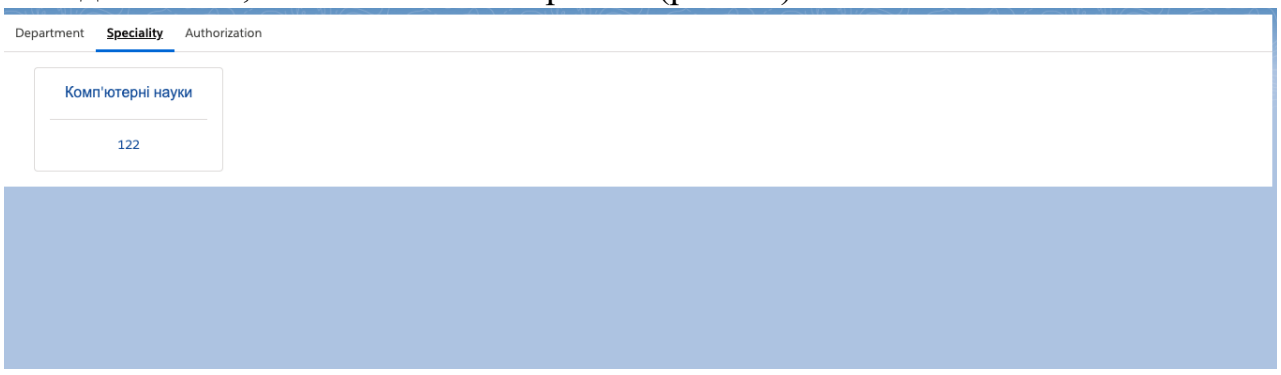

Рисунок 5.6 Сторінка вибору спеціальності

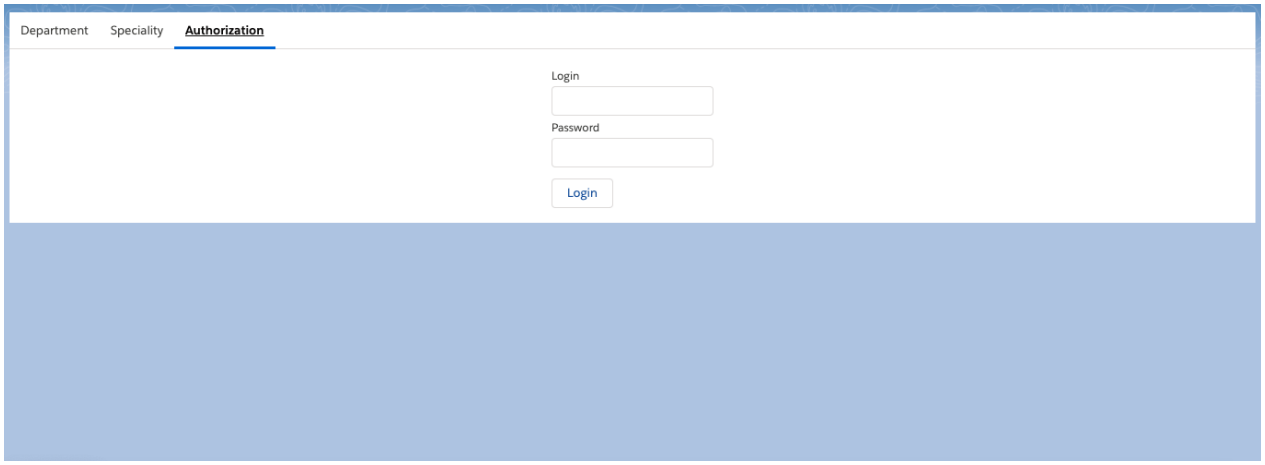

Рисунок 5.7 Сторінка авторизації користувача

Навігація між варінтами вибору кафедри здійснюється за допомогою кнопок, що розташовані зверху.

Роглязнемо сторінку кафедри.

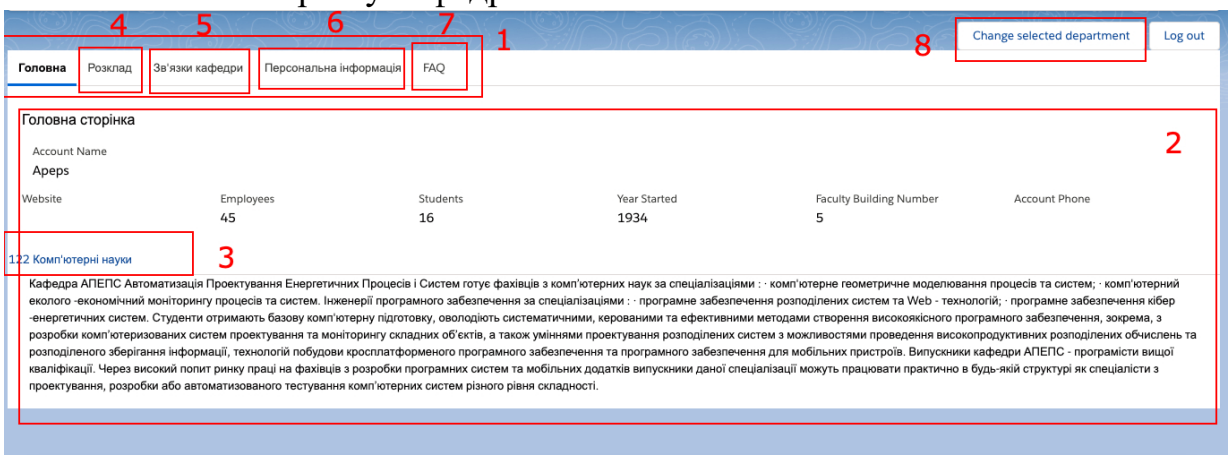

Рисунок 5.8 Сторінка кафедри

1. Панель навігації
2. Опис кафедри
3. Спеціалізції кафедри, після вибору яких відкривається їх опис
4. Вкладка розкладу (аналог rozklad.kpi.ua)
5. Зовнішні зв'язки кафедри (відкривається інформація про зовнішніх партнерів, містить в собі актуальну інформацію про програми обміну студентів, лабораторії цих компаній на кафедрі, доступні бази практики, вакансії на роботу за спеціальностями)
6. Вкладка персональної інформації. Після натискання відкривається вікно авторизації після якої користувачу доступна його персональна інформація.
7. Вкладка з відповідями на популярні питання
8. Кнопка вибору іншої кафедри

Рисунок 5.9 Сторінка розкладу занять

В даному прикладі інтерфейсу користувач має можливість вибрати будь-яку активну групу, яка зараз навчається, натиснувши на її ім'я, після чого відкриється розклад групи.

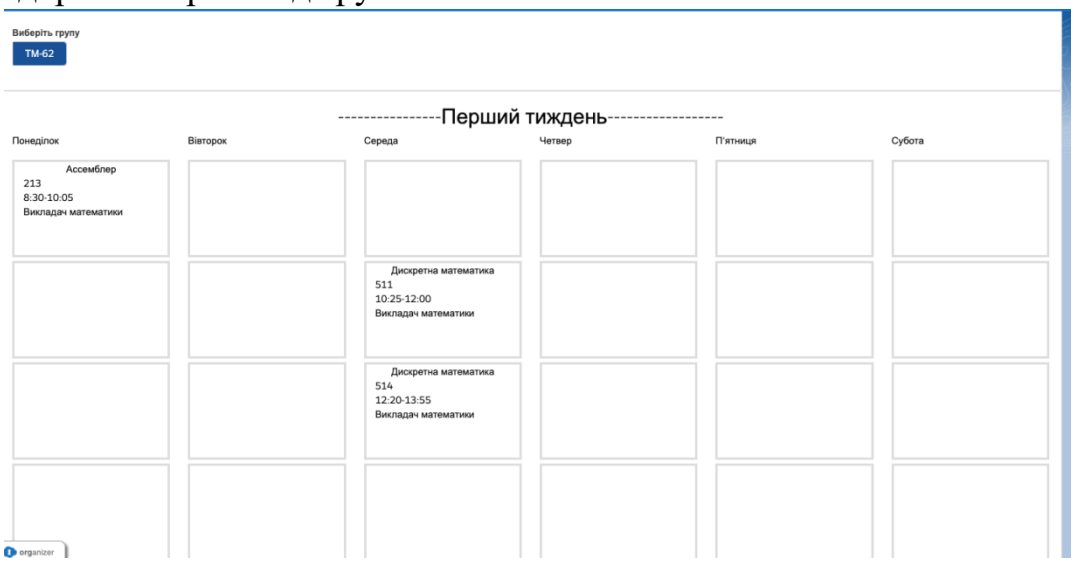

Рисунок 5.10 Розклад занять вибраної групи

Для викладачів розклад виглядає аналогічно, окрім того, що замість груп виводяться всі викладачі прив'язані до кафедри.

Рисунок 5.11 Вибір розкладу викладача

Кафедра обов'язково має зовнішні зв'язки. Це можуть бути вакансії на роботу по спеціальності, конференції, стажування, навчальні курси, певні події для студентів. Всю цю корисну інформацію для студентів відображає вкладка "Зв'язки кафедри" (рис. 5.12).

