Міністерство освіти і науки України
Сумський державний університет
Кафедра комп’ютерних наук
Секція комп’ютеризованих систем управління

ЗАТВЕРДЖУЮ
Зав. кафедри
_____ А. С. Довбиш
__“_____“ 2018 р.

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

за спеціальністю 151"Автоматизація та комп'ютерно-інтегровані технології"

Комп'ютерно-інтегрована система налаштування автоматизованого управління підприємством

Магістрант:
студент гр. СУ.м-71 Вороненко О.С.

Керівник роботи
к.ф.м.н, доцент Журба В.О.

Суми – 2018

Міністерство освіти і науки України

Сумський державний університет

Кафедра комп’ютерних наук

Секція комп’ютеризованих систем управління

ЗАТВЕРДЖУЮ

Зав. кафедри

_____ А. С. Довбиш

__“_____“ 2018 р.

ЗАВДАННЯ

на магістерську роботу студенту

Вороненко Олексію Сергійовичу

1. Тема проекту: Комп'ютерно-інтегрована система налаштування автоматизованого управління підприємством. Затверджено наказом ректора університету від 19 жовтня 2018 р. № 2246 – ІІІ.
2. Термін здавання студентом завершеної роботи "11" грудня 2018р.
3. Вихідні дані до проекту: технічна документація та перелік літературних джерел з матеріалами опису і автоматизації технологічного процесу відповідної установки.
4. Зміст матеріалів роботи: аналіз загальних відомостей про роботу системи налаштування автоматизованого управління підприємством; опис системи автоматизації; розроблення відповідних схем автоматизації; проектування програмного забезпечення та людино-машинного інтерфейсу; моделювання роботи системи та інтеграція системи.
5. Перелік графічних матеріалів: діаграми сутностей системи, схема інформаційно-матеріальних потоків, схема підключення пристрою, функціональна схема пристрою.

6. Календарний план

Номер етапу	Зміст етапу проектування	Термін виконання (початок - кінець)
1	Аналіз проблем. Складання задач. Підбір та аналіз літератури.	18.09.2018 – 20.09.2018
2	Розгляд загальних технологічних питань.	21.09.2018– 1.10.2018
3	Проектування системи налаштування автоматизованого управління підприємством	2.10.2018 – 25.10.2018
4	Реалізація системи налаштування автоматизованого управління підприємством	26.10.2018 – 29.10.2018
5	Проектування інтеграції системи з системою прогнозування	30.10.2018 – 16.11.2018
6	Створення програмного забезпечення та людино-машинного інтерфейсу.	17.11.2018 – 06.12.2018
7	Технічне оформлення роботи. Здача керівнику.	08.12.2018 – 11.12.2018

7. Дата видачі завдання " 08 " вересня 2018 р.

Керівник проекту:
к.ф.м.н, доцент Журба В.О.

До виконання прийняв:
студент-магістрант
групи СУ.м-71 Вороненко О.С.

РЕФЕРАТ

Вороненко Олексій Сергійович. Комп'ютерно-інтегрована система налаштування автоматизованого управління підприємством. Магістерська робота. Сумський Державний Університет, Суми, 2018 р.

Робота містить 84 сторінок, 19 рисунків, 3 діаграми; 4 схеми. При виконанні магістерської роботи було використано 12 літературних джерел.

Робота присвячена дослідженню сучасних методів автоматизації роботи підприємств. Запропоновано рішення щодо автоматизації керування та налаштування роботи підприємства. Результати дослідження, що представлені в магістерській роботі, були подані для участі у всеукраїнському конкурсі студентських наукових робіт у 2018 р.

Об’єктом детального вивчення є процес проектування та створення автоматизованих систем управління та налаштування роботи підприємства. Була розглянута та проаналізована технічна документація по даному об’єкту: загальні відомості; основні робочі показники; структурна складова системи. Були розглянуті аналоги таких системи, їх сильні та слабкі сторони. Відповідно до отриманих результатів була спроектована власна комп'ютерно-інтегрована система налаштування автоматизованого управління підприємством. Також була реалізована робоча модель системи та проведена інтеграція з системою прогнозування температури. Для взаємодії з системою був розроблений людино-машинний інтерфейс для взаємодії з системою та для її налаштування.

Ключові слова: автоматизація підприємства, веб-додаток, монолітна модель, інтеграція систем, схема керування.

ABSTRACT

Voronenko Oleksii Sergiyovich. Computer-integrated system for the configuration of automated control systems enterprise. Master thesis. Sumy State University, Sumy, 2018

The work contains 85 pages, 19 figures, 3 charts; 4 schematics. During the implementation of the master's thesis, 12 literary sources were used.

The work is devoted to the research of modern methods of automation of the enterprises. The decision is made to automate the management and configuration of the company systems. The results of the research, presented in the master's thesis, were sent to the contest of student science research.

The object of detailed study is the process of designing and creating automated control systems and control of the enterprise. Following technical documentation for this object was analyzed: general information; main operating indicators; structural component of the system. Analogues of such systems were reviewed and their strengths and weaknesses were researched. In accordance with the results, a computer-integrated system for setting up automated enterprise management was designed. A working model of the system was also implemented and integrated with the temperature prediction system. To interact with the system and to configure it., a human-machine interface was developed.

Keywords: enterprise automation, web application, monolithic model, system integration, control scheme.

ЗМІСТ

СКОРОЧЕННЯ І УМОВНІ ПОЗНАЧЕННЯ

CRM – Customer Relationship Management

ERD – Entity Relationship Diagram

ERP – Enterprise Resource Planning System

MRP – Material Requirement Planning

ROR – Ruby on Rails

RO – Read Only

RW – Read/Write

SQL – Structured Query Language

БД – База даних

МП – Мова программування

ВСТУП

Планування ресурсів підприємства (Enterprise Resource Planning System) - корпоративна інформаційна система (KIS), призначена для автоматизації бухгалтерського обліку та управління. Як правило, ERP-системи будуються за модульним принципом і певною мірою охоплюють всі основні процеси компанії.

Класичні системи ERP забезпечують управління завданнями: фінансовий менеджмент, планування виробництва та управління, керування запасами та управління розподілом, управління впровадженням та маркетингом, керування утриманням клієнтів, управління постачанням, управління проектами, управління сервісами, процедури управління якістю продукції.

Задачі дипломної роботи:

1. Дослідити та проаналізувати основні принципи створення автоматизованих систем управління та налаштування підприємства

2. На основі прикладної задачі спроектувати систему налаштування автоматизованного управління підприемством.

3. Зробити огляд основних інструментів для реалізації системи.

4. Реалізувати систему на основі вибраних інструментів, оцінити характеристики системи та результати роботи системи базуючись на тестових даних.

5. Зробити висновки щодо отриманих результатів.

Ціль дипломної роботи

Основною ціллю дипломної роботи є створення проекту системи прогнозування налаштування автоматизованного управління підприємством та аналіз роботи спроектованої системи.

Ця інформаційна система, яка підтримує автоматизацію функцій управління на підприємстві (в корпорації) і постачає інформацію для прийняття управлінських рішень. У ній реалізована управлінська ідеологія, яка об'єднує бізнес-стратегію підприємства і прогресивні інформаційні технології. Такі системи набувають різного виду залежно від роду діяльності підприємства та його потреб.

Розділ 1. Розгляд автоматизованих систем управління підприємством

1.1 Актуальність задачі

Відповідно до основних положень теорії систем, будь-який об'єкт, явище або процес (включаючи підприємство) можна розглядати як систему. Під системою розуміють сукупність взаємопов'язаних в одному цілі елементи. Елемент системи є частиною цілого, який в процесі аналізу не підрозділяється на компоненти. Отже, для будь-якої системи характерно, що:

1. він складається з двох або більше елементів;
2. кожен з його елементів має свої якості;
3. існують зв'язки між елементами системи, з якими вони впливають один на одного;
4. система не може існувати за межі часу та простору. Вона має тимчасовий характер (його склад можна визначити в будь-який момент), а також його межі та середовище.

Перша особливість підприємства як системи полягає в тому, що підприємство є відкритою системою, яка може існувати лише тоді, коли вона активно взаємодіє із середовищем. Він вибирає в проміжному та загальному середовищі основні чинники виробництва, перетворюючи їх у продукти (товари, послуги, інформацію) і відходи, переводить їх знову в зовнішнє середовище. Стан життєздатності системи є корисним (сприятливим) обміном між "входом" і "виходом".

Створення автоматизованої системи управління підприємством допомагає вирішити багато проблем:

1. Забезпечити прозорість використання інвестицій у бізнес-капітал;

2. Надати повну інформацію про економічну доцільність стратегічного планування;

3. Професійно керувати витратами, чітко та своєчасно, таким чином мінімізуючи витрати;

4. Реалізувати операційне управління підприємством за обраними ключовими показниками (вартість виробництва, структура витрат, рівень прибутковості тощо);

5. Забезпечити гарантовану рентабельність підприємства шляхом оптимізації та прискорення ряду процесів (термінів виконання нових замовлень, перерозподілу ресурсів тощо).

Повна система управління підприємством повинна забезпечувати інформаційну прозорість підприємства, формувати єдиний Інформаційний простір, який поєднує в собі потік інформації, що надходить від виробництва до неї, з даними фінансових та економічних послуг та видає необхідні комунікації для всіх рівнів управління підприємством.

1.2 Види автоматизованих систем керування підприємством

Корпоративні інформаційні системи поділяються на такі класи:

ERP (система планування ресурсів підприємства)

Сучасні ERP з'явилися в результаті майже 40 років еволюції управління та інформаційних технологій. Вони призначені в основному для побудови єдиного інформаційного простору підприємства (об'єднання всіх підрозділів та функцій), ефективного управління всіма ресурсами компанії, пов'язаними з продажем, виробництвом, обліком замовлень. Система ERP побудована на модульній основі і зазвичай включає модуль захисту, який запобігає як внутрішньому, так і зовнішньому крадіжці інформації.

Проблеми, однак, виникають в основному через помилки в застосуванні або початковому плані впровадження системи. Наприклад, скорочені інвестиції в навчання персоналу для роботи в системі значно знижують ефективність. Тому ERP-системи, як правило, реалізуються не відразу повністю, але в окремих модулях, особливо на початковому етапі.

Рисунок 1. Функціональний склад ERP

CRM (Система управління взаєминами з клієнтами)

Управління взаємовідносинами з клієнтами - це концепція, яка охоплює концепції, що використовуються компаніями для управління їх взаємовідносинами з клієнтами, включаючи збирання, зберігання та аналіз інформації про споживачів, постачальників, партнерів та інформацію про їх відносини з ними. Сучасне CRM має на меті вивчення ринку та специфічні потреби клієнтів. На основі цих знань розробляються нові товари або послуги, завдяки чому компанія досягає своїх цілей та покращує фінансові показники. [3]

Існує три підходи CRM, кожен з яких може бути реалізований окремо від інших.

1. Оперативна - автоматизація споживчих бізнес-процесів, що допомагає персоналу працювати з клієнтами для виконання своїх функцій.
2. Співпраця. Програма взаємодіє з споживачами без участі персоналу, що працює з клієнтами.
3. Аналітичний аналіз інформації про споживачів з різними цілями.

Принципи CRM-систем

1. Наявність єдиного сховища інформації, звідки вся інформація про всі випадки взаємодії з клієнтом доступна в будь-який час;
2. Синхронізація керування кількома каналами взаємодії;
3. Постійний аналіз зібраної інформації про клієнтів та прийняття відповідних організаційних рішень - наприклад, "сортування" клієнтів на підставі їх значимості для компанії.

Особливості CRM-систем:

- Швидкий доступ до найновішої інформації про клієнтів;
- Оперативне обслуговування клієнтів та операції;
- Формалізація схем взаємодії з клієнтами, автоматизація документообігу;
- Швидкий доступ до всієї необхідної звітної та аналітичної інформації; - * Зменшення операційних витрат керівників;
- Контроль роботи керівників;
- Координована взаємодія між співробітниками та підрозділами.
- Управління бізнес-процесами - автоматизує послідовні операції, що виконуються співробітниками організації;
- Управління контактами, історія взаємодії з клієнтами - єдина база даних всіх контрагентів компанії (клієнтів, постачальників, конкурентів) з раніше наданою докладною інформацією про них, їх працівників тощо.

Система дозволяє швидко шукати важливу інформацію про контрагентів, отримати всю історію зустрічей, переговорів, кореспонденції, транзакцій тощо. Це дуже зручний інструмент для швидкої та якісної роботи з величезними масивами інформації про клієнтів. Система автоматично нагадує про необхідність здійснити дзвінок, про заплановані зустрічі та інші події;

CPM дозволяє планувати різні показники (дохід від продажів для менеджерів, відділів, продуктів ...). У історії проектів ви можете відновити послідовність продажів, що дозволяє визначити проблемні області в циклах продажів. Планування та контроль фактичної реалізації плану. Існує можливість проведення різноманітних прайс-листів (оптовий, дрібнотоварний, роздрібний), враховувати рекламні пропозиції, знижки на обсяг покупки.

Вся робота з клієнтом відбувається в одній системі: планування заходів, виконання угод, підготовка та видача необхідних бухгалтерських документів;

- Планування та управління покупками та постачаннями - у системі менеджери завжди можуть бачити наявність та кількість товарів на складі. Відповідальні працівники можуть контролювати виконання плану закупівель;
- Управління маркетингом - електронна пошта, пряма пошта, SMS-розсилка. Система дозволяє управляти маркетинговою діяльністю та визначати її ефективність. Можливість сегментації існуючих клієнтів (активних та потенційних) на основі певних параметрів маркетингових подій;
- Автоматизація документів - шаблони будь-яких документів, що використовуються в організації, можуть бути введені в систему, отже, необхідність вручну складати новий документ зникає, коли виникає подія. Шаблони шаблонів шаблон автоматичного заповнення, які зберігаються в системі. Автоматичне виставлення рахунків та контроль за ними платежів за допомогою сумісності з Клієнт-банком;
- Можливість роботи в мережі;
- Імпорт контрагентів з інших баз даних;

МНС (система виконання виробничих робіт)

Системи класу MES призначені для виробничого середовища підприємства. Системи в цьому класі відслідковують і документують весь виробничий процес, відображаючи виробничий цикл у режимі реального часу. На відміну від ERP, який не має прямого впливу на процес, можна налаштувати (або повністю перебудувати) процес з МОНу стільки разів, скільки потрібно. Іншими словами, системи цього класу призначені для оптимізації виробництва та підвищення його рентабельності. Зібравши та аналізуючи дані, отримані, наприклад, з технологічних ліній, вони дають більш детальне уявлення про виробничу активність підприємства (від складання замовлення до відвантаження готової продукції), покращення фінансових показників підприємство

Усі основні показники, що входять до основного курсу економіки галузі (прибуток від основних фондів, рух грошових коштів, собівартість, прибуток та продуктивність), деталізовані в процесі виробництва. Експерти називають МЕС мостом між фінансовими операціями ERP-систем та оперативною діяльністю підприємства на рівні майстерні, ділянки або лінії.

Рисунок 2. Функціональний склад MES

WMS (система управління складом)

Система управління складом - це система управління, яка забезпечує автоматизацію та оптимізацію всіх процесів складської роботи профільного підприємства.

Архітектура автоматизованої системи управління інформаційним складом базується на трирівневому принципі:

- Перший компонент - видима частина користувача - інтерфейс "людина-машина" - "клієнтське додаток", за допомогою якого користувач вводить, змінює та видаляє дані, запити виконання операцій та запити для відбору даних (отримання звітів).
- другий компонент (прихований від частини користувача системи) - сервер бази даних, який зберігає дані. Користувач через клієнтську програму ініціює процедуру запиту вибірки, вставки, модифікації або видалення даних у базі даних (БД).
- Третій компонент - бізнес-логіка ("завдання" або "процеси" - спеціалізовані програми обробки) здійснює обробку даних, ініційовані користувачем, і повертає

оброблені дані в базу даних, спілкуючись з користувачем через екран клієнтської програми завершення запитувана обробка.

Цілі впровадження:

- Активне управління складом;
- збільшити швидкість набору товарів;
- отримання точної інформації про місцезнаходження товарів на складі;
- ефективне управління товарами з обмеженим терміном зберігання;
- отримання інструменту підвищення ефективності та розвитку процесів обробки товарів на складі;
- Оптимізація використання складських приміщень.

EAM (Enterprise Asset Management)

Система управління основними засобами підприємства, що дозволяє скоротити просте обладнання, витрати на технічне обслуговування, ремонт та матеріально-технічне забезпечення. Це необхідний інструмент у роботі капіталомістких галузей (енергетика, транспорт, житлово-комунальне господарство, добувна промисловість та НД).

Основні засоби - це засоби праці, які неодноразово задіяні у виробничому процесі, зберігаючи свою природну форму, поступово зношуючи її, переносячи його вартість у частини для новостворених виробів. У бухгалтерському та податковому обліку відображаються в грошовому вираженні, основні засоби називаються основними активами. Історично склалося, що системи EAM виникли з CMMS-систем (інший клас IP, ремонт).

В даний час модулі EAM також є частиною великого пакету систем ERP (таких як mySAP Business Suite, IFS Applications, Oracle E-Business Suite тощо).

HRM (управління людськими ресурсами)

Система управління персоналом - це одна з найважливіших компонентів сучасного менеджменту. Основною метою таких систем є залучення та збереження цінних людських ресурсів для компанії. HRM-системи вирішують два основні завдання: оптимізувати всі облікові та розрахункові процеси, пов'язані з персоналом, і зменшити відсоток від'їзду працівників. Таким чином, системи управління персоналом в певному сенсі можна назвати

"CRM-системами навпаки", залучати і зберігати не покупці, а власні працівники компаній. Звичайно, методи, використані тут, зовсім різні, але загальні підходи аналогічні.

Функції HRM:

- пошук персоналу;
- Підбір і підбір персоналу;
- оцінка персоналу;
- навчання та розвиток персоналу;
- Управління корпоративною культурою;
- мотивація персоналу;
- Організація роботи.

1.3 Зарубіжні ERP-системи

Серед найпомітніших програмних продуктів, які впроваджують концепцію ERP, насамперед є mySAP ERP, MySAP All-in-One і SAP Business One SAP AG, Oracle E-Business Suite, JD Edwards і PeopleSoft Enterprise Oracle. Підприємства Microsoft Microsoft Dynamics AX (Axapta) і NAV (Navision) підтримують малий та середній бізнес-сегмент Microsoft на українському ринку. Також впевнено вийшов на український ринок такі рішення ERP, як ALTUM і ALTUM XL, Comarch, а також шведська система ERP та CRM System Enterprise від HansaWorld.

Інші рішення включають системи infor: ERPnext COM, MAX +, SSA ERP LN (Baan) і SyteLine від Infor.

Існують також менш універсальні рішення, які спрямовані на розширення функціональності з конкретними специфічними галузями. Прикладом є прикладна система IFS IFS IFS з розширеною функціональністю для виготовлення та ремонту.

Ряд російських та українських програмних систем також так чи інакше реалізують функціональність зазначеного ERP.

Отже, система 1С: Enterprise Management 8.0 часу, деякі розглядають повнофункціональну ERP-систему. Корпоративна інформаційна система (МКС) "Парус - підприємство 8.5" також є яскравим прикладом російської системи ERP. Ця система використовується як в Росії, так і в Україні.

Інший приклад російських систем ERP: INTALEV: Корпоративний менеджмент, флагман, Фрегат - Корпорація, AVA Systems.

Білоруська ERP та CRM POWER SYSTEM Компанія-розробник – VIPSOFT

Microsoft Dynamics

Microsoft Dynamics — це лінійка продуктів програмного забезпечення для бізнесу від корпорації Microsoft. Спочатку вона існувала під кодовою назвою Project Green. А у вересні 2005 року Microsoft Dynamics замінила старий бренд Microsoft Business Solutions.

До Microsoft Dynamics входять такі програмні продукти:

- Управління відносинами з клієнтами
- Microsoft Dynamics CRM
- Управління ресурсами підприємства
- Microsoft Dynamics AX (попередня назва Axapta)
- Microsoft Dynamics GP (попередня назва Great Plains)
- Microsoft Dynamics NAV (попередня назва Navision)
- Microsoft Dynamics SL (попередня назва Solomon)
- Управління роздрібним підприємством
- Microsoft Retail Management System (попередня назва QuickSell)

В Україні Microsoft Dynamics представлена такими рішеннями: Microsoft Dynamics AX, Microsoft Dynamics NAV та Microsoft Dynamics CRM. Напрямок Microsoft Business Solutions тут було офіційно відкрито у травні 2006 року. Наразі рішення на базі Microsoft Dynamics використовуються у понад 100 українських компаніях.

Бізнес-логіка Microsoft Dynamics CRM

У структурі є три головні структурні одиниці: користувачі, команди та бізнес-підрозділи. На основі структури бізнес-підрозділів визначаються принципи базового, локального, глибокого та глобального доступу. В рамках організаційної структури Microsoft Dynamics CRM немає можливості вбудовувати матричні організації. Окремі користувачі можуть належати до бізнес-підрозділу, але не можуть безпосередньо бути частиною об'єкта організації.

Завдяки принципу мультиарендності в архітектурі Microsoft Dynamics CRM кілька організацій можуть отримати хостинг на сервері Microsoft Dynamics CRM.

Безпека

Microsoft Dynamics CRM реалізує модель безпеки, яка контролює цілісність та збереження даних, а також доступ до даних у команді та спільної роботи. Основними цілями цієї моделі є підтримка моделі ліцензування для користувачів, надання доступу до відповідних рівнів інформації, класифікація типів користувачів за їхньою роллю, диференціювання рівнів доступу та підтримка обміну даними для спільної роботи з ними. Управління безпекою базується на ролі та об'єктах.

База даних

Microsoft Dynamics CRM - це програмний продукт, який працює з метаданими. Ряд метаданих узагальнює деталі з сховища даних, наприклад схеми та звернення до записів. Метадані створюють опис структури даних, в якій програма використовує та відображає їх. У поточній версії використовується новий інтерфейс Application Programming Interface (API), який дозволяє додавати або оновлювати метадані.

Функція конфігурації робочого процесу вбудована в Windows Workflow Foundation та дає змогу моделювати модель, збільшувати продуктивність та інструменти для швидкого побудови робочих циклів.

Розширення бізнес-логіки

Починаючи з Microsoft Dynamics CRM 4.0, система забезпечує механізм налаштування бізнес-логіки. Розробники можуть створювати його не тільки через зміну процесів робочого циклу. Вони також можуть створювати бізнес-логіку, яка інтегрується з Microsoft Dynamics CRM, і буде виконувати дії у відповідь на системні події в певному об'єкті. Розширення бізнес-логіки на цьому етапі не підтримуються в Microsoft Dynamics CRM Online.

Функціональність

Функціональність Microsoft Dynamics CRM розділена на п'ять категорій. Перші три - продаж (продаж), маркетинг (маркетинг) і обслуговування клієнтів (обслуговування клієнтів) - існують як окремі модулі. Категорії звітування та аналітики включаються до кожного модуля в перших трьох категоріях.

Модуль збуту дозволяє автоматизувати написання ринкових пропозицій та обробку замовлень. Програма дозволяє створювати профілі з інформацією про клієнтів, списки зустрічей та переговорів для кожного клієнта, календарі подій та графіки продажів.

Модуль маркетингу дозволяє створювати тематичні групи клієнтів, розробляти маркетингові сценарії та надсилати індивідуальні пропозиції. Користувачі мають інструменти для вимірювання ефективності прийнятих заходів, для обчислення коефіцієнта витрат / прибутку маркетингових кампаній, для створення облікових записів для потенційних клієнтів, ринкових територій та географічних областей.

Модуль обслуговування клієнтів дозволяє аналізувати запитувану інформацію та поширювати відповідний персонал для надання даних клієнтам, підготовки "списку очікування", використання бази даних для відповіді на запити та обліку тимчасових ресурсів.