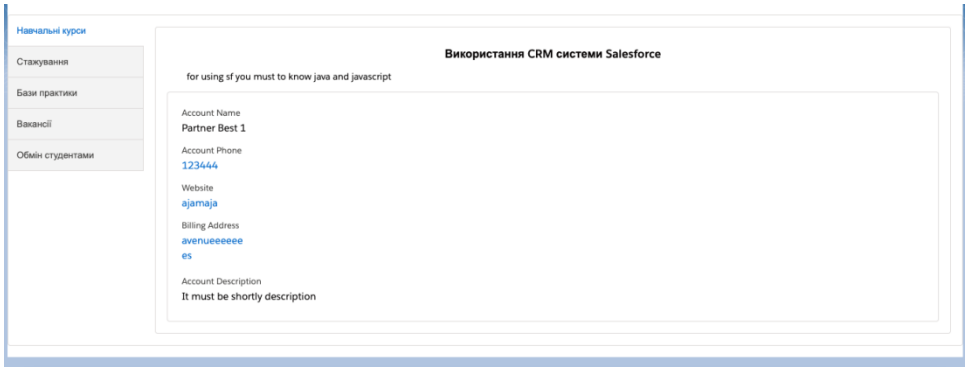

Рисунок 5.12 Вкладка зв’язків кафедри

Вкладка відображає тему показаної інформацію, опис типу зв’язку і контактну інформацію з партнером кафедри, який надає певні можливості для студента. Навігаційна панель зліва дозволяє користувача змінювати тип відображуваних зв’язків. Всі записи зв’язків відображаються у вигляді списку у вкладці.

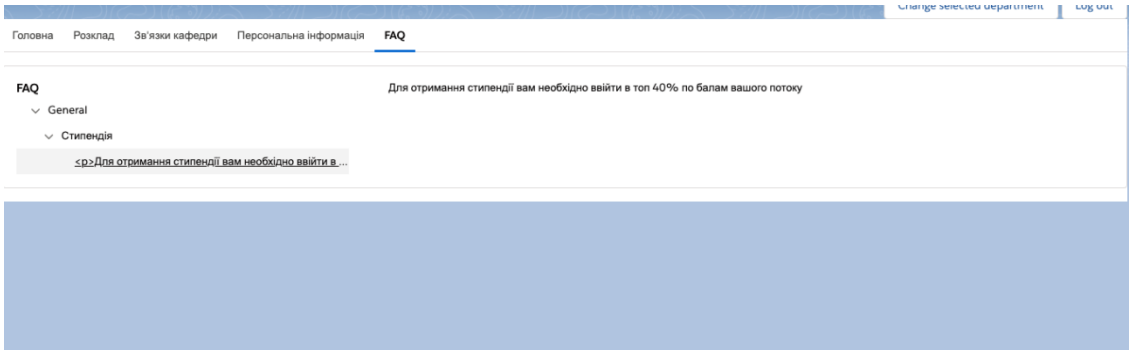

Рисунок 5.13 Вкладка FAQ

У користувачів завжди виникають питання. Відповіді на найбільш популярні питання містить вкладка FAQ (рис. 5.13). Зліва розташована навігаційна панель з переліченими запитаннями. Вони відображається у вигляді списку. Всі запитання груповані за темою. Після вибору теми користувач має вибрати цікаве йому питання, після чого вибрати відповідь і вона відобразиться на основному полотні справа.

Вкладка персональної інформації потребує обов’язкової авторизації користувача. Якщо ви на початковій сторінці для вибору кафедри використовували авторизацію, то вводити свої логін і пароль на сторінці персональної інформації не треба. Для авторизації необхідно ввести логін і пароль, після чого система визначає тип користувача (студент чи викладач) і відкриває нове вікно для відповідного типу.

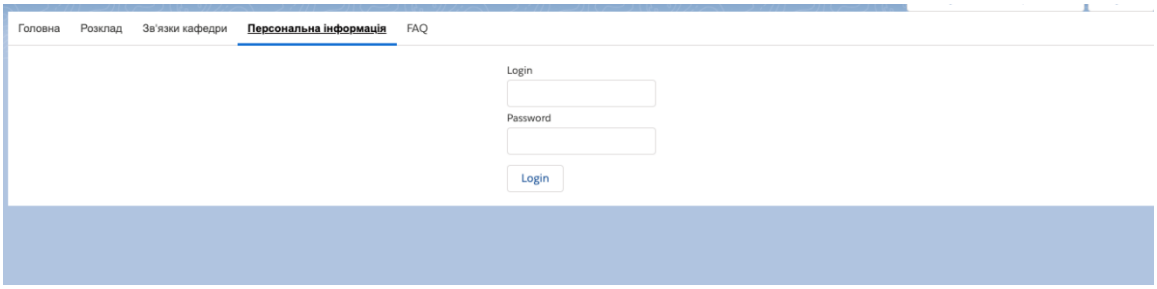

Рисунок 5.14 Вікно авторизації, для доступу до персональних даних

У випадку якщо користувач студент відкривається вікно зображене на рисунку 5.15

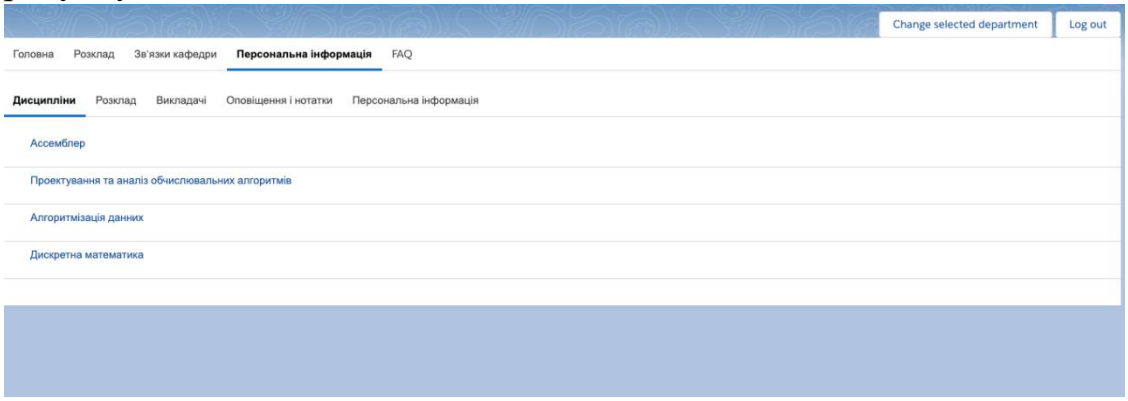

Рисунок 5.15 Вікно персональних даних студента

Вкладка дисциплін, містить перелік всіх дисциплін, які на даний момент вивчає студент. Користувач має вибрати необхідний йому предмет для перегляду своєї успішності по ньому (рис. 5.16). На відкритій сторінці відображається назва предмету, викладач і записи успішності студента у вигляді таблиці. Кожен запис таблиці містить тему пари, на якому була виставлена оцінка, кількість балів, тип заняття, присутність студента і дату проведення. Для вибору іншого предмета користувачу необхідно натиснути кнопку Back після чого відкриється вкладка з усім предметами студента.

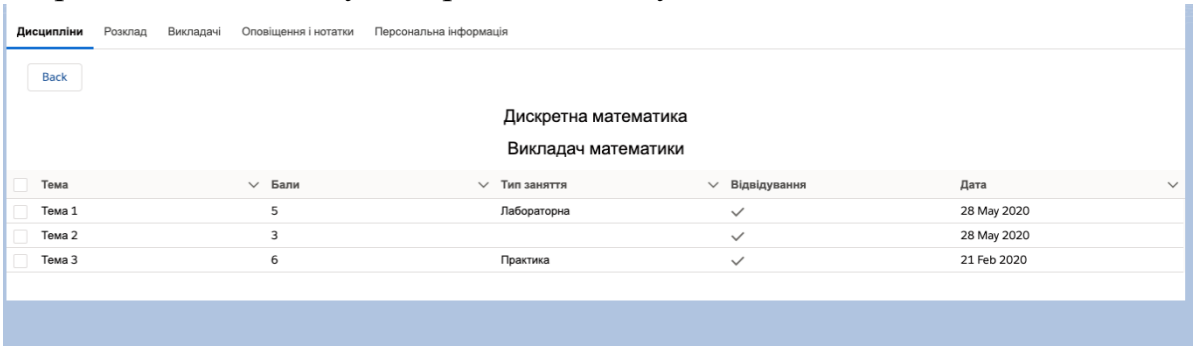

Рисунок 5.16 Успішність студента

Вкладка розклад в персональній інформації відображає актуальний розклад для користувача. Він автоматично завантажується при авторизації. В ньому немає можливості переглянути розклад іншої групи чи викладача.