Oracle E-Business Suite

Oracle E-Business Suite (також відомий як Applications / Apps, EB-Suite / EBS або "E-Biz") складається з набору планування корпоративного ресурсу (ERP), управління відносинами з клієнтами (CRM) та постачання Chain Management (SCM) комп'ютерні програми, розроблені або придбані компанією Oracle. Програмне забезпечення використовує базову технологію керування реляційною базою даних Oracle в Oracle. E-Business Suite містить кілька продуктів, часто відомі коротким скороченням.[3]

Істотні технології, включені в додатки, включають технології бази даних Oracle (двигуни для RDBMS, PL / SQL, Java, .NET, HTML та XML), "технологічний стек" (Oracle Forms Server, Oracle Reports Server, Apache Web Server, Oracle Discoverer, Jinitiator і Sun Java). Корпорація Oracle представляє онлайнову технічну документацію E-Business Suite як eTRM - "Технічні підручники з E-Business Suite".

Це робить наступні корпоративні програми доступними в рамках Oracle eBusiness Suite:

- Управління життєвим циклом активів
- Відстеження майна
- Управління нерухомістю
- Управління взаєминами з клієнтами

- Планування ресурсів підприємства
- Фінансовий менеджмент
- Управління людським капіталом
- Управління портфоліо проектів
- Закупівля
- Oracle Advanced Procurement
- Oracle Sourcing
- Управління життєвим циклом продукту
- Управління ланцюгами постачання
- Планування поставок
- Логістика та управління транспортом
- Управління замовленням
- Управління цінами
- Виробництво
- Дискретне виробництво
- Процес виготовлення

Oracle E-Business Suite надає безліч фінансових програм, що використовуються на міжнародному рівні у бізнесі. Корпорація Oracle поділяє ці додатки на "комплекти", які вони визначають як набори спільних інтегрованих програм, призначених для виконання певних бізнес-процесів.

Oracle Financials відноситься до тісно пов'язаних фінансових модулів, таких як:

- Oracle Assets
- Oracle General Ledger
- Oracle Payables
- Oracle Дебіторська заборгованість
- Oracle Cash Management
- Oracle Tabs

Ключовими бізнес-процесами, включеними фінансовими програмами, є:

- Процес поточного бізнес-процесу із закупівлею, що включає такі заходи, як закупівля, придбання, здійснення оплати Постачальникам та наступний облік.

- Потік бізнес-процесів на замовлення до готівки включає в себе такі заходи, як Замовлення клієнтів, виконання замовлення, отримання платежів від Клієнтів та наступний облік.

У 2008 році компанія Oracle запустила набір додатків для середніх компаній, які отримали назву Oracle Accelerate. Прискорення забезпечує доступ до продуктів ERP Oracle через локальну партнерську мережу та упаковує продукти для задоволення вертикальних галузевих вимог.

1.4 Українські ERP-системи

Дебет Плюс

"Дебет Плюс" - українська міжплатформенна автоматизована система управління підприємством (ASUP) та програмна платформа, написана на вершині Eclipse RCP. Має багаті можливості конфігурації. Ви можете використовувати вбудований Apache Derby, Open Source SQL Server для бази даних PostgreSQL і MySQL, або для платної корпоративної версії комерційних Oracle, Microsoft SQL Server або DB2 як СУБД.

Готові конфігурації спрямовані на облік відповідно до законодавства України, але продовжується робота з адаптації системи до роботи в Російській Федерації.

Безкоштовна версія системи не має жодних обмежень на час роботи, обсязі баз даних, можливостях конфігурації та дозволяє повністю працювати в мережі.

На заключному етапі тестування розробляється повний веб-інтерфейс системи (робота через браузер).

Підтримує механізми оновлення програми, аналогічні аналогічним механізмам для платформи 1С: Підприємство (оновлення програми можна самостійно без допомоги).

Доступні пакети інтеграції з клієнтсько-банківськими системами українських банків та Російського ощадного банку Росії.

Основна архітектура системи

Вся інформація в системі зберігається у вигляді первинних документів, публікацій, каталогів та початкових залишків. Система "Дебет Плюс" заснована на модульному принципі: для бухгалтерського обліку, управління, банківського, касового обліку, оплати праці та обліку персоналу використовуються модулі (підсистеми), що автоматизують

ведення відповідного розділу бухгалтерського обліку. Стандартна базова конфігурація "Debit Plus v12" включає 11 підсистем:

• Врегулювання балансу;

• Облік банківських операцій;

• Облік готівкових операцій;

• Облік інвентарю (склад, торгівля, виробництво);

• Облік основних засобів;

• облік заробітної плати;

• облік персоналу;

• Управлінський облік;

• CRM (взаємодія з клієнтами);

• Автотранспорт (реєстрація транспортних засобів, витрати на ПММ, маршрути проїзду);

• Адміністратор комплексу.

Можливе автономне використання окремої підсистеми, а сама система працює в цілому: поведінка, проведена в одній підсистемі, як видно з решти, не вимагає додаткових операцій для передачі інформації. Податковий облік як підсистема не виділяється, журнали податкового обліку знаходяться в підсистемі "Облік ТМК".

Налаштування системи

Дебет Плюс підтримує формати JasperReports, Open Office та MS Office, можна створювати або редагувати вбудованим Debian Plus iReport або через відповідний офісний пакет.

Крім того, користувач системи може повністю налаштовувати схему облікових записів (шляхом редагування посібника "Розклад акаунту").

Кожна папка (тип) документів відповідає діловому транзакції - сукупності пакетів, що формуються для таких документів. Набір завдань для кожної бізнес-операції може бути встановлений у візуальному режимі, але не змінювати вихідний код бізнес-логіки.

Розширення можна додати до каталогів, папок і типів документів - додаткові поля користувача (ці поля можна заповнити вручну або автоматично програмою).

Для формування вихідних даних для податкового обліку використовуються акти верифікації з постачальниками та покупцями тощо. Користувач може, при необхідності, створювати нові журнали або редагувати існуючі.

Ви також можете налаштувати розрахунки - формули для обчислення відпускних і облікових цін, з урахуванням специфіки обліку на підприємстві, розрахунку знижок, розрахунку додаткових полів у розширеннях і т. Д., А також схеми пов'язаних документів, що дозволяють щоб автоматично створювати інший документ на основі одного документа (наприклад, з перевірки можна автоматично створити коносамент).

Для більш глибокої конфігурації системи використовується мова Java Script (використовується Mhzilla Foundation, створений двигуном Rhino). Інтерфейс користувача описаний у форматі XML (інструменти візуального редагування екранних форм наразі недоступні).

Особливості ліцензування

Хоча вихідний код платформи Debit Plus закритий, і він поширюється за власною ліцензією (забороняється розбирати та декомпілювати ядро системи та змінювати вихідний код ядра), у випадку бізнес-логіки це легко помітити ряд функцій відкритої ліцензії. JavaScript та XML-коди бізнес-логіки повністю відкриті, а ліцензійна угода Free Debit Plus дозволяє продавати похідні продукти комерційно або безкоштовно, за умови, що логотип Debit Plus зберігається, і помітне повідомлення про те, що зміна конфігурації та збереження в Похідна текстова ліцензія на продукт.[5]

Універсал ERP

Програмний комплекс Universal (синонім - PC Universal) - це програмний продукт класу ERP, розроблений українською компанією SoftPro. Поширюється на українському ринку з 1992 року. Основна мета - система управління ресурсами та автоматизація бізнес-процесів для середніх підприємств.

Перші версії програми (версії 1 та 2) з'явилися в 1992-1994 роках і були призначені для вирішення бухгалтерських проблем. Операція була проведена на декількох заводах харчової промисловості Києва, Харкова, Дніпропетровська.

У 1995 році була випущена перша дублікатна версія - "Універсальна" 3, яка дозволяє автоматизувати бухгалтерський та матеріальний облік, а також розрахунок заробітної плати для малих та середніх підприємств. Продукт має модульну структуру, клієнтські додатки працюють під операційною системою DOS. Доступ до бази даних здійснювався в архітектурі файл-сервера.

Спроби перевести текстовий інтерфейс у графічне програмне забезпечення (GUI) цих років не вдалося. Як результат, розвиток "Універсалу" 4, який тривав близько року, було зупинено. Іншою проблемою в той час була очевидна невідповідність між можливостями файл-серверної архітектури системи накопичених обсягів даних.

Результати роботи над вирішенням цих двох завдань (переклад в GUI-інтерфейс, переклад на архітектуру клієнта / сервера) призвели до виходу нової версії - Universal 5. По-перше, продукт, що працює на платформі Advantage Database Server був випущений (версія 5.0 була випущена в 1998 році). Тоді (восени 1999 року) була випущена версія 5.1, в якій були клієнтські програми, що працюють під операційною системою Windows і мали свій стандартний GUI.

Цікавий той факт, що протягом деякого часу (близько 2 років) в рамках деяких систем існували програми DOS та Windows, оскільки перші отримали заслужену популярність завдяки своїм швидкодіючим характеристикам і невибагливості до комп'ютерних технологій.

Версія Universal 5+ припинила технологічний розвиток в наш час, але продовжує використовуватися на малих та середніх підприємствах за підтримки розробника з точки зору дотримання чинного законодавства України.

З метою створення нової версії продукту, який відповідає вимогам великих підприємств, SoftPro почав розробку Versailles версії 6 у 2002 році. Її головна ідея полягала у тому, щоб забезпечити крос-платформну міграцію частини сервера між різними клієнтськими / серверними платформами, присутніми на ринку Однак, як і версія 4, розробка версії 6 була скорочена після року роботи. Однією з причин була неможливість передачі додатків, створених технологією ISAM [EN] - доступ до даних до технології SQL Server. Подальший розвиток продукту призвело до випуску в 2008 році "Універсальний" -7 ". Його основна ідея полягала в тому, щоб перенести базу даних в "рідну" архітектуру Advantage Database Server,

що дозволило зняти обмеження на обсяг збереженої та обробленої інформації, а також збільшити загальну продуктивність системи.

Крім того, розробка отримала напрямок обробки багатопоточних даних, що використовується на сервері проміжного посилання "Універсал-комунікатор", який є частиною системи.

Контури Universal 7: ERP

• Постачання (MRP)

• Управління контрактами на поставку (облік договорів, контроль цін та виконання умов доставки)

• Розрахунок потреби товарів (у торгівлі), сировини та матеріалів (у виробництві)

• вибір постачальників (необов'язково)

• Формування заявок постачальникам

• Відповідність моніторингу

• Управління виробництвом (MRP II)

• Підготовка продукції (розробка технологічних маршрутів, оперативна оцінка часу та матеріалів, перевірка експлуатації)

• Надання продукції (матеріалів, обладнання, робочої сили)

• План виробництва (планування виробництва, планування оптимального завантаження виробничих потужностей, формування виробничих завдань, заповнення виробничих звітів (у тому числі від датчиків виробничого обладнання)

• Написання сировини та матеріалів для випуску ГП

• Складський облік

• Облік прийому ДП, ТМК

• Облік зберігання

• Облік внутрішніх переміщень

• Бухгалтерський облік для відвантаження

• Управління продажами

• Управління торговими контрактами (облік договорів, контроль за строками та умовами угод та цінами покупця при відвантаженні, розрахунок бонусів споживачів)

• Облік заявок покупців. Реєстрація заяв на придбання, автоматичне або ручне підтвердження заявки на відвантаження, бронювання та бронювання товару.

• Бухгалтерський облік для відвантаження. Формування черг рахунків-фактур на основі запитів як в розділі маршруту, так і вручну. Зв'язок з програмами транспортної логістики. Друкувати пакети документів за маршрутом.

• Облік торгового обладнання. Розрахунок іпотечної вартості, стану, комплектації, ремонт тощо. Аналіз збуту, перерозподіл ПО між торговими площами.

• Ціноутворення. Проведення системи прайс-листів, управління запасами.

• Облік (адаптація до законодавства України)

• Облік банківських і касових операцій

• Облік ТМК, МШП,

• Бухгалтерський облік оптової та роздрібної торгівлі

• Виробничі записи

• Облік наданих та отриманих послуг

• Облік імпортних та експортних операцій

• Облік основних засобів, MNMA

• Облік роботи автомобілів та автотранспортних засобів

• заробітна плата та персонал.

• Податковий облік

• Фінансовий менеджмент

• Бюджетування

• Платіжний календар

• Аналіз діяльності

• Інструменти аналізу (Business Intelligence), OLAP, ABC / XYZ Analysis,

• Розрахунок вартості

• Розрахунок балансу підприємства

• Документація

• Управління стандартними бізнес-процесами (на основі шаблонів). Редагування шаблонів бізнес-процесів

• Управління неформалізованими бізнес-процесами

• Адміністрація

• Проектування системних ресурсів

• Управління доступом

Архітектура системи

Universal ERP версії 7 - це система клієнт-сервер, що працює на сервері Advantage Database Server версії 9.1 та вище. Він може бути використаний сервером проміжного посилання "Універсальний комунікатор", який буде виконувати запити до бази даних для читання та запису з виконанням ділових правил системи. Запити можуть генеруватися сторонніми програмами або серверами системи суміжних додатків "Universal: Web Server".

Рисунок 3. Структура продукту Универсал 7

1.5 Висновки

У даному розділі розглянуто та досліджено задачу побудови систем налаштування та управління підприємством. У ході дослідження було розглянуто типи реалізацій таких систем, задачі які вони вирішують та інструменти для їх реалізації.

Враховуючи предмет дослідження та специфіку даного завдання, було сформульовано постановку задачі, де описано основні проблеми, які треба вирішити для створення системи за допомогою якої можна дослідити практичну ефективність систем налаштування та управління підприємством.

У ході дослідження було розглянуто конкретні системи створенні в Україні, а також системи створенні іноземнними компаніями.

Розділ 2. Аналіз існуючих систем автоматизованого контролю та налаштування систем планування ресурсів підприємства

2.1 Розгляд проблематики для використання ERP-системи

Планування ресурсів підприємства (ERP) - інтегроване управління основними бізнес-процесами, часто в режимі реального часу та опосередковане програмним забезпеченням та технологією.

ERP зазвичай називають категорією програмного забезпечення для управління бізнесом, як правило, комплектом інтегрованих програм, які організація може використовувати для збирання, зберігання, управління та інтерпретації даних з цих численних бізнес-операцій.

ERP забезпечує інтегрований та постійно оновлюваний огляд основних бізнес-процесів з використанням спільних баз даних, що підтримуються системою керування базами даних. Системи ERP відстежують бізнес-ресурси - готівку, сировину, виробничі потужності та стан ділових зобов'язань: замовлення, замовлення на закупівлю та фонд заробітної плати. Додатки, що складають систему, діляться даними між різними підрозділами (виробництвом, придбанням, продажем, обліком тощо), які надають дані. ERP полегшує обмін інформацією між усіма діловими функціями та керує зв'язками із зовнішніми зацікавленими сторонами.

Програмне забезпечення системи підприємства - це багатомільярдна індустрія, яка виробляє компоненти, що підтримують різні бізнес-функції. IT-інвестиції стали найбільшою категорією капітальних видатків в американських компаніях за останні десятиліття. Незважаючи на те, що ранні ERP-системи зосереджені на великих підприємствах, підприємства меншого розміру все частіше використовують системи ERP.

Система ERP об'єднує різноманітні організаційні системи та полегшує безпомилкові транзакції та виробництво, підвищуючи тим самим ефективність організації. Однак розробка системи ERP відрізняється від традиційного розвитку системи. ERP-системи працюють на різних комп'ютерних апаратних і мережевих конфігураціях, зазвичай використовуючи базу даних як сховище інформації.

Походження

Група Gartner вперше використовувала абревіатуру ERP в 1990-х роках, щоб розширити можливості планування матеріальних вимог (MRP) та подальшого планування виробничих ресурсів (MRP II), а також комп'ютерне інтегроване виробництво. Не замінивши ці терміни, ERP прийшов представляти велике ціле, що відображало еволюцію інтеграції додатків за межі виробництва.

Не всі пакети ERP розроблені з ядра виготовлення; ERP-постачальники по-різному почали збирати свої пакети з фінансово-бухгалтерськими, технічними та людськими ресурсами. До середини 1990-х років системи ERP вирішували всі основні функції підприємства. Уряди та неприбуткові організації також почали використовувати системи ERP.

Розширення

ERP-системи швидко розвивалися в 90-х роках. Через проблему 2000 року та введення євро, що порушили застарілі системи, багато компаній скористалися можливістю замінити їх старі системи на ERP.

ERP-системи спочатку були спрямовані на автоматизацію функцій заднього офісу, які безпосередньо не впливають на клієнтів та громадськість. Передові офісні функції, такі як управління взаємовідносинами з клієнтами (CRM), безпосередньо стосуються клієнтів або електронних бізнес-систем, таких як електронна комерція, електронне урядування, електронне телебачення та управління електронними фінансами або відносинами з постачальниками (SRM). інтегрований пізніше, коли Інтернет спростив спілкуватися з зовнішніми сторонами ."ERP II" була висунута у 2000 році у статті "Публікації Gartner", яка називається "ERP Is Dead-Long Live ERP II". Він описує веб-програмне забезпечення, яке забезпечує доступ до ERP в режимі реального часу співробітникам та партнерам (таким як постачальники та клієнти). Роль ERP II розширює традиційну оптимізацію ресурсів ERP та обробку транзакцій. Замість того, щоб просто керувати покупками, продажами тощо, ERP II використовує інформацію в ресурсах під його керівництвом, щоб допомогти підприємству співпрацювати з іншими підприємствами. ERP II є більш гнучким, ніж ERP першого покоління. Замість того, щоб обмежувати можливості системи ERP в організації, він виходить за рамки корпоративних стін для взаємодії з іншими системами. Комплект додатків для підприємств - альтернативне ім'я для таких систем. Системи ERP II, як правило,

використовуються для забезпечення спільних ініціатив, таких як управління ланцюгами постачання (СКМ), управління відносинами з клієнтами (CRM) та бізнес-розвідки (БІ) серед ділових партнерських організацій за допомогою різних технологій електронного бізнесу.

Розробники тепер роблять більше зусиль, щоб інтегрувати мобільні пристрої з системою ERP. Постачальники ERP розширюють ERP на ці пристрої поряд з іншими бізнес-додатками. Технічні ставки сучасної ERP стосуються інтеграції устаткування, програм, мереж, ланцюгів постачання. ERP тепер охоплює більше функцій і ролей, включаючи прийняття рішень, відносини між зацікавленими сторонами, стандартизацію, прозорість, глобалізацію та ін.

Системи ERP зазвичай включають такі характеристики:

- Інтегрована система
- Працює в режимі реального часу (або близько)
- Спільна база даних, яка підтримує всі програми
- Послідовний вигляд і відчуття через модулі
- Встановлення системи з детальною інтеграцією додатків / даних за допомогою відділу інформаційних технологій (IT), якщо реалізація не виконується на невеликих етапах
- Параметри розгортання включають: локальну, обладнану або SaaS

Функціональні області

- Система ERP охоплює наступні загальні функціональні області. У багатьох ERP-системах вони називаються та груповуються разом як модулі ERP:
- Фінанси та бухгалтерський облік: головний бухгалтер, основні засоби, кредиторська заборгованість, включаючи ваучер, узгодження та оплата, дебіторська заборгованість Управління грошовими коштами та збори, управління грошовими коштами, фінансова консолідація
- Управлінський облік: бюджетування, калькуляція, управління витратами, витрати на основі діяльності
- Людські ресурси: рекрутинг, навчання, облік робочих місць, фонд заробітної плати, пільги, пенсійні та пенсійні плани, управління різноманіттям, випуск на пенсію, відокремлення

- Виробництво: інжиніринг, інкасація матеріалів, замовлення на роботу, планування, потужність, управління робочими процесами, контроль якості, виробничий процес, виробничі проекти, виробничий потік, управління життєвим циклом продукту
- Обробка замовлень: замовлення на готівку, замовлення, перевірка кредиту, ціноутворення, доступна обіцянка, інвентаризація, доставка, аналіз та звітність збуту, введення в експлуатацію.
- Управління ланцюжком постачання: планування поставок, планування постачальників, конфігуратор продукції, замовлення на готівку, придбання, інвентаризація, обробка претензій, складування (прийом, переміщення, збір та упаковка).
- Керування проектами: планування проекту, планування ресурсів, вартість проекту, структура розбиття на роботі, білінг, час і витрати, одиниці ефективності, управління діяльністю
- Управління взаємовідносинами з клієнтами: продаж та маркетинг, комісійні, сервіс, контакт з клієнтом, підтримка центрів обробки викликів - CRM системи не завжди розглядаються як частина ERP-систем, а системи бізнес-підтримки (BSS).
- Послуги передачі даних: різні "самообслуговування" інтерфейси для клієнтів, постачальників та / або працівників

Кращі практики

Більшість систем ERP включають найкращі практики. Це означає, що програмне забезпечення відображає тлумачення постачальника найбільш ефективного способу виконання кожного бізнес-процесу. Системи різняться в тому, наскільки зручно клієнт може змінити ці методи. Крім того, передові практики зменшили ризик на 71% порівняно з іншими реалізаціями програмного забезпечення.

Використання найкращих практик полегшує дотримання таких вимог, як МСФЗ, Сарбейнса-Оксі або Базель II. Вони також можуть допомогти виконати де-факто галузеві стандарти, такі як електронний переказ коштів. Це пояснюється тим, що процедура може бути легко кодифікована в рамках програмного забезпечення ERP та реплікована з упевненістю у кількох компаніях, які поділяють цю бізнес-вимогу.

ERP-системи підключаються до даних у реальному часі та даних транзакцій різними способами. Ці системи, як правило, налаштовуються системними інтеграторами, які приносять унікальні знання про процес, обладнання та рішення постачальників.

Прямі інтеграції ERP-системи мають підключення (комунікації до обладнання на поверховому майданчику) як частину пропонованого продукту. Це вимагає, щоб постачальники пропонували конкретну підтримку обладнанням на підлозі для своїх клієнтів. Постачальники ERP повинні бути експертами у власних продуктах та з'єднанням з іншими продуктами постачальників, включаючи їхні конкуренти.

Інтеграція з базами даних - системи ERP підключаються до джерел даних джерел на базі станцій через розташування таблиць у базі даних. Системи рослинних поверхонь вкладають необхідну інформацію в базу даних. Система ERP зчитує інформацію в таблиці. Перевагою стадії є те, що постачальникам ERP не потрібно усвідомлювати складність інтеграції обладнання. Підключення стає обов'язком системного інтегратора.

Модулі транзакції Enterprise Appliance (EATM) - ці пристрої безпосередньо зв'язуються з обладнанням обладнання на поверхні та системою ERP за допомогою методів, підтримуваних системою ERP. EATM може використовувати проміжну таблицю, веб-служби або системні інтерфейси програми (API). EATM пропонує вигоду від рішення, що не потребує вирішення.

Інтеграційні рішення. Багато системних інтеграторів пропонують індивідуальні рішення. Ці системи, як правило, мають найвищий рівень початкової вартості інтеграції та можуть мати більш високі витрати на технічне обслуговування та надійність у довгостроковій перспективі. Довгострокові витрати можна звести до мінімуму завдяки ретельному тестуванню системи та детальній документації. Зазвичай інтегровані рішення зазвичай працюють на робочих станціях або комп'ютерах на сервері.

2.2 Кроки реалізації систем керування підприємством

Обсяг ERP зазвичай передбачає суттєві зміни в процесах і практиці роботи персоналу. Загалом, доступні три типи послуг, які допоможуть здійснити такі зміни: консультування, налаштування та підтримка. Час впровадження залежить від розміру

бізнесу, кількості модулів, налаштування, масштабу змін процесу та готовності замовника взяти на себе відповідальність за проект. Модульні ERP-системи можуть бути реалізовані поетапно. Типовий проект для великого підприємства займає близько 14 місяців і вимагає близько 150 консультантів. Малі проекти можуть вимагати місяців; багатонаціональні та інші великі реалізації можуть зайняти роки. Налаштування може значно збільшити час виконання.

Крім того, обробка інформації впливає на різні бізнес-функції, наприклад, деякі великі корпорації, такі як Wal-Mart, користуються системою інвентаризації в часі. Це зменшує запаси запасів та підвищує ефективність доставки, а також вимагає наявності найновіших даних. До 2014 року Walmart використовувала систему під назвою Inforem, розроблену IBM для управління поповненням.