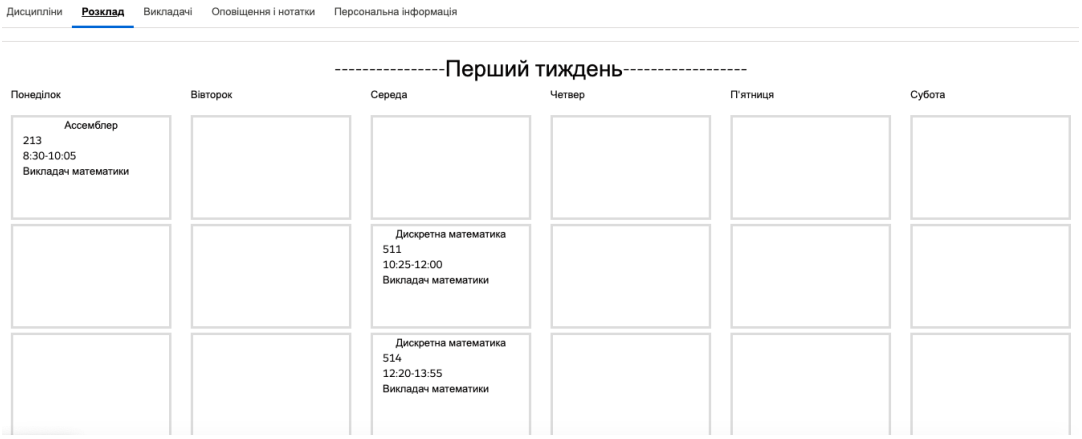

Рисунок 5.17 Персональний розклад студента

Вкладка “Викладачі” містить список всіх викладачів, які ведуть предмети у студента. Для відображення детальної інформації про викладача користувач має натиснути на його ім’я. Після вибору викладача відкривається вкладка з контактною інформацією викладача і список предметів, які він веде. Користувач може вибрати предмет і передивитись актуальну інформацію успішності по даному предмету.

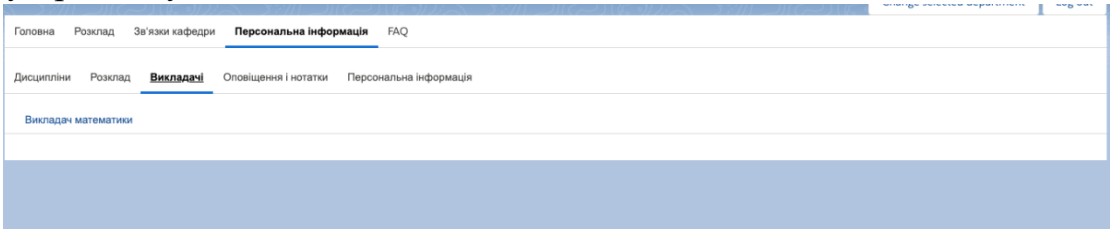

Рисунок 5.18 Список викладачі

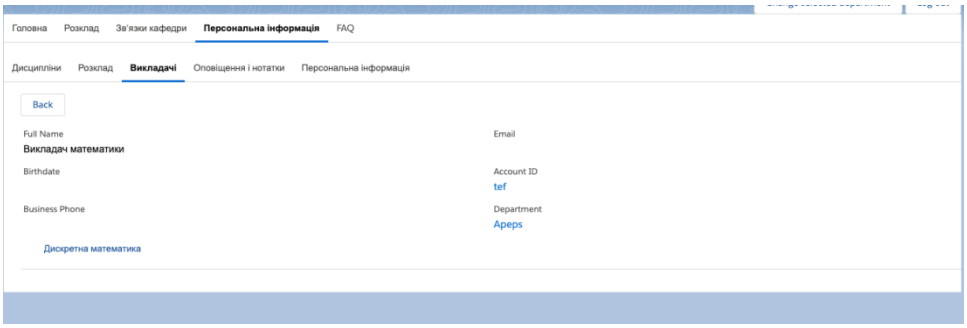

Рисунок 5.19 Контактна інформація викладача та список предметів

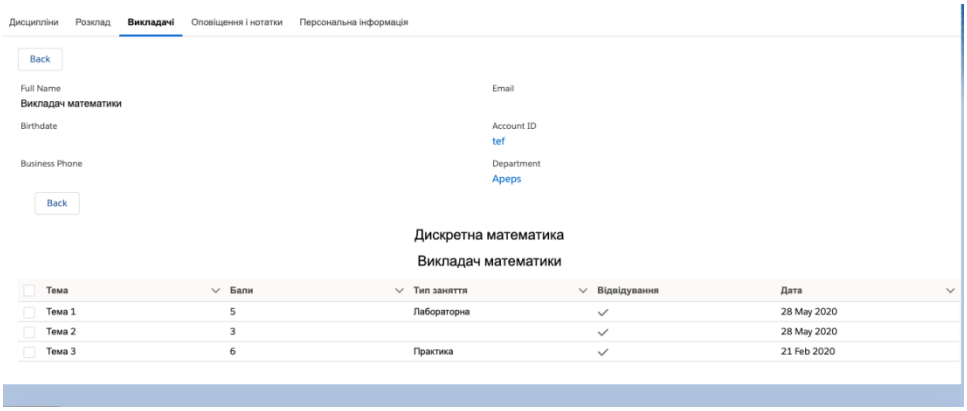

Рисунок 5.20 Успішність студенти по вибраному предмету

В розробленій системі адміністрація має можливість надіслати повідомлення студенту. Для цього необхідно створити запис повідомлення і воно автоматично відобразиться на вкладці персональної інформації.

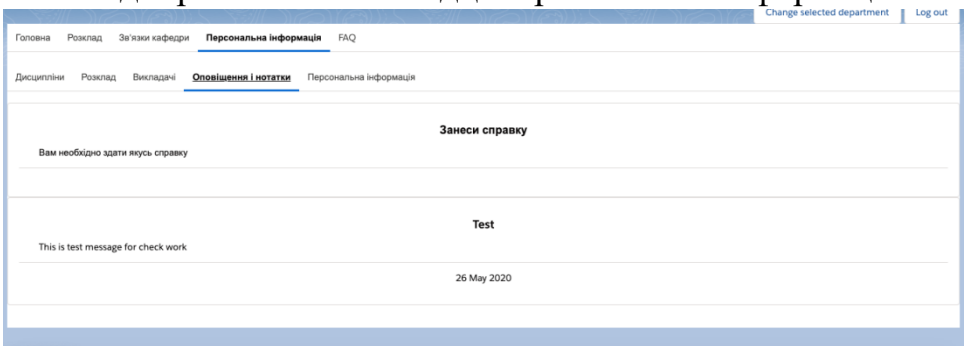

Рисунок 5.21 Оповіщення студента

Авторизований користувач також має можливість переглянути свою контактну інформацію, яка зберігається в системі.