Процес підготовки

Реалізація ERP зазвичай потребує змін у існуючих бізнес-процесах . Погане розуміння необхідних змін процесу до початку реалізації є основною причиною невдалої роботи проекту. Труднощі можуть бути пов'язані з системою, бізнес-процесом, інфраструктурою, навчанням або відсутністю мотивації.

Тому важливо, щоб організації ретельно аналізували бізнес-процеси, перш ніж вони впроваджували програмне забезпечення ERP. Аналіз може визначити можливості для модернізації процесу. Це також дозволяє оцінити узгодження поточних процесів з тими, що забезпечуються системою ERP. Дослідження показують, що ризик невідповідності бізнес-процесів зменшується на:

- Поєднання поточних процесів із стратегією організації
- Аналізуючи ефективність кожного процесу
- Розуміння наявних автоматизованих рішень

Реалізація ERP значно складніше (і політично) у децентралізованих організаціях, оскільки вони часто мають різні процеси, бізнес-правила, семантику даних, ієрархії авторизації та центри прийняття рішень. Це може вимагати перенесення деяких бізнес-підрозділів перед іншими, затримуючи реалізацію, щоб працювати через необхідні зміни для кожної одиниці, можливо, зменшивши інтеграцію (наприклад, зв'язок за допомогою управління основними даними) або налаштування системи для задоволення конкретних потреб.

Потенційним недоліком є те, що прийняття "стандартних" процесів може призвести до втрати конкурентних переваг. Хоча це сталося, втрати в одній сфері часто компенсуються перевагами в інших сферах, збільшуючи загальну конкурентну перевагу.

Конфігурація

Налаштування системи ERP полягає у більшою мірою у збалансуванні того, як організація хоче, щоб система працювала з тим, як вона була розроблена для роботи. Системи ERP зазвичай містять багато налаштувань, які змінюють операції системи. Наприклад, організація може вибрати тип обліку запасів - FIFO або LIFO-для використання; чи визнавати доходи за географічними одиницями, товарними лініями чи каналами розповсюдження; і чи потрібно оплатити витрати на доставку на прибуток клієнтів.

Планування ресурсів двох рівнів підприємства

Дворівневе ERP-програмне та апаратне забезпечення дозволяє підприємствам одночасно виконувати еквівалент двох систем ERP: один на корпоративному рівні, а один - на підрозділі або дочірньому рівні. Наприклад, компанія-виробник може використовувати систему ERP для управління всією організацією, використовуючи незалежні глобальні або регіональні розподільні, виробничі або торговельні центри, а також постачальники послуг для підтримки основних клієнтів компанії. Кожен незалежний центр або дочірня компанія може мати свої бізнес-моделі, робочі процеси та бізнес-процеси.

З огляду на реалії глобалізації, підприємства постійно оцінюють, як оптимізувати свої регіональні, дивізійні та виробничі стратегії, щоб підтримувати стратегічні цілі та скоротити час до ринку, одночасно підвищуючи прибутковість та доставку вартості. З дворівневим ERP регіональні розподільні, виробничі або торговельні центри та постачальники послуг продовжують працювати за власною бізнес-моделлю - окремо від основної компанії, використовуючи власні ERP-системи. Оскільки процеси і робочі процеси цих невеликих компаній не прив'язані до основних процесів та робочих процесів компанії, вони можуть відповідати вимогам місцевого бізнесу в різних місцях.

Фактори, що впливають на прийняття підприємств дворівневими системами ERP, включають:

- Виробнича глобалізація, економіка джерел у країнах з перехідною економікою

- Потенціал для більш швидких, менш дорогих реалізацій ERP на дочірні компанії, заснований на виборі програмного забезпечення, більш відповідного для невеликих компаній
- Додаткові зусилля (часто пов'язані з використанням інтеграції корпоративних додатків) потрібні, коли дані повинні проходити між двома системами ERP Дворівневі стратегії ERP дають змогу швидко реагувати на потреби ринку та пристосовувати IT-системи до корпоративних в той час як неминуче призводить до більшої кількості систем порівняно з однією системою ERP, що використовується в організації.

Налаштування

Системи ERP теоретично базуються на передових практиках галузі, і їх виробники мають намір, щоб організації їх розгортали "як є". Постачальники ERP пропонують параметри конфігурації клієнта, які дозволяють організаціям застосовувати власні правила бізнесу, але прогалини функцій часто залишаються навіть після закінчення конфігурації.

Клієнти ERP мають кілька варіантів, щоб узгодити прогалини функцій, кожна з яких має власні плюси / мінуси. Технічні рішення включають переписування частини поставляємого програмного забезпечення, написання власного модуля для роботи в системі ERP або взаємодія з зовнішньою системою. Ці три варіанти складають різну ступінь пристосування до системи - перша є найбільш інвазивною та дорогою для підтримки. Крім того, існують нетехнічні параметри, такі як зміна ділової практики або організаційної політики, щоб краще відповідати наданому набором функцій ERP. Основні відмінності між настройкою та конфігурацією включають:

- Налаштування завжди є необов'язковим, тоді як програмне забезпечення завжди має бути налаштовано перед використанням (наприклад, налаштування структур центру витрат / прибутку, організаційних дерев, правил затвердження покупки тощо).
- Програмне забезпечення розроблено для обробки різноманітних конфігурацій і веде себе передбачувано в будь-якій дозволеній конфігурації.
- Ефект зміни конфігурації на поведінку системи та продуктивність є передбачуваним і є відповідальним за постачальника ERP. Ефект налаштування менш передбачуваний. Це є відповідальністю замовника та підвищує тестування.

Зміни конфігурації витримують оновлення до нових версій програмного забезпечення. Деякі налаштування (наприклад, код, який використовує заздалегідь визначені "гачки", які викликаються до / після відображення екранів даних), зберігають оновлення, хоча вони потребують повторного тестування. Інші налаштування (наприклад, ті, що пов'язані з внесенням змін до основних структур даних) перезаписуються під час оновлення, і вони повинні бути повторно впроваджені.

Переваги налаштування включають в себе:

- Покращує прийом користувачів
- Надає потенціал для отримання конкурентних переваг по відношенню до компаній, які використовують лише стандартні функції

Недоліки в налаштуваннях включають в себе те, що це може:

- Збільшити час та ресурси, необхідні для впровадження та підтримки
- Запобігання бездоганної взаємодії / інтеграції між постачальниками та замовниками через різницю між системами
- Обмежити здатність компанії у майбутньому вдосконалювати програмне забезпечення ERP
- Створюйте залежність від налаштування, підриваючи принципи ERP як стандартну програмну платформу

Переміщення даних - це процес переміщення, копіювання та перебудови даних з існуючої системи в систему ERP. Міграція має вирішальне значення для досягнення успіху та потребує значного планування. На жаль, оскільки міграція є однією з остаточних заходів до етапу виробництва, вона часто отримує недостатню увагу. Наступні кроки можуть структурувати планування міграції:

- Визначте дані для перенесення.
- Визначте терміни міграції.
- Створення шаблонів міграції даних для ключових компонентів даних
- Заморозити набір інструментів.
- Визначте налаштування ключових бізнес-облікових записів, пов'язаних з міграцією.
- Визначте політику та процедури архівування даних.

Часто міграція даних неповна, оскільки деякі дані в існуючій системі є або несумісними, або не потрібні в новій системі. Таким чином, існуючу систему, можливо, доведеться зберігати як архівну базу даних, щоб повернутися до того, як нова система ERP встановлена.

Переваги

Найважливіша перевага ERP полягає в тому, що інтеграція безлічі бізнес-процесів заощаджує час і витрати. Керування може приймати рішення швидше і з меншою кількістю помилок. Дані стають видимими в організації. Завдання, на користь цієї інтеграції, включають:

- Прогнозування продажів, що дозволяє оптимізувати інвентаризацію.
- Хронологічна історія кожної транзакції за допомогою відповідної компіляції даних у кожній області роботи.
- Відстеження замовлення, від прийняття до виконання
- Відстеження доходів, з рахунку-фактури через касовий квиток
- Відповідність замовлень на закупівлю (що було замовлено), надходження товарно-матеріальних цінностей (що прибули) та вартість (що було виставлено продавцем)

ERP системи централізують бізнес-дані, які:

- Усуває необхідність синхронізувати зміни між кількома системами - консолідація фінансів, маркетингу, продажів, кадрових ресурсів та виробничих додатків
- Забезпечує легітимність та прозорість для кожного біта статистичних даних
- Сприяє стандартному іменування / кодування продуктів
- Забезпечує вичерпний перегляд підприємства (відсутність "островів інформації"), надання інформації в режимі реального часу доступне для управління будь-де, в будь-який час для прийняття належних рішень
- Захищає конфіденційні дані шляхом об'єднання кількох систем безпеки в єдину структуру

ERP може покращити якість та ефективність бізнесу. Забезпечуючи безперебійну роботу внутрішніх бізнес-процесів компанії, ERP може призвести до кращих результатів, які можуть принести користь компанії, наприклад, у сфері обслуговування та виробництва. ERP підтримує управління вищим рівнем, надаючи інформацію для прийняття рішень. ERP створює більш гнучку компанію, яка краще пристосовується до змін. Це також робить компанію більш гнучкою та менш жорсткою структурою, тому

компоненти організації працюють більш однорідно, покращуючи бізнес як внутрішньо, так і зовні.

ERP може покращити безпеку даних у закритому середовищі. Спільна система управління, така, як система, пропонована ERP-системами, дозволяє організаціям легше забезпечувати безпеку ключових даних компанії. Однак це змінюється з більш відкритою обстановкою, що вимагає додаткової перевірки особливостей безпеки ERP та внутрішньої політики компанії щодо безпеки. ERP надає більші можливості для співпраці. Дані на сучасному підприємстві включають багато форм, включаючи документи, файли, форми, аудіо та відео, а також електронні листи. Часто кожен носій даних має свій власний механізм для допуску до співпраці. ERP надає спільну платформу, яка дозволяє співробітникам витрачати більше часу на співпрацю над змістом, а не на оволодіння кривою навчання спілкування в різних форматах через розподілені системи.

Недоліки

- Налаштування може бути проблематичним. Порівняно з найкращим підходом, ERP можна розглядати як задоволення найнижчих потреб організації в загальному значенні, змушуючи організацію знаходити шляхи вирішення, що відповідають унікальним вимогам.
- Реінжиніринг бізнес-процесів відповідно до системи ERP може пошкодити конкурентоспроможність або переорієнтувати фокус з інших важливих заходів.
- ERP може коштувати більше, ніж менш інтегрованих або менш комплексних рішень.
- Високі витрати на комутацію ERP можуть збільшити обмінну потужність постачальника ERP, що може збільшити витрати на підтримку, обслуговування та оновлення.
- Подолання опору для обміну конфіденційною інформацією між відомствами може відвернути увагу керівництва.
- Інтеграція справді незалежного бізнесу може створити непотрібні залежності.
- Великі вимоги до навчання приймають ресурси з щоденних операцій.
- Гармонізація систем ERP може бути мамонтовым завданням (особливо для великих компаній) і вимагає багато часу, планування та грошей.

Майбутнє ERP

Термін "постмодерністський ERP" був створений компанією Gartner у 2013 році, коли він вперше з'явився у серії "Прогнози 2014". Відповідно до визначення стратегії постмодернізму ERP у Gartner, спадкові, монолітні та спеціально налаштовані набори ERP, в яких всі компоненти значною мірою залежать один від одного, повинні рано чи пізно замінюватися сумішшю як обласних, так і локальних додатків, які більш вільно пов'язані і можуть бути легко обмінені, якщо це необхідно.

Основна ідея полягає в тому, що все ще має бути базовий ERP-рішення, яке би охоплювало найважливіші бізнес-функції, тоді як інші функції будуть покриватися спеціалізованими програмними рішеннями, які просто розширюють основні ERP. Ця концепція подібна до так званого найкращого підходу до впровадження програмного забезпечення, але з нею не слід плутати. Хоча в обох випадках додатки, що складаються з усього, відносно слабко пов'язані і досить легко взаємозамінні, у випадку останнього немає рішень ERP взагалі. Замість цього кожна бізнес-функція охоплюється окремим програмним рішенням.

Проте, немає жодного золотого правила щодо того, які ділові функції повинні бути частиною основної ERP, і що слід охопити додатковими рішеннями. За словами Gartner, кожна компанія повинна визначити свою постмодерністську стратегію ERP, засновану на внутрішніх та зовнішніх потребах компанії, операціях та процесах. Наприклад, компанія може визначити, що основне ERP-рішення повинно охоплювати ті бізнес-процеси, які повинні залишатися за брандмауером, і, таким чином, вибирати залишити свій основний ERP на місці. У той же час, інша компанія може вирішити розмістити основне ERP-рішення в хмарі та перемістити лише кілька ERP-модулів як додаткові рішення для локальних.

Основною перевагою, яку компанія отримує від реалізації постмодерної стратегії ERP, є швидкість та гнучкість при реагуванні на несподівані зміни в бізнес-процесах або на організаційному рівні. З більшістю додатків, що мають відносно вільний зв'язок, їх досить легко замінити або модернізувати, коли це необхідно. Крім того, згідно з наведеними вище прикладами, компанії можуть вибирати та об'єднувати хмарне та місцеві рішення, які найбільш підходять для їх потреб ERP. Недоліком постмодерністського ERP є те, що це, швидше за все, призведе до збільшення кількості постачальників програмного забезпечення, яким компанії доведеться керувати, а також поставити додаткові проблеми інтеграції для центрального IT.

2.3 Інструменти реалізації систем автоматизованого управління підприємства

Класичні системи, на відміну від так званого програмного забезпечення «в коробці», належать до категорії «важких» замовних програмних продуктів — їхній вибір, придбання і впровадження, як правило, вимагають ретельного планування в рамках тривалого проекту з участю партнерської компанії — постачальника або консультанта. Оскільки КІС будуються за модульним принципом, замовник часто (принаймні, на ранній стадії таких проектів) купує не повний спектр модулів, а обмежений їхній комплект. У ході впровадження проектна команда, як правило, протягом декількох місяців (до року) здійснює налаштування модулів, що поставляються.

Загалом такі системи реалізуються на різних високорівневих мовах програмування та в сукупності з іншими технологіями як бази даних, серверні масиви і тд.

Також існує декілька підходів для формування структури таких систем:

Монолітні додатки

У програмній інженерії монолітне застосування описує однорівневе програмне додаток, в якому користувальницький інтерфейс і код доступу до даних об'єднуються в єдину програму з однієї платформи.

Монолітне додаток є автономним і незалежним від інших обчислювальних додатків. Філософія дизайну полягає в тому, що програма відповідає не лише за певне завдання, але й може виконувати кожен крок, необхідний для виконання певної функції. Сьогодні деякі програми персональних фінансів є монолітними в тому сенсі, що вони допомагають користувачеві виконувати повне завдання, закінчуються, і є "приватними інформаційними сховищами", а не частинами більшої системи програм, які працюють разом. Деякі текстові процесори є монолітними додатками. Ці програми іноді пов'язані з мейнфреймами.

У програмній інженерії монолітне застосування описує програмне додаток, розроблений без модульності. Модульність бажана, в цілому, оскільки вона підтримує повторне використання частин логіки додатків, а також полегшує технічне обслуговування, дозволяючи ремонтувати або замінювати частини програми без необхідності оптової заміни.

Модульність досягається різними масштабами за допомогою різних модулярних підходів. Модулярність на основі коду дозволяє розробникам повторно використовувати та відновлювати частини програми, але для виконання цих функцій технічного

обслуговування потрібні інструменти розробки (наприклад, можливо, потрібно буде перекомпілювати програму). Модульність на основі об'єкта забезпечує додаток як сукупність окремих виконуваних файлів, які можна незалежно підтримувати та замінювати без перерозподілу всієї програми (наприклад, файли Microsoft DLL, файли спільного об'єкта Sun / UNIX). Деякі можливості обміну повідомленнями об'єктів дозволяють розповсюджувати об'єктові програми на кількох комп'ютерах (наприклад, Microsoft COM +). Сервісні архітектури використовують специфічні стандарти зв'язку / протоколи для зв'язку між модулями.

У своєму оригінальному використанні термін "моноліт" описував величезні основні програми для рамок без використання модульності. Це - у поєднанні з швидким збільшенням обчислювальної потужності і, отже, стрімким зростанням складності проблем, які можуть бути вирішені програмним забезпеченням, - призвели до незмінних систем та «кризи програмного забезпечення».

Мікросервіси

Мікросервіси - архітектурний стиль, в якому єдине додаток будується як сукупність невеликих сервісів, кожен з яких працює в своєму власному процесі та спілкується з іншими, використовуючи легкі механізми, як правило, HTTP. Ці служби розроблені відповідно до потреб бізнесу та розгортаються незалежно від використання типово повністю автоматизованого середовища. Існує абсолютний мінімум централізованого управління цими послугами. Самі по собі вони можуть бути написані з використанням різних мов та технологій зберігання даних.

Архітектура мікро-серверів добре підходить для безперервного процесу постачання, на відміну від сервісної орієнтованої мікро-сервісної архітектури, вона спрямована на створення однієї програми, тоді як сервісно-орієнтована система - це набір програм, які взаємодіють один з одним.

Основні риси:

- Високий рівень незалежності: незалежний розвиток, незалежне розгортання
- Незалежне масштабування
- Мала кодова база зменшує кількість конфліктів і дозволяє швидко залучати нових розробників
- Простота заміни однієї реалізації послуги на іншу

- Легкість додавання нових функцій до системи
- Ефективне використання ресурсів
- Еластичність: відмова однієї служби зазвичай не призводить до відмови системи
- Послуги організовані відповідно до ділової логіки, яку вони виконують
- Кожна служба може бути реалізована незалежно від будь-якої іншої мови програмування, СУБД та ін.
- Архітектурно побудовано за симетричним принципом (виробник-споживач)

Філософія мікросервісного підходу аналогічна філософії Unix: "Робіть одне і робіть це якісно":

- Послуги невеликі, розбиті на одну функцію
- Організаційна культура повинна включати автоматизацію розгортання та тестування
- Принципи культури та дизайну повинні охоплювати обробку збоїв та недоліків
- Кожна послуга є гнучкою, відмовостійкою, легко інтегрованою з іншими службами, функціонально мінімальною та повною.З точки зору якості існує методологія, яка описує основні риси, які мають бути притаманні застосунку з добре продуманою архітектурою: методологія застосунку дванадцяти факторів. Методологія добре підходить для розробки застосунків, зокрема мікросервісів, які призначені для розгортання в хмарному середовищі.

Критика

Архітектура мікросервісів піддається критиці переважно через такі проблеми:

- Мікросервіс успадковує всі проблеми розподілених систем (складність розподілених транзакцій, остаточна послідовність, теорема CAP)
- Значні накладні витрати на інфраструктуру, моніторинг та операційну діяльність
- ускладнена налагодження, налагодження, трасування
- Відсутність узгодження між розробниками: різні погляди на переваги мікро-серверної архітектури порівняно з традиційними монолітними можуть призвести до великої кількості обговорень, що призводить до втрати часу та зниженої продуктивності.

- Обмеження, такі як "служба однієї команди", створюють перешкоди: коли одна команда для розробки свого сервісу блокується відсутністю необхідної функціональності послуги, розробленої іншою командою.
- Незалежність послуг призводить до дублювання коду (утиліти, робота з базами даних, об'єкти передачі даних тощо)
- Проблеми зі стабільністю мережевого зв'язку між службами, латентність мережі, маршування / демаркація даних
- Складне тестування та розгортання
- Складна безпека

2.4 Висновки

В даному розділі розглянуто приклади систем автоматизації, контролю та налаштування підприємства. Також була досліджені різні приклади реалізації таких систем, підходи до їх реалізації та проблематика яку вирішують дані системи.

Згідно отриманних даних можна розпочати проектування власної автоматизованної системи для управління підприємством. Відповідно до сучасних реалій, найбільш поширенна моделі розробики це: монолітні системи або системи які складаються з декількох невеликих систем. Вибір архітектури залежить від кількості проблем які система буде вирішувати.

Розділ 3. Проектування та реалізація системи налаштування автоматизованого управління підприємством

3.1 Вибір архітектури додатку

Клієнт-серверна архітектура набула своєї популярності завдяки динамічному розвитку мережі Інтернет та зосередження значної частини інформації в базах даних на серверах.

Створення архітектури «клієнт-сервер» знаменувало новий етап розвитку мережевих інформаційних технологій. Це стало можливим завдяки збільшенню об'ємів внутрішньої та зовнішньої пам'яті, підвищенню швидкодії електронно-обчислювальних машин (ЕОМ), збільшенню швидкості передачі даних.

Концепція «клієнт-сервер» пов'язана з комп'ютерами спільного користування (серверами), які керують спільними ресурсами, і надають доступ до цих ресурсів як до сервісу своїм клієнтам. Обчислювальні мережі, побудовані на основі концепції «клієнт-сервер», дають змогу: реалізувати кооперативне управління ресурсами ЕОМ; виробити розподіл доступу до даних і процесів їх оброблення між множиною робочих станцій та сервером.

Клієнт — робоча станція, що взаємодіє з користувачем, здатна виконувати потрібні обчислення і забезпечує приєднання до обчислювальних ресурсів та БД, засобів їх оброблення, а також засобів організації інтерфейсів. Як ЕОМ клієнта, може бути використана будь-яка ЕОМ.

Концепція «клієнт-сервер» означає, що кожна технологічна процедура потребує наявності трьох елементів: клієнта, який запитує інформацію; серверу, що цю інформацію надає; власне мережі. Сервер можна розглядати: як елемент апаратури, який забезпечує спільно використовуваний сервіс у мережевому середовищі; як програмний компонент, що надає спільний функціональний сервіс іншим програмним компонентам; як поєднання ЕОМ і програми. Клієнта можна розглядати: як ЕОМ; як додаток, що формує і спрямовує запит до серверу. Він відповідає за оброблення, виведення інформації та передачу запитів серверу. Програма-сервер приймає запит, обробляє його і відправляє результат клієнту.

Користувач взаємодіє тільки з програмою-клієнтом. При цьому в концепції «клієнт-сервер» програми клієнта та його запити зберігаються окремо від системи керування БД.

Клієнт-серверну архітектуру можна означити, як концепцію інформаційної мережі в якій основна частина її ресурсів зосереджена в серверах, обслуговуючих своїх клієнтів. Така архітектура визначає такі типи компонентів

- набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;
- набір клієнтів, які використовують сервіси, що надаються серверами;
- мережа, яка забезпечує взаємодію між клієнтами та серверами.

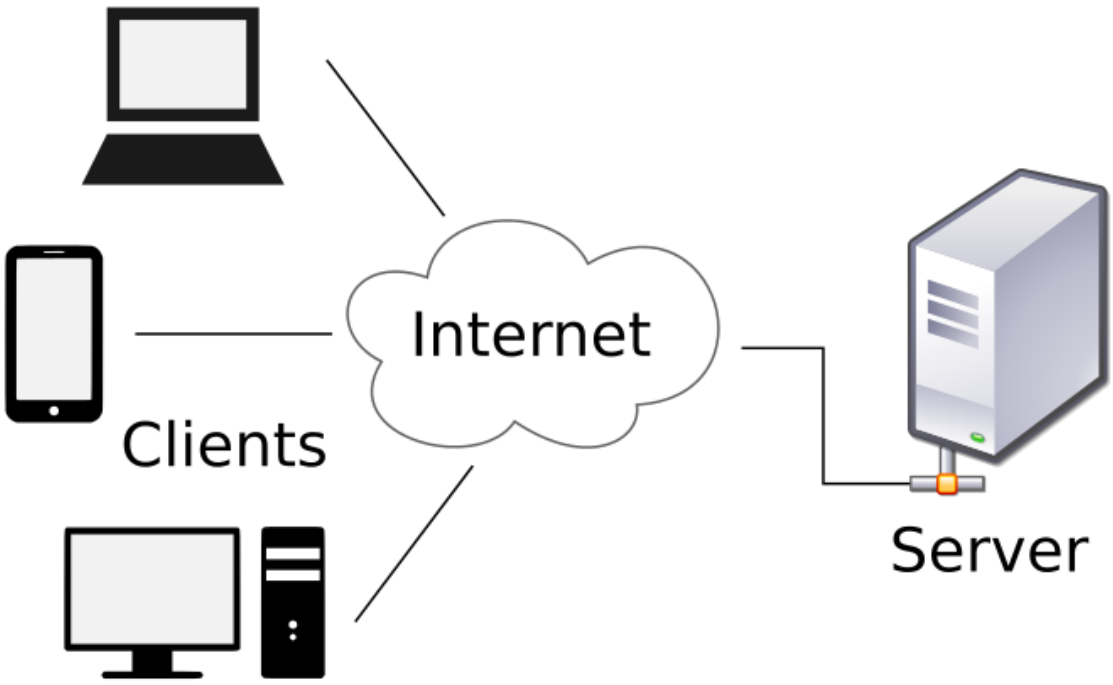


Рисунок 4. Загальна структура клієнт-серверної архітектури

Правила взаємодії між клієнтом і сервером називаються протоколом обміну (протоколом взаємодії)

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

- рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;
- прикладний рівень, який реалізує основну логіку застосунку і на якому здійснюється необхідна обробка інформації;
- рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дволанкова клієнт-серверна архітектура передбачає взаємодію двох програмних модулів—клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

- модель тонкого клієнта, в рамках якої вся логіка застосунку та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;
- модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Товстими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, мобільні телефони та ін.