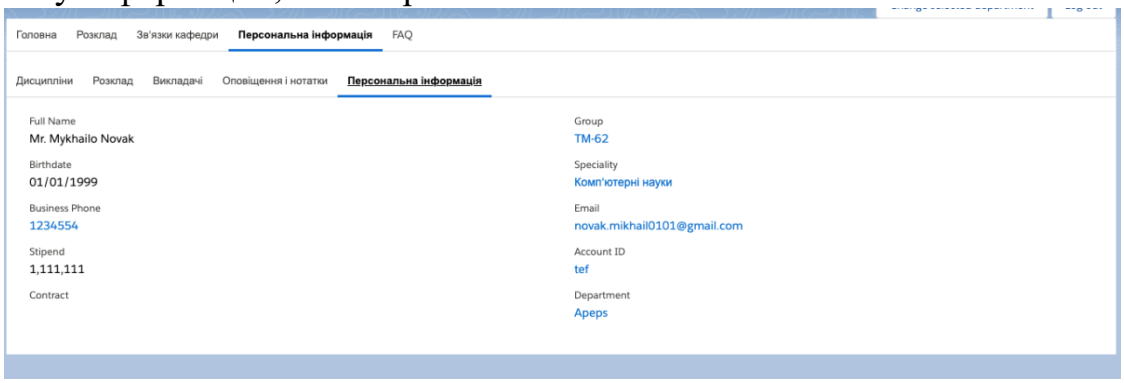

Рисунок 5.22 Контактна інформація авторизованого користувача

У випадку якщо користувач викладач відкривається вікно зображене на рисунку 5.23

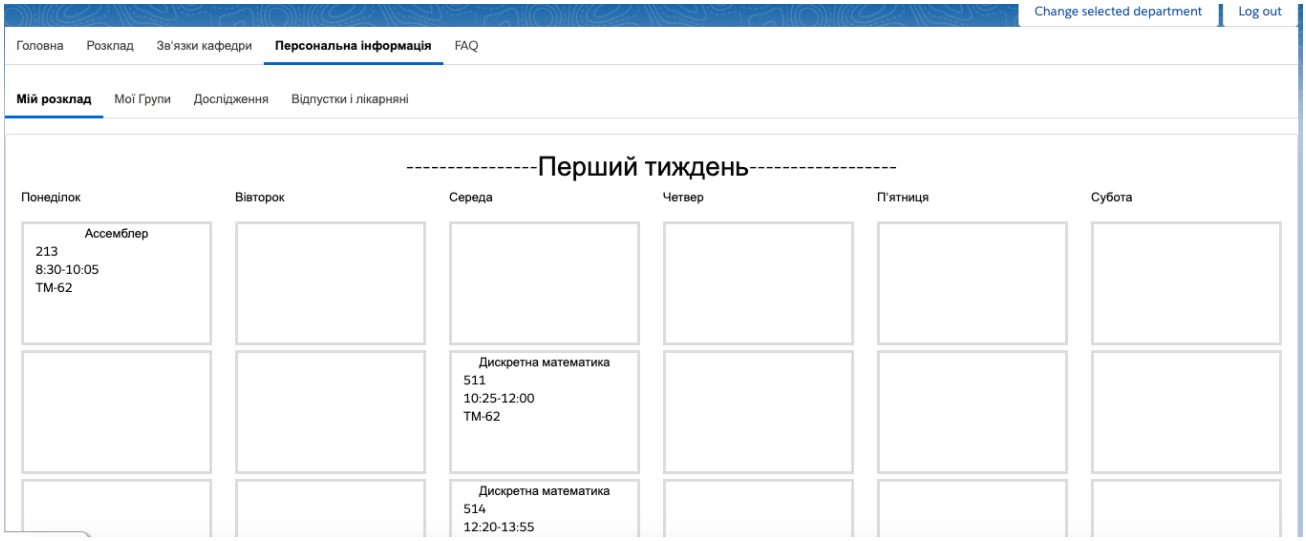

Рисунок 5.23 Вікно персональних даних викладача

Перша вкладка, яка відкривається для користувача типу викладач, це його персональний розклад.

Вкладка “Мої групи” відображає всі групи в яких викладач веде дисципліни (рис. 5.24). Користувачу необхідно вибрати групу, після чого у вкладці відображається нова інформація, яка містить назву групи, опція вибору предмету, якщо в одній групі викладач веде декілька предметів, список студентів групи (рис. 5.25). Коли користувач вибирає студента зі списку, натиснувши на нього відображається успішність студента по вибраному предмету (рис.5.26). Користувач має можливість редагувати записи успішності і додавати нові записи успішності.Для редагування запису необхнідно навести на необхнідну комірку таблиці, після чого зв’явицька кнопка із зображенням олівця натиснувши яку включається режим редагування комірки (рис. 5.27). Змінені підсвічуються жовтим фоном і зв’являються кнопки збереження змін і відміни змін. Для додавання нового запису успішності необхідно натиснути кнопку “Новий запис”, після чого відкриється нове вспливаюче вікно, де користувач вводить необхідні дані і створює новий запис (рис. 5.28).

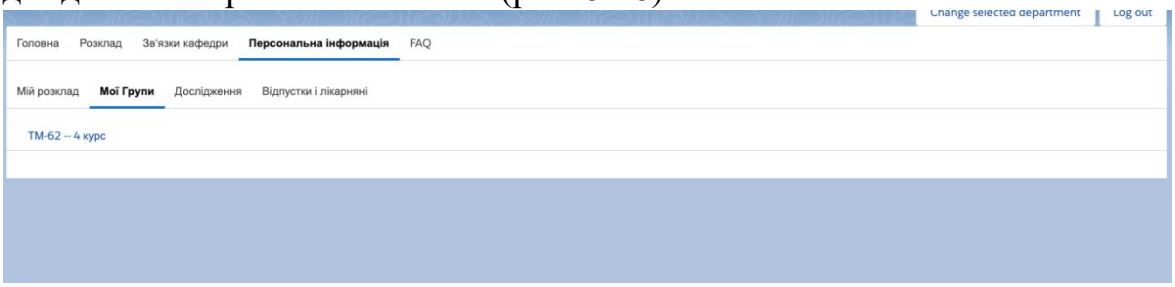

Рисунок 5.24 Список груп викладача

Рисунок 5.25 Опція вибору предмету і список групи

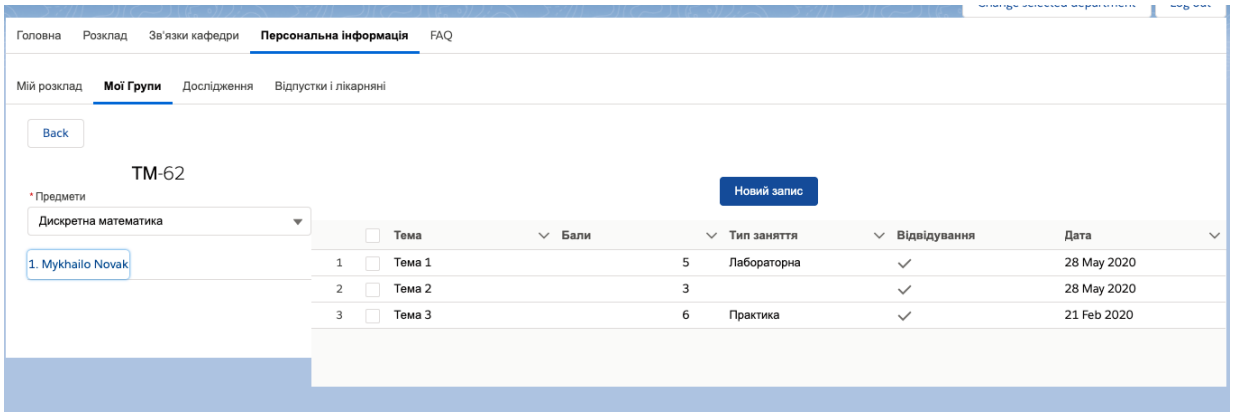

Рисунок 5.26 Успішність студента

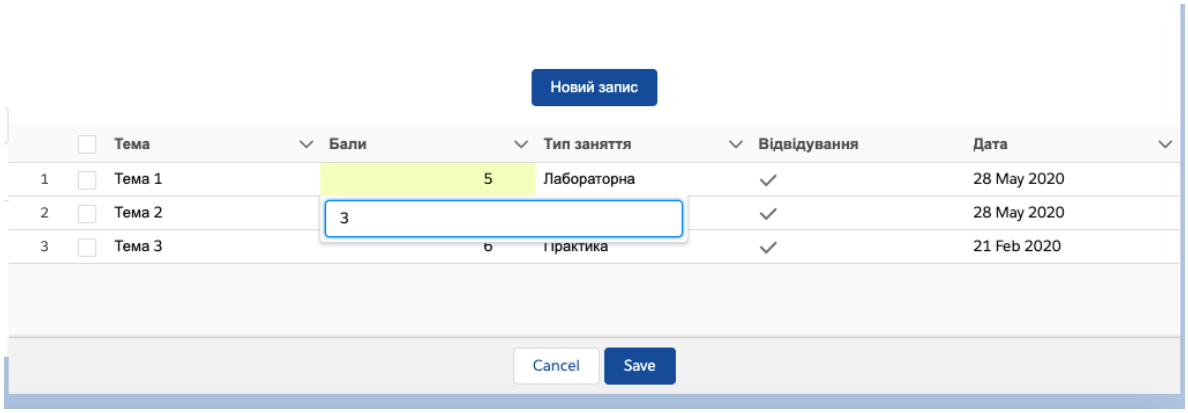

Рисунок 5.27 Режим редагування записів

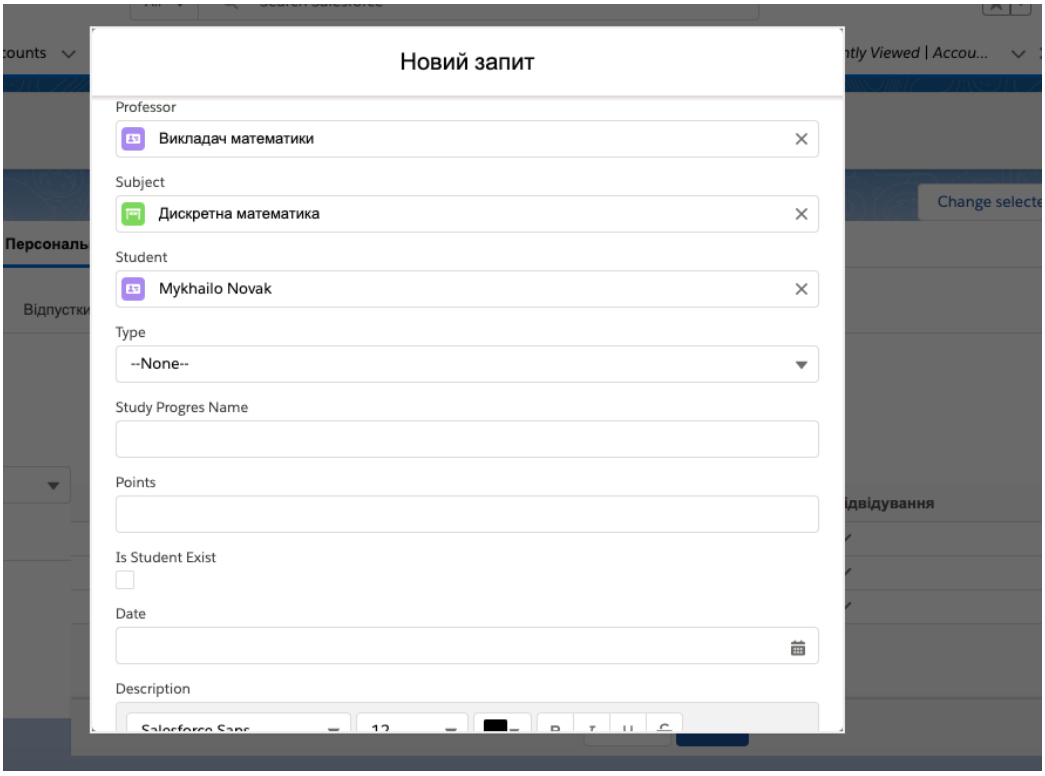

Рисунок 5.28 Вікно створення нового запису успішності

Вкладка “Дослідження” містить інформацію про наукову діяльність викладача. Відображається у вигляді списку блоків. В кожному блоці відображається тема дослідження, посилання на публікацію, короткий опис і статус дослідження.