Трьохланкова клієнт-серверна архітектура, яка почала розвиватися з середини 90-х років, передбачає відділення прикладного рівня від управління даними. Відокремлюється окремий програмний рівень, на якому зосереджується прикладна логіка застосунку. Програми проміжного рівня можуть функціювати під управлінням спеціальних серверів застосунків, але запуск таких програм може здійснюватися і під управлінням звичайного веб-сервера. Нарешті, управління даними здійснюється сервером даних.

Дволанкова архітектура простіша, так як всі запити обслуговуються одним сервером, але саме через це вона менш надійна і висуває підвищені вимоги до продуктивності сервера.

Триланкового архітектура складніша, але завдяки тому, що функції розподілені між серверами другого і третього рівня, ця архітектура проявляє:

1. високий ступінь гнучкості і масштабованості.
2. високу безпеку (тому що захист можна визначити для кожного сервісу або рівня).
3. високу продуктивність (тому що завдання розподілені між серверами).

Прикладом клієнт-серверної взаємодії є сервіс WWW. Існує величезна кількість веб-серверів, на яких розміщується та чи інша інформація. У найпростішому випадку ця інформація являє собою набір веб-сторінок, які можуть зберігатися на сервері у вигляді файлів, розмічених за допомогою мови розмітки HTML. Але ситуація, як правило, є складнішою; значна частина веб-ресурсів на сучасному етапі є динамічними, тобто вони не існують в заздалегідь підготовленому вигляді, а створюються безпосередньо в процесі обробки запиту від користувача.

Основна ідея архітектури «клієнт-сервер» полягає в поділі мережевого додатки на кілька компонентів, кожен з яких реалізує специфічний набір сервісів. Компоненти такого додатку можуть виконуватися на різних комп'ютерах, виконуючи серверні і/або клієнтські функції. Це дозволяє підвищити надійність, безпеку і продуктивність мережевих додатків і мережі в цілому.[6]

Ролі серверів

Роль—це функція сервера (наприклад поштовий, контролер домена тощо). Один сервер може відігравати як одну так і декілька ролей одночасно.

Взалежності від ролі, сервісу який надається, розрізняють такі сервери:

Веб-сервер (Web Server)

Сервер, що приймає HTTP-запити від клієнтів, зазвичай веб-браузерів, видає їм HTTP-відповіді, зазвичай разом з HTML-сторінкою, зображенням, файлом, медіа-потоком або іншими даними. Веб-сервер—основа Всесвітньої павутини.

Веб-сервером називають як програмне забезпечення, що виконує функції веб-сервера, так і комп'ютер, на якому це програмне забезпечення працює.

Клієнти дістаються веб-сервера за URL-адресою потрібної їм веб-сторінки або іншого ресурсу.

Сервер застосунків (Application Server)

Сервер, що виконує деякі прикладні програми. Термін також відноситься і до програмного забезпечення, що встановлено на такому сервері і забезпечує виконання прикладного ПЗ.

Сервери баз даних

Сервери баз даних використовуються для обробки запитів користувачів на мові SQL. При цьому СУБД знаходиться на сервері, до якого і підключаються клієнтські програми.

Файловий сервер (File Server)

Сервер що зберігає інформацію у вигляді файлів і представляє користувачам доступ до неї. Як правило файл-сервер забезпечує і певний рівень захисту від несакціонірованного доступу.

Сервер друку (Print Server)

Сервери друку використовуються для надання та управління доступом до принтерів. Роль сервера друку дозволяє управляти принтерами через веб-оглядач, друкувати через URL принтера, використовуючи протокол IPP, а також підключати принтери, використовуючи Point and Print.

Поштовий сервер (Mail Server)

Дозволяє обслуговувати базові поштові скриньки ваших користувачів і дозволяє приймати і відправляти пошту з сервера. Вхідна пошта може зберігатися на сервері, а потім забиратися користувачем по протоколу POP3. Для ролі поштового сервера ви повинні мати: Активне з'єднання з інтернет Зареєстроване доменне ім'я Запис MX у провайдера для вашого поштового домен

Термінальний сервер (Terminal Server)

Сервер, що надає клієнтам обчислювальні ресурси (процесорний час, пам'ять, дисковий простір) для вирішення завдань. Технічно термінальний сервер — надпотужний комп'ютер (або кластер), підключений до мережі з термінальними клієнтами — у котрих є, як правило, малопотужні або застарілі робочі станції або спецрішення для доступу до термінального сервера. Термінальний сервер служить для віддаленого обслуговування користувача з наданням робочого столу.

Remote Access/VPN Server

Сервери віддаленого доступу і VPN надають точку входу в вашу мережу для віддалених користувачів. Використовуючи роль Remote Access / VPN Server, ви можете реалізувати протоколи маршрутизації для середовищ LAN і WAN. Ця роль підтримує модемні з’єднання і VPN через інтернет.

DNS Server

Служба DNS дозволяє перетворювати доменні імена (FQDN) в адреси IP.

DHCP Server

Сервер DHCP дозволяє клієнтам отримувати свій IP за потребою. Сервер DHCP також надає додаткову інформацію для конфігурації мережі—адреса серверів DNS, WINS і т.п.

Streaming Media Server

Використовуються для управління і доставки мультимедійного контенту—потокового відео та аудіо—через інтранет або інтернет

Ігровий сервер (Game server)

Сервер, що забезпечує зв’язок між різними клієнтами, надаючи їм можливість комунікації один з одним в рамках програмної оболонки конкретної гри.

3.2 Проектування серверу обробки запитів

В якості серверного рішеня при проектувані системи, я вирішив використовувати Ruby фреймворк – Ruby on Rails, з яким я ознайомився під час проходження преддипломної практики.

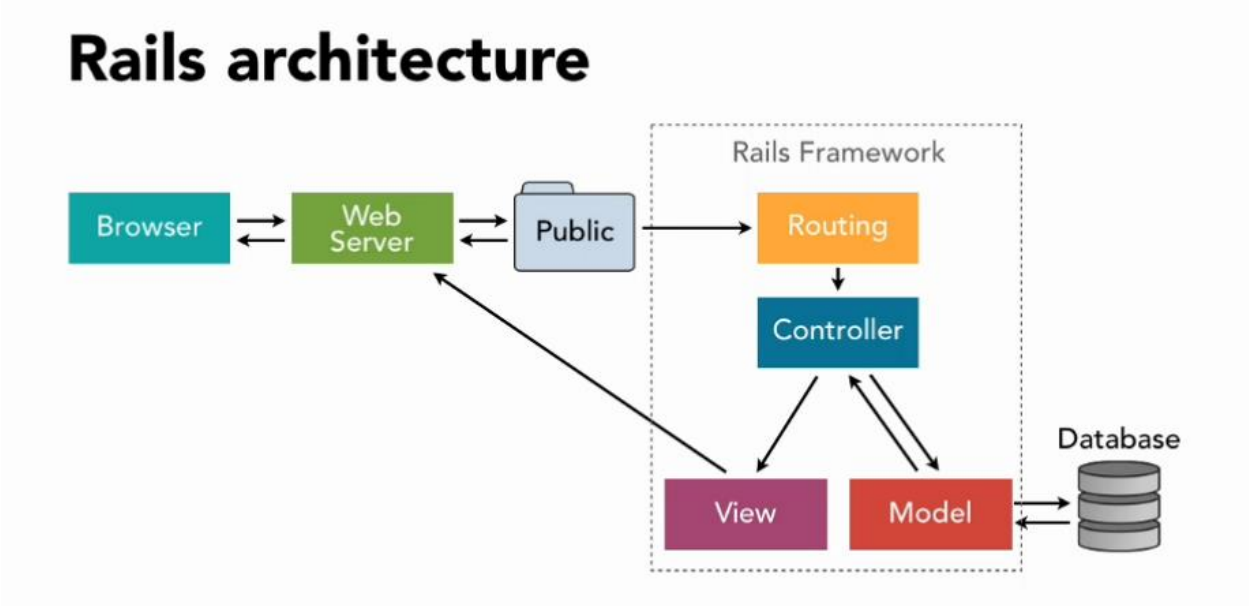


Рисунок 5. Структура Ruby on Rails

Особливості фреймворку Ruby on Rails

Ruby – динамічна мова програмування, яка має на меті збільшити простоту та продуктивність вихідного коду. Він має зручний синтаксис, який приємно читати та легко писати.

Ruby on Rails – фреймворк, написаний на мові програмування Ruby, тобто програмне забезпечення, що полегшує розробку та об'єднання декількох окремих складових проекту (наприклад, аутентифікація та авторизація користувачів або каталог статей в блозі).

Фреймворк, на відміну від CMS (система керування вмістом), яку може розгорнути та налаштувати навіть не-програміст, потребує проектування та розробки кваліфікованими спеціалістами. Але на ньому зручніше та швидше створювати проекти, які зовсім відрізняються функціоналом від типового сайту. А до веб-студій та агентств нечасто приходять за повністю типовими сайтами, оскільки замовники часто змінюють поведінку "на льоту". [7]

Ruby on Rails або Rails - це графічна веб-програма на стороні сервера, написана в Ruby під ліцензією MIT. Rails - це структура моделі-перегляду-контролера (MVC), яка забезпечує стандартні структури для бази даних, веб-сервісу та веб-сторінок. Він заохочує та полегшує використання веб-стандартів, таких як JSON або XML для передачі даних,

HTML, CSS та JavaScript для взаємодії користувачів. На додаток до MVC, Rails підкреслює використання інших відомих шаблонів та парадигм розробки програмного забезпечення, включаючи конвенцію над конфігурацією (CoC), не повторюйте себе (DRY) та активний шаблон запису.

Виникнення Ruby on Rails у 2000-х роках значною мірою вплинуло на розробку веб-додатків завдяки інноваційним функціям, таким як створення безперервних таблиць баз даних, міграцій та створення поглядів для швидкого створення додатків. Вплив Ruby on Rails на інші веб-ринки залишається очевидним і сьогодні з багатьма картами інших мов, що запозичують свої ідеї, включаючи Django у Python, Catalyst у Perl, Laravel в PHP, Phoenix in Elixir і Sails.js в Node.js.

Історія

Девід Хайнемейер Ханссон витягнув Ruby on Rails із своєї роботи з інструментом управління проектами Basecamp в компанії веб-додатків, яка також називається Basecamp. Ханссон вперше випустив Rails як відкрите джерело в липні 2004 року, але не ділив права користування проектом до лютого 2005 року. У серпні 2006 року рамки стали віхою, коли Apple оголосила про те, що вона буде завантажувати Ruby на Rails з Mac OS X v10.5 "Leopard", який був випущений в жовтні 2007 року.