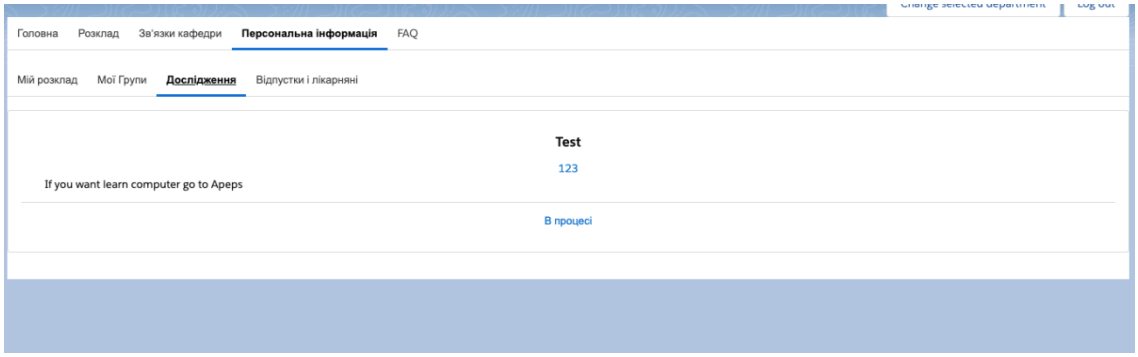

Рисунок 5.29 Наукові роботи викладача

Вкладка “Відпустки і лікарняні” надає функціонал для відстеження інформації про відпустки викладача. Користувач бачить кількість доступних днів відпустки, кількість лікарняних і чи були вони легальні чи ні. Також викладач може створити новий запит відпустки за допомогою кнопки “Новий запит відпустки/лікарняного”, після натискання якої відкривається нове вікно створення запису. Під кнопкою відображаються дати минулих відпусток і лікарняних. Доступні дні відпутски і лікарняних автоматично оновлюються системою за допомогою Apex Schedule Batch Job, що є асинхронним

інструментом програмування мови Apex. Кожного місяця даний функціонал нараховує півтора дня відпустки кожному викладачу. Також кожного року минулі відпустки, кількість доступних днів обнуляються за допомогою цього ж інструмента.

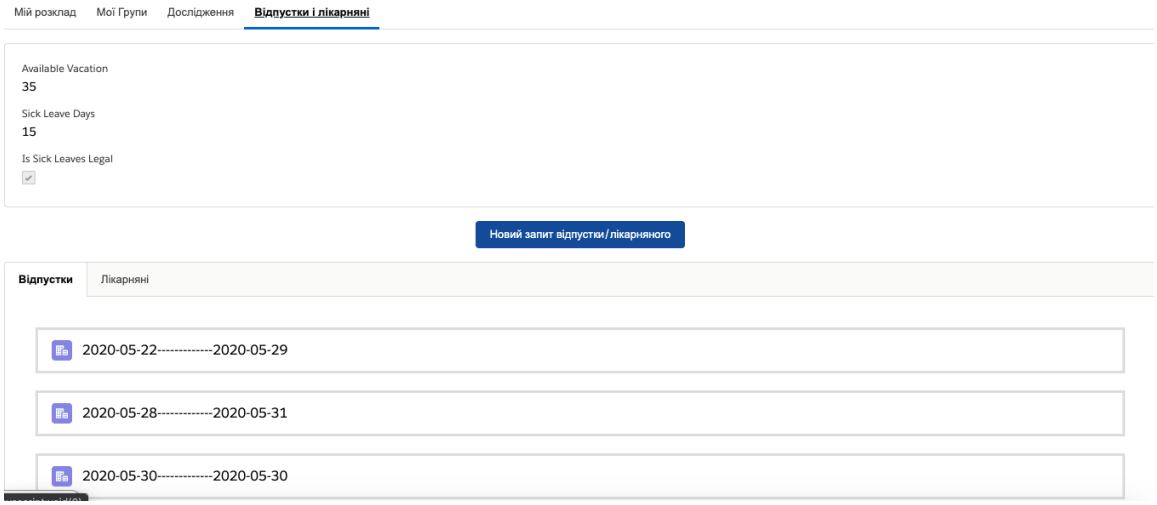

Рисунок 5.30 Вкладка моніторингу відпусток

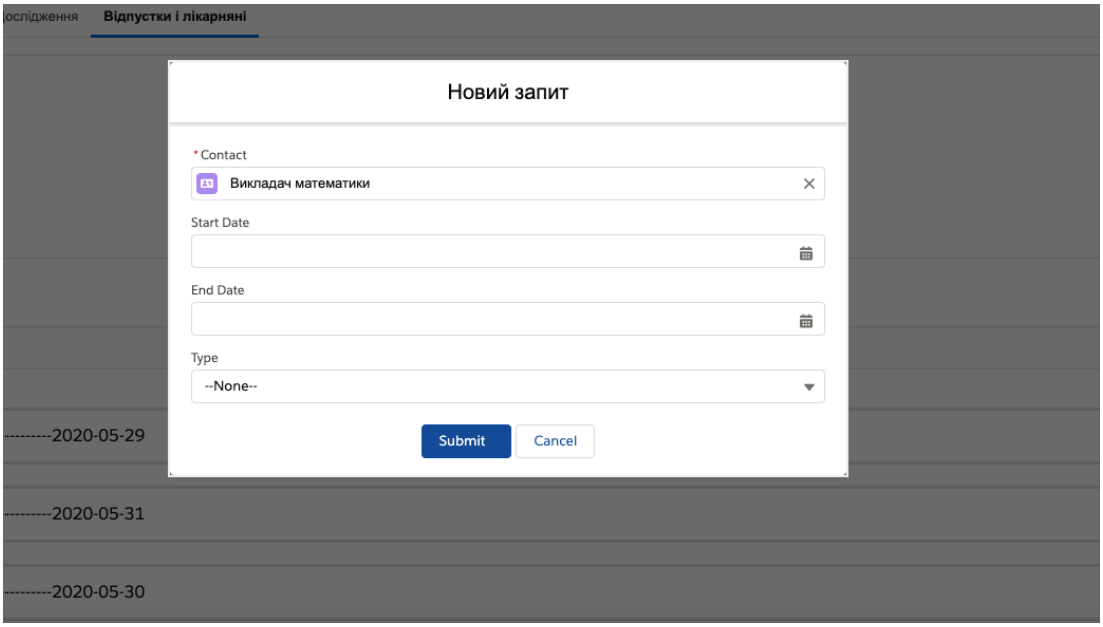

Рисунок 5.31 Вікно створення запиту нової відпустки або лікарняного

Висновки

В даній роботі було досліджено бізнес процес роботи кафедри. Проаналізовано достоінства і недоліки.

У зв’язку з доволі великим обсягом і складності бізнес процесу вибір впав на CRM систему, що дало значущий плюс в швидкості розробки, так як деякий функціонал доступний відразу.

Порівнявши існуючі CRM системи та інші технології розробки вибір впав на Salesforce. Після цього дана робота була зосереджена на дослідженні побудови проектів на вибраній платформі. Перш за все, було описано процес проектування нового проекту, виділення ключових моментів.

Вибрана CRM система мала декілька підходів до розробки, тому було досліджено кожну з них.

Платформа Salesforce надає різноматні методи розробки. Було досліджено мову програмування Apex, для написання серверної логіки проекту. Для написання користувацького інтерфейсу використано Lightning Framework, який надає три технології розробки на вибір - Visualforce, Aura, Lightning Web Component. Вибір впав на Aura, так як Visualforce застарілий, а Lightning Web Component був випущений менше року назад. Aura надає можливість розробки користувацького інтерфесу з використанням найновіших стандартів Javascript.

Окрім написання коду як методу розробки, в Salesforce існує декларативні інструменти розробки, які надають можливість вирішувати тривіальні задачі швидко і ефективно. Коректність роботи декларативних методів розробки гарантується компанією Salesforce не залежно від версії технології, в свою чергу з кожним релізом є можливість того, що написаний код буде працювати не коректно.

В результаті аналізу бізнес процесу університету виведено основні його недоліки і спроектовано весь процес на CRM систему з метою покращення ефективності і зменшенню кількості помилок.

Перелік використаних джерел

1. Heroku [Електронний ресурс] – Режим доступу до ресурсу:https://developer.salesforce.com/product-heroku.
2. Fun way to learning Salesforce [Електронний ресурс] – Режим доступу до ресурсу: https://trailhead.salesforce.com/home.
3. Технічна документація мови JavaScript [Електронний ресурс] / – Режим доступу до ресурсу: https://learn.javascript.ru/
4. ДНК пользователей Salesforce: результаты анализа компаний которые используют Salesforce CRM [Електронний ресурс] / 5 – Режим доступу до ресурсу: https://habr.com/ru/post/244989/.
5. Lightning Component Framework [Електронний ресурс] – Режим доступу до ресурсу:https://developer.salesforce.com/docs/atlas.en-us.lightning.meta/lightning/intro_framework.htm.
6. SQOL [Електронний ресурс] – Режим доступу до ресурсу: https://developer.salesforce.com/docs/atlas.en-us.soql_sosl.meta/soql_sosl/sforce_api_calls_soql_sosl_intro.htm.
7. Salesforce. Apex Developer Guide [Електронний ресурс] / Salesforce – Режим доступу до ресурсу: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/.
8. Автоматизація бізнес процесів з використанням CRM системи Salesforce[Доповідь на конференцію] НОВАК М.С., студент 4 курсу, група ТМ- 62 Науковий керівник – доц., к.в.н. Онисько А.І.
9. Производственный менеджмент: Учебник /Под ред. В. А. Козловского. — М.:Инфра-М., 2003. — 574 с.
10. Эрик Эванс. Предметно-ориентированное проектирование (DDD): структура сложных программных систем — М.: ООО «И.Д. Вильямс», 2011. — 448 с.
11. Роберт С. Мартин. Быстрая разработка программного обеспечения: принципы, примеры, практика. — М.: ООО «И.Д. Вильямс», 2004. — 752 с.
12. Роберт С. Мартин. Чистый код: создание, анализ и рефакторинг. — СПб.: Питер, 2010. — 464 с.
13. Сімоненко В. П., Сімоненко А.В. Метод пошагового конструирования для составлении расписания занятий в учебных заведениях. — К.: Наукові вісті НТУУ «КПІ», 2008 – 85 с.

14. Мартин Фаулер. Архитектура корпоративных программных приложений. — М.: ООО «И.Д. Вильямс», 2006. — 544 с.
15. International Book Series "Information Science and Computing"). Режим доступу: http://www.foibg.com/ibs_isc/ibs-07/IBS-07-p25.pdf
16. *Новак М.С.* Автоматизація бізнес процесів…. // Матеріали XVIII Міжнародної науково-практичної конференції молодих вчених і студентів 2020 року. У 2 т. – К. : 7 КПІ ім. Ігоря Сікорського, 2020. – Т. 2. – 160 с.

ДОДАТОК А

Автоматизація бізнес процесів з використанням CRM системи Salesforce

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62205_20Б-1

Аркушів 1

Київ 2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ ”_ТЕФ_АПЕПС_ ТМ62205_20Б-1	Дипломна робота Новак М.С.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ”КПІ ”_ТЕФ_АПЕПС_ ТМ62205_20Б-2	StudiesTools.cmp MainDepartmentPage.cmp GeneralSchedule.cmp Relationships.cmp StudentInfo.cmp ProfesorInfo.cmp Authorization.cmp FAQ.cmp SelectDepartment.cmp SelectDepartmentController.apxc ProfessorInfoController.apxc StudentInfoController.apx ScheduleController.apx	Основні компоненти
УКР.НТУУ”КПІ ”_ТЕФ_АПЕПС_ ТМ62205_20Б-3	Додаток В	Опис програмного модуля