Rails версії 2.3 була випущена 15 березня 2009 року, з основними новими розробками в шаблонах, двигунах, Rack і вкладеній модельній формі. Шаблони дозволяють розробнику створювати додатки скелетів з власними каменями та конфігураціями. Двигуни дають розробникам можливість повторно використовувати приладові деталі разом з маршрутами, дорогами та моделями. Інтерфейс веб-сервера Rack та Metal дозволяють писати оптимізовані фрагменти коду, які прокладаються навколо Action Controller. 23 грудня 2008 року було запущено Merb, інша платформа веб-додатків, і компанія Ruby on Rails оголосила, що вона буде працювати з проектом Merb, щоб вивести "найкращі ідеї Merb" в Rails 3, закінчивши "непотрібне дублювання" в обох спільнотах. Merb був об'єднаний з Rails як частина релізу 3.0 Rails.

Rails 3.1 був випущений 31 серпня 2011 року за участю міграції оборотної бази даних, активації трубопроводу, потокового, jQuery як бібліотеки JavaScript за замовчуванням, а також нещодавно введені в групу CoffeeScript та Sass.

Rails 3.2 був випущений 20 січня 2012 року за допомогою більш швидкого режиму розробки та двигуна маршрутизації (також відомий як двигун Подорожі), автоматичного запиту пояснення та позначення журналу. Rails 3.2.x - це остання версія, що підтримує Ruby 1.8.7. Rails 3.2.12 підтримує Ruby 2.0.

Rails 4.0 був випущений 25 червня 2013 р., Ввів російський ляльковий кешування, Turbolinks, Live Streaming, а також зробив активний ресурс, Active Record Observer та інші компоненти необов'язковими, розділивши їх як дорогоцінних каменів.

Rails 4.1 був випущений 8 квітня 2014 року, представивши передплатники Spring, Variants, Enums, Mailer та secrets.yml.

Rails 4.2 був випущений 19 грудня 2014 року, ввів Active Job, асинхронні електронні листи, Adequate Record, веб-консоль та зовнішні ключі.

Rails 5.0 був випущений 30 червня 2016 року, представивши Action Cable, API-режим і Turbolinks 5.

Rails 5.0.0.1 був випущений 10 серпня 2016 року за допомогою ексклюзивного використання рейок CLI над Rake та підтримкою Ruby версії 2.2.2 та вище.

Rails 5.1 був випущений 27 квітня 2017 р., Ввівши зміни в JavaScript інтеграції (керування JavaScript залежностями від NPM через пряжу, необов'язкове складання JavaScript за допомогою Webpack та перезапису Rails UJS для використання vanilla JavaScript, а не залежно від jQuery), системні тести використовуючи Capybara, зашифровані таємниці, параметризовані поштові повідомлення, прямі та вирішені маршрути та уніфікований форма_з помічником, що заміняє form_tag / form_for помічників.

Технічний огляд

Як і інші веб-додатки, Ruby on Rails використовує модель-view-controller (MVC) для організації прикладног опрограмування.

У конфігурації за замовчуванням модель на карті Ruby on Rails картує до таблиці в базі даних та файлі Ruby. Наприклад, типовий користувач класу, як правило, визначається у файлі 'user.rb' у каталозі додатків / моделей та пов'язаний з таблицею 'users' у базі даних. Хоча розробники можуть ігнорувати цю конвенцію та вибирати різні імена для своїх моделей, файлів і таблиць бази даних, це не є загальноприйнятою практикою і, як правило, не рекомендується відповідно до "конвенції над конфігурацією" філософії.

Контролер - це серверний компонент Rails, який реагує на зовнішні запити веб-сервера до програми, визначаючи, який файл перегляду потрібно відтворити. Контролер може також вимагати запит однієї або декількох моделей для інформації та передавати їх на вигляд. Наприклад, в системі бронювання авіакомпанії, контролер, що виконує функцію пошуку польотів, повинен буде запитати модель, що представляє окремі рейси, щоб знайти рейси, що відповідають пошуковому запиту, і, можливо, також потрібно буде запитувати моделі, що представляють аеропорти та авіакомпанії, щоб знайти пов'язані вторинні дані. Після цього контролер може передавати певну підмножину даних про польоти у відповідне вікно, яке міститиме суміш статичного HTML та логіки, які використовуватимуть дані польоту, щоб створити HTML-документ, що містить таблицю з однією рядком на один рейс. Контролер може надати одне або кілька дій. У Ruby on Rails дія, як правило, є базовою одиницею, яка описує, як реагувати на певний зовнішній запит веб-браузера. Також зауважте, що контролер / дія буде доступним для зовнішніх веб-запитів лише у тому випадку, якщо до нього буде зв'язаний відповідний маршрут. Rails заохочує розробників використовувати RESTful маршрути, які включають дії, такі як створення, створення, редагування, оновлення, знищення, показу та індексування. Ці відображення вхідних запитів / маршрутів до дій контролерів можна легко налаштувати в файлі конфігурації routes.rb.

Вигляд у налаштуваннях за замовчуванням Rails - це файл erb, який оцінюється та перетворюється в HTML під час виконання. Крім того, для перегляду можуть використовуватися багато інших систем шаблонів.

Ruby on Rails включає в себе інструменти, які полегшують загальні завдання розробки "поза рамкою", наприклад, будівельних лісів, які можуть автоматично створювати деякі моделі та перегляди, необхідні для базового веб-сайту. Також є WEBrick, простий веб-сервер Ruby, який поширюється разом з Ruby, і Rake, система збірки, яка поширюється як дорогоцінний камінь. Разом з Ruby on Rails, ці інструменти забезпечують базову середовище розробки.

Починаючи з версії 2.0, Ruby on Rails пропонує як HTML, так і XML як стандартні формати виводу. Останній - це засіб для RESTful веб-сервісів. Rails 3.1 представив Sass як стандартний шаблон CSS. За замовчуванням сервер використовує Embedded Ruby у переглядах HTML, при цьому файли мають розширення html.erb. Rails підтримує можливість заміни альтернативних мов шаблонів, таких як HAML та Mustache. Ruby on Rails 3.0 розроблено для роботи з Ruby 1.8.7, Ruby 1.9.2 та JRuby 1.5.2+; попередні

версії не підтримуються. Ruby on Rails 3.2 - це остання серія релізів, що підтримують Ruby 1.8.7.

Рамкова структура

Ruby on Rails розділений на різні пакети, зокрема ActiveRecord (об'єктно-реляційна система відображення для доступу до бази даних), Active Resource (забезпечує веб-сервіси), Action Pack, Active Support та Action Mailer. До версії 2.0 Ruby on Rails також містив пакет Action Web Service, який тепер замінено активним ресурсом. Окрім стандартних пакетів, розробники можуть створювати плагіни для розширення існуючих пакетів. Раніше Rails підтримує плагіни в межах власної системи; Версія 3.2 призупиняє їх на користь стандартного Ruby "дорогоцінних каменів".

Розгортання

Ruby on Rails часто встановлюється за допомогою RubyGems, менеджера пакетів, який входить до складу поточних версій Ruby. Багато безкоштовних Unix-подібних систем також підтримують установку Ruby on Rails та її залежностей за допомогою власної системи керування пакунками.

Ruby on Rails, як правило, розгортається за допомогою сервера баз даних, таких як MySQL або PostgreSQL, а також веб-сервер, такий як Apache, що працює на модулі Phusion Passenger.

Філософія та дизайн

Ruby on Rails має на меті підкреслити конвенцію над конфігурацією (CoC), а також принцип «Не повторюйте себе (DRY)».

"Конвенція про конфігурацію" означає, що розробник повинен лише вказати нетрадиційні аспекти програми. Наприклад, якщо в моделі є клас Sale, то відповідна таблиця в базі даних називається продажами за замовчуванням. Лише в тому випадку, якщо він відхиляється від цієї конвенції, наприклад, виклик таблиці "продукти, що продаються", розробник повинен написати код щодо цих імен. Взагалі, конвенції Ruby on Rails призводять до меншого коду і меншого повторення.

"Не повторюй себе" означає, що інформація розташована в єдиному, однозначному місці. Наприклад, використовуючи модуль ActiveRecord Rails, розробникам не потрібно

вказати назви стовпців бази даних у визначенні класу. Замість цього, Ruby on Rails може отримати цю інформацію з бази даних на основі назви класу.

"Товсті моделі, худі контролери" означає, що більша частина логіки застосування повинна бути поміщена в модель, залишаючи контролер настільки світлим, наскільки це можливо.

Безпека

У березні 2012 року дослідник з питань безпеки Єгор Хомаков виявив вразливість "масове призначення", що дозволило віддалити певні програми Rails, і продемонструвала це, не зловмисну демонстрацію GitHub після того, як його попередні спроби розкрити інформацію були відхилені.

24 вересня 2013 року в Ruby on Rails було повідомлено про недоліки безпеки сеансу cookie. У конфігурації за замовчуванням весь хеш сесії зберігається в файлі cookie сеансу, відомі як CookieStore, що дозволяє будь-якому аутентифікованому сеансу, що має файл cookie сеансу, входити як цільовий користувач у будь-який час у майбутньому. Як спосіб вирішення проблеми, адміністраторам рекомендується налаштовувати файли cookie для зберігання на сервері за допомогою таких механізмів, як ActiveRecordStore.

Дослідники Даніель Джексон та Джозеф Бейдж розробили відладчик даних, який називається "Spacе", який може аналізувати доступ до даних програми Rails і визначити, чи правильно дотримується правил щодо обмежень доступу. 15 квітня 2016 р. Поруч повідомили, що аналіз 50 популярних веб-застосунків, виявив 23 недоліки безпеки, раніше невідомі.

Переваги платформи Ruby on Rails

Основною перевагою мови програмування Ruby та фреймворку Ruby onRails є швидкість розробки. На практиці швідкість розробки проектів на RoR вище на 30-40% по відношенню до інших мов програмування або фреймворків.

Такий приріст швидкості розробки пояснюється широким набором готових до роботи «з коробки» інструментів RoR, можливістю використовувати готові бібліотеки інших розробників та, звичайно, зручністю програмування на Ruby.

Крім цього, на відміну від інших фреймворків, до складу RoR входять ефективні засоби для автоматизованого тестування, що прискорює перехід проекту від стадії

“програму написано” до стадії “програма працює без помилок”. Цей перехід майже завжди займає найбільшу кількість часу під час реалізації майже будь-яких проектів.

Також варто відмітити, що Ruby on Rails забезпечує кращу безпеку для додатків. Під час використання інструментів RoR виключені SQL-ін’єкції та XSS-атаки, всі вхідні параметри екрануються за замовчанням, вихідні змінні в шаблонах також екрануються. У розробника майже немає шансів допустити помилку безпеки.

Деякі розробники, що недостатньо добре знайомі з цією технологією, чомусь вважають, що інтернет-проекти на RoR погано масштабуються. Як приклад майже всі розробники наводять Twitter, який в свій час відмовився від Rails через якісь внутрішні причини. Але треба звернути увагу на більш відомі проекти, як Kickstarter, Groupon або Basecamp – всі ці проекти написані звикористанням Rails без проблем з масштабуванням. В будь-якому випадку, проблеми продуктивності будь-якого проекту, - це не проблеми помилковоговибору платформи чи мови програмування. Найчастіше ці проблеми були викликані помилками під час проектування архітектури проекту, кешуванням даних або неоптимальним вибором СУБД.

Обмеження фреймворку

Розробників на Ruby on Rails менше, ніж розробників на PHP та його фреймворках, оскільки тут вищий поріг входження та, зазвичай, програміст приходить до Ruby вже після декількох років PHP. Але при цьому слід пам’ятати, що досвідчених розробників дуже мало в цих двох сферах.

Важливим обмеженням RoR вважається вбудована ORM ActiveRecord та модуль Active Support, які містять велику кількість допоміжних методів, які майже не використовуються в проектах, але ці модулі можна замінити чимось менш ресурсозатратним (Sequel).

Типові проекти на Ruby on Rails

Фреймворк RoR найкраще підходить для наступних типів проектів:

• Інтернет-магазини з складною системою фільтрації вмісту, модулями

підбору та інтеграціями із зовнішніми сервісами.

• Купонні сервіси, веб-сервіси для колективних закупівель.

• Інформаційні портали, електронні видання.

• Біржи