ДОДАТОК Б

Автоматизація бізнес процесів з використанням CRM системи Salesforce

Код програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТМ62205_20Б-2

Аркушів 12

Київ 2020

```
<aura:component implements="flexipage:availableForAllPageTypes,flexipage:availableForRecordHome" access="global">
    <aura:attribute name="departmentId" type="String" />
```

```
<aura:attribute name="contactId" type="String" />
<aura:attribute name="facultyId" type="String" />
<aura:attribute name="contactType" type="String" />
 <aura:if isTrue="{!v.departmentId}">
    <div class="slds-clearfix">
        <div class="slds-float_right">
          <lightning:button label="Change selected department" title="Back" onclick="{! c.handleBack }"/>
          <lightning:button label="Log out" title="Back" onclick="{! c.handleLogout }"/>
        </div>
   </div>
    <lightning:tabset >
   <lightning:tab label="Головна">
      <c:MainDepartmentPage departmentId="{!v.departmentId}" />
   </lightning:tab>
   <lightning:tab label="Розклад">
      <aura:if isTrue="{!v.departmentId}">
      <c:GeneralSchedule departmentId="{!v.departmentId}"/>
      </aura:if>
   </lightning:tab>
   <lightning:tab label="Зв'язки кафедри">
      <c:Relationships />
   </lightning:tab>
   <lightning:tab label="Персональна інформація">
      <aura:if isTrue="{!v.contactId}">
         <aura:if isTrue="{!v.contactType == 'student'}">
            <c:StudentInfo contactId="{!v.contactId}"/>
            <aura:set attribute="else">
               <c:ProfesorInfo departmentId="{!v.departmentId}" contactId="{!v.contactId}"/>
            </aura:set>
         </aura:if>
         <aura:set attribute="else">
            <c:Authorization departmentId="{!v.departmentId}" contactId="{!v.contactId}" contactType="{!v.contactType}"/>
         </aura:set>
      </aura:if>
   </lightning:tab>
   <lightning:tab label="FAQ">
      <c:FAQ departmentId="{v.departmentId}"/>
```

```
        </lightning:tab>
    </lightning:tabset>
    <aura:set attribute="else">

        <c:selectDepartment recordId="{!v.departmentId}" contactId="{!v.contactId}"/>
    </aura:set>
  </aura:if>



</aura:component>
({
    handleBack : function(component, event, helper) {
        component.set("v.departmentId", '');
        component.set("v.contactId", '');
    },

    handleLogout : function(component, event, helper) {
        component.set("v.contactId", '');
    }
})
<aura:component controller="ProfessorInfoController">
    <aura:attribute name="studentProgress" type="Study_Progres__c[]" />
    <aura:attribute name="profId" type="String" />
    <aura:attribute name="studentId" type="String" />
    <aura:attribute name="subjectId" type="String" />
    <aura:attribute name="columns" type="List" />
    <aura:attribute name="errors" type="Object" default="[]"/>
    <aura:attribute name="draftValues" type="Object" default="[]"/>
    <aura:attribute name="sortDirection" type="String" default="asc" />
    <aura:attribute name="defaultSortDirection" type="String" default="asc" />
    <aura:attribute name="sortedBy" type="String" />
    <aura:attribute name="showCreateModal" type="Boolean" default="false" />
    <aura:handler name="init" value="{! this }" action="{! c.init }"/>

    <lightning:button variant="brand"
                      label="Новий запис"
                      title="Brand action"
```

```
            class="slds-align_absolute-center slds-m-top_medium slds-m-bottom_medium"
            onclick="{! c.handleClick }" />

<lightning:datatable
    keyField="Id"
    columns="{!v.columns }"
    data="{!v.studentProgress}"
    hideCheckBoxColumn="true"
    errors="{! v.errors }"
    draftValues="{! v.draftValues }"
    onsave="{! c.handleSaveEdition }"
    defaultSortDirection="{!v.defaultSortDirection}"
    sortedDirection="{!v.sortDirection}"
    sortedBy="{!v.sortedBy}"
    onsort="{!c.handleSort}">
</lightning:datatable>

<aura:if isTrue="{!v.showCreateModal}">
    <section role="dialog" aria-modal="true" class="slds-modal slds-fade-in-open slds-backdrop">
        <div class="slds-modal__container">
            <header class="slds-modal__header" style="outline: 2px solid #f1eded;">
                <h2 class="slds-text-heading_medium slds-hyphenate">Новий запит</h2>
            </header>
            <div class="slds-modal__content slds-p-around_medium" style="outline: 2px solid #f1eded;">
                <lightning:recordEditForm objectApiName="Study_Progres__c" onsuccess="{!c.handleSuccess}">
                    <lightning:messages/>
                    <lightning:inputField fieldName="Professor__c" value="{!v.profId}"/>
                    <lightning:inputField fieldName="Subject__c" value="{!v.subjectId}"/>
                    <lightning:inputField fieldName="Student__c" value="{!v.studentId}"/>
                    <lightning:inputField fieldName="Type__c"/>
                    <lightning:inputField fieldName="Name"/>
                    <lightning:inputField fieldName="Points__c"/>
                    <lightning:inputField fieldName="Is_Student_Exist__c"/>
                    <lightning:inputField fieldName="Date__c"/>
                    <lightning:inputField fieldName="Description__c"/>
                    <div class="slds-align_absolute-center">
                        <lightning:button
```

```
                        variant="brand"
                        class="slds-m-top_small slds-p-right_large"
                        type="submit"
                        label="Submit"/>
                    <lightning:button
                        class="slds-m-top_small"
                        label="Cancel"
                        onclick="{!c.cancelCreate}"/>
                </div>
            </lightning:recordEditForm>
        </div>
    </div>
</section>
</aura:if>
</aura:component>
<aura:component >
<aura:attribute name="myGrops" type="Group__c[]" />
<aura:attribute name="groupId" type="String" />
<aura:attribute name="profId" type="String" />
<aura:if isTrue="{!not(v.groupId)}">
<div>
<ul class="slds-has-dividers_bottom slds-has-block-links_space">
<aura:iteration items="{!v.myGrops}" var="item">
<li class="slds-item">
<lightning:button variant="base"
label="{!item.Name + '  --  ' + item.Course__c + ' курс'}"
value="{!item.Id}"
class="slds-m-left_large"
onclick="{! c.handleSelectGroup }"/>
</li>
</aura:iteration>
</ul>
</div>
<aura:set attribute="else">
<lightning:button label="Back" class="slds-m-left_large slds-m-bottom_medium" title="Back" onclick="{! c.handleBack
}"/>
<c:GroupStudents groupId="{!v.groupId}" profId="{!v.profId}"/>
</aura:set>
```

```
    </aura:if>


</aura:component>
<aura:component controller="VacationsController">
    <aura:attribute name="contactId" type="String" />
    <aura:attribute name="showCreateModal" type="Boolean" default="false" />
    <aura:attribute name="vac" type="Vacation__c" />
    <aura:attribute name="sick" type="Vacation__c" />


    <aura:handler name="init" value="{! this }" action="{! c.init }"/>


    <lightning:recordViewForm recordId="{!v.contactId}" objectApiName="Contact">
        <div class="slds-box">
            <lightning:outputField fieldName="Avaiable_Vacation__c"/>
            <lightning:outputField fieldName="Sick_Leave_Days__c"/>
            <lightning:outputField fieldName="Is_Sick_Leave_Legal__c"/>
        </div>
    </lightning:recordViewForm>


    <lightning:button variant="brand"
                      label="Новий запит відпустки/лікарняного"
                      title="Brand action"
                      class="slds-align_absolute-center slds-m-top_medium slds-m-bottom_medium"
                      onclick="{! c.handleClick }" />


    <lightning:tabset variant="scoped">
        <lightning:tab label="Відпустки">
            <aura:iteration items="{!v.vac}" var="item">
                <div style="margin:20px; border-style: solid;border-color: gainsboro;">
                <lightning:card variant="Narrow" title="{!item.Start_Date__c + '----------' + item.End_Date__c}"
iconName="standard:account">
                    <p class="slds-p-horizontal_small">
                        {!v.item.Description__c}
                    </p>
                </lightning:card>
                    </div>
            </aura:iteration>
```

```
    </lightning:tab>
    <lightning:tab label="Лікарняні">
        <aura:iteration items="{!v.sick}" var="item">
            <lightning:card variant="Narrow" title="{!item.Start_Date__c + '-' + item.End_Date__c}" iconName="standard:account">
                <p class="slds-p-horizontal_small">
                    {!v.item.Description__c}
                </p>
            </lightning:card>
        </aura:iteration>
    </lightning:tab>
</lightning:tabset>


<aura:if isTrue="{!v.showCreateModal}">
    <section role="dialog" aria-modal="true" class="slds-modal slds-fade-in-open slds-backdrop">
            <div class="slds-modal__container">
                <header class="slds-modal__header" style="outline: 2px solid #f1eded;">
                    <h2 class="slds-text-heading_medium slds-hyphenate">Новий запит</h2>
                </header>
                <div class="slds-modal__content slds-p-around_medium" style="outline: 2px solid #f1eded;">
                    <lightning:recordEditForm objectApiName="Vacation__c" onsuccess="{!c.handleSuccess}">
                        <lightning:messages/>
                        <lightning:inputField fieldName="Contact__c" value="{!v.contactId}"/>
                        <lightning:inputField fieldName="Start_Date__c"/>
                        <lightning:inputField fieldName="End_Date__c"/>
                        <lightning:inputField fieldName="Type__c"/>
                        <div class="slds-align_absolute-center">
                            <lightning:button
                                variant="brand"
                                class="slds-m-top_small slds-p-right_large"
                                type="submit"
                                label="Submit"/>
                            <lightning:button
                                class="slds-m-top_small"
                                label="Cancel"
                                onclick="{!c.cancelCreate}"/>
                        </div>
                    </lightning:recordEditForm>
```

```
                </div>


            </div>
        </section>


    </aura:if>
</aura:component>
<aura:component controller="StudentInfoController">
    <aura:attribute name="studentId" type="String" />
    <aura:attribute name="selectedProfessor" type="String" />
    <aura:attribute name="professors" type="Contact[]" />
    <aura:attribute name="subjects" type="Subject__c" />


    <aura:if isTrue="{!not(v.selectedProfessor)}">
        <div>
            <ul class="slds-has-dividers_bottom slds-has-block-links_space">
                <aura:iteration items="{!v.professors}" var="item">
                    <li class="slds-item">
                        <lightning:button variant="base"
                                          class="slds-m-left_large"
                                label="{!item.Name}"
                                value="{!item.Id}"
                                onclick="{! c.handleSelectProf }"/>
                    </li>
                </aura:iteration>
            </ul>
        </div>
        <aura:set attribute="else">
            <div class="slds-m-left_large">
            <lightning:button label="Back" class="slds-m-bottom_small" title="Back" onclick="{! c.handleBack }"/>
            <lightning:recordViewForm recordId="{!v.selectedProfessor}" objectApiName="Contact">
                <div class="slds-grid">
                    <div class="slds-col slds-size_1-of-2">
                        <lightning:outputField fieldName="Name" />
                        <lightning:outputField fieldName="Birthdate" />
                        <lightning:outputField fieldName="Phone" />
```

```
                </div>
                <div class="slds-col slds-size_1-of-2">
                    <lightning:outputField fieldName="Email" />
                    <lightning:outputField fieldName="AccountId" />
                    <lightning:outputField fieldName="Department__c" />
                </div>
            </div>
        </lightning:recordViewForm>
        <c:StudentSubjects mySubjects="{!v.subjects}" contactId="{!v.studentId}"/>
            </div>
    </aura:set>
</aura:if>
</aura:component>
<aura:component controller="StudentInfoController">
    <aura:attribute name="subjectId" type="String" />
    <aura:attribute name="contactId" type="String" />
    <aura:attribute name="subjetName" type="String" />
    <aura:attribute name="professorName" type="String" />
    <aura:attribute name="progress" type="Study_Progres__c[]" />
    <aura:attribute name="columns" type="List" />
    <aura:attribute name="sortDirection" type="String" default="asc" />
    <aura:attribute name="defaultSortDirection" type="String" default="asc" />
    <aura:attribute name="sortedBy" type="String" />

    <aura:handler name="init" value="{! this }" action="{! c.init }"/>

    <p class="slds-text-align_center slds-text-heading_medium slds-m-bottom_small">{!v.subjetName}</p>
    <p class="slds-text-align_center slds-text-heading_medium slds-m-bottom_small">{!v.professorName}</p>
    <lightning:datatable
        keyField="id"
        columns="{!v.columns }"
        data="{!v.progress}"
        hideCheckBoxColumn="true"
        defaultSortDirection="{!v.defaultSortDirection}"
        sortedDirection="{!v.sortDirection}"
        sortedBy="{!v.sortedBy}"
        onsort="{!c.handleSort}">
```

```
    </lightning:datatable>
</aura:component>
public class StudentInfoController {
    @AuraEnabled
    public static List<Direct_Message__c> getMessages(String personId){
        return [SELECT Id, Message__c, Name, Created_Date__c FROM Direct_Message__c WHERE Contact__r.Id =: personId];
    }

    @AuraEnabled
    public static studentWrapper getContactInfo(String personId){
        Contact personInfo  = [SELECT Id, Name, Group__r.Id,
                               phone, Stipend__c, Department__c
                               FROM Contact
                               WHERE id =: personId LIMIT 1];
        Id groupId = personInfo.Group__r.Id;
        Group__c myGroup = [SELECT Id, Name, Speciality__r.Id, Course__c
                            FROM Group__c
                            WHERE id =: groupId
                           LIMIT 1];
        Id specId = myGroup.Speciality__r.Id;
        Speciality__c mySpec = [SELECT Id, Name, Speciality_Code__c
                                FROM Speciality__c WHERE id =: specId
                                LIMIT 1];
        Study_Program__c specStudyProgram = [SELECT id, Name, Number_Of_Subject__c
                                             FROM Study_Program__c
                                             WHERE Assigned_Speciality__c =: mySpec.Id
                                             LIMIT 1];
        List<Subject__c> mySubj = [SELECT id, Name, Credits__c
                                   FROM Subject__c
                                   WHERE Study_Program__c = : specStudyProgram.Id];
        List<Contact> myProf = [SELECT Id, Name
                                FROM Contact
                                 WHERE id IN (SELECT Professor__c FROM Study_Progres__c WHERE Student__c =: personInfo.Id)];
        return new studentWrapper(personInfo, myGroup, mySpec, myProf,
                                  specStudyProgram, new List<Study_Progres__c>(), mySubj);
    }
```

```
@AuraEnabled
public static List<Study_Progres__c> getProgress(String personId, String subjId){
    return [SELECT id, Name, Is_Student_Exist__c, Points__c, Professor__r.Name,
            Type__c, Date__c, Subject__r.Name
            FROM Study_Progres__c
            WHERE Student__c =: personId AND Subject__c =: subjId];

}

@AuraEnabled
public static List<Subject__c> getProfessorSubj(String profId, String personId){
    return [SELECT id, Name, Credits__c
             FROM Subject__c
             WHERE id IN (SELECT Subject__c
                          FROM Study_Progres__c
                          WHERE Student__c =: personId AND Professor__c =: profId)];
}

public class studentWrapper{
    @AuraEnabled
    public Contact personInfo {get; set;}
    @AuraEnabled
    public Group__c myGroup {get; set;}
    @AuraEnabled
    public Speciality__c mySpeciality {get; set;}
    @AuraEnabled
    public List<Contact> myProfessors {get; set;}
    @AuraEnabled
    public Study_Program__c myStudyProgram {get; set;}
    @AuraEnabled
    public List<Study_Progres__c> myStudyProgress {get; set;}
    @AuraEnabled
    public List<Subject__c> mySubjects {get; set;}

    public studentWrapper(Contact person, Group__c myGr, Speciality__c meSpec, List<Contact> profs,
                          Study_Program__c stProg, List<Study_Progres__c>  progres, List<Subject__c> subj){
```

```
            this.personInfo = person;
            this.myGroup = myGr;
            this.mySpeciality = meSpec;
            this.myProfessors = profs;
            this.myStudyProgram = stProg;
            this.myStudyProgress = progres;
            this.mySubjects = subj;
        }
    }
}
public class ProfessorInfoController {
    @AuraEnabled
    public static List<Group__c> getProfessorGroups(String personId){
        List<Contact> students = [SELECT Id, Group__c
                                  FROM Contact
                                  WHERE id IN (SELECT Student__c FROM Study_Progres__c WHERE Professor__c =: personId)];
        Set<id> groupsId = new Set<Id>();
        for(contact st : students){
            groupsId.add(st.Group__c);
        }
        return [SELECT Id, Name, Course__c FROM Group__c WHERE id IN :groupsId];
    }

    @AuraEnabled
    public static List<Contact> getStudents(String grId){
        return [SELECT id, Name, Group__r.Name FROM Contact WHERE Group__c =: grId];
    }

    @AuraEnabled
    public static List<Subject__c> getGroupSubject(String grId, String profId){
        return [SELECT Id, Name
                FROM Subject__c
                WHERE Id IN (SELECT Subject__c FROM Schedule__c WHERE Group__c =: grId AND Contact__c =: profId )];
    }

    @AuraEnabled
    public static List<Study_Progres__c> getStudentProgress(String personId, String subjId, String profId){
```

```
        System.debug('get proggresss for prjfessors');
        System.debug(personId);
        System.debug(subjId);
        System.Debug(profId);
        System.debug([SELECT id, Name, Is_Student_Exist__c, Points__c,
                Type__c, Date__c
                FROM Study_Progres__c
                WHERE Professor__c =: profId AND Subject__c =: subjId AND Student__c =: personId]);
        return [SELECT id, Name, Is_Student_Exist__c, Points__c,
                Type__c, Date__c
                FROM Study_Progres__c
                WHERE Professor__c =: profId AND Subject__c =: subjId AND Student__c =: personId];
    }

    @AuraEnabled
    public static void saveChange(List<Study_Progres__c> changedValues){
        update changedValues;
    }

} public without sharing class SelectDepartmentController {
    @AuraEnabled
    public static DataWrapper retrieveData(){
        Id facultyRTId = Schema.SObjectType.Account.getRecordTypeInfosByDeveloperName().get('Faculty').getRecordTypeId();
        List<Account> facAccs= [SELECT id, Name, Faculty_Building_Number__c, NumberOfEmployees, Number_Of_Students__c
                    FROM Account WHERE RecordTypeId =: facultyRTId];
        Id departmentRTId =
Schema.SObjectType.Account.getRecordTypeInfosByDeveloperName().get('Department').getRecordTypeId();
        List<Account> depAccs= [SELECT id, Name, Faculty_Building_Number__c, NumberOfEmployees, Number_Of_Students__c
                    FROM Account WHERE RecordTypeId =: departmentRTId];
        List<Speciality__c> spec = [SELECT id, Name, Speciality_Code__c FROM Speciality__c];
        DataWrapper wrap = new DataWrapper(facAccs, depAccs, spec);
        return wrap;
    }

    @AuraEnabled
    public static Account getDepartmentInfo(String depId){
        return [SELECT id, Name, Faculty_Building_Number__c, NumberOfEmployees, Number_Of_Students__c,
```

```
            YearStarted, Description, (SELECT Id, Name, Speciality_Code__c FROM Department__r)
                FROM Account WHERE id =: depId LIMIT 1];
    }

    @AuraEnabled
    public static List<Account> getDepartments(String recId){
        return [SELECT id, Name, Faculty_Building_Number__c, NumberOfEmployees, Number_Of_Students__c
                FROM Account WHERE ParentId =: recId];
    }
    public class DataWrapper {
        @AuraEnabled
        public List<Account> faculties;
        @AuraEnabled
        public List<Account> depatments;
        @AuraEnabled
        public List<Speciality__c> specialities;

        dataWrapper(){
            faculties = new List<Account>();
            depatments = new List<Account>();
            specialities = new List<Speciality__c>();
        }

        dataWrapper(List<Account> fac, List<Account> dep, List<Speciality__c> spec){
            faculties = fac;
            depatments = dep;
            specialities = spec;
        }
    }
}
```

ДОДАТОК В

Автоматизація бізнес процесів з використанням CRM системи Salesforce

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТМ62205_20Б-3

Аркушів

Київ 2020

АНОТАЦІЯ

Додаток містить описання роботи з CRM системою. Основною одиницею програмного продукту є org на платформі Salesforce. Для різних типів користувачів існують різні способи доступу до функціоналу. Описується рекомендації щодо взаємодії з інтерфейсом користувача. Програмна реалізація виконувалась на мовах програмування Apex, Javascript, HTML, CSS, фрейморки Aura, Lightning Design System.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

Кожен тип користувача вимагає різних режимів доступу. Для користувачів типу Адміністрація необхідно створити юзера в середині платформи, після чого їм прийде електронний лист з верифікацією аккаунта, після чого необхідно зайти на сайт login.salesforce.com і ввести свій аккаунт. У разі успішної авторизації користувачу стане доступний інтерфейс управління даними в бд і вбудовані можливості системи. Для користувачів типу Викладач або Студент адміністрації необхідно створити запису об’єкта Контакт, видати логін і пароль, посилання на публічний сайт. Публічний сайт стає доступний після початку хостингу з системи, тому вказувати URL-адресу цього сайту в даній роботі немає сенсу. Після переходу на публічний сайт викладачі і студенти отримують доступ до функціоналу системи.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Призначенням програмного продукту є автоматизація бізнес процесу роботи кафедри. Для цього застосовано технологію CRM системи, яка об’єднує в собі технології бази даних, веб розробки, хмарної технології. Завдяки тому, що Salesforce хмарна CRM система, користувачі мають доступ до актуальних даних в будь-якому місці і в будь-який момент часу. Окрім цього немає необхідності в будь-якому серверному обладнанні. Для доступу до продукту необхідний комп’ютер з браузером. Використання даного підходу до автоматизації бізнес процесу дозволяє зменшити помилки спричинені людським фактором, зменшити витрати на апаратну частину підтримки веб продукту, зменшити тривалість розробки.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Доступ до інформації про особисті дані, прогрес у навчанні, можливість генерування звітів, зменшення паперової роботи, зменшення помилок спричинених людським фактором.

ТЕХНІЧНІ ЗАСОБИ ЩО ВИКОРИСТОВУЮТЬСЯ

Програмний продукт розроблено на платформі Salesforce за допомогою декларативних методів налаштування фукціоналу, який доступний при створенні org та використанням мов програмування Apex, Javascript, HTML, CSS, фрейморки Aura, Lightning Design System. Розробка велась в онлайн середовищі розробки представлене компанією Salesforce і доступне тільки з вашої org – Developer console, а також середовище розробки Visual Studio Code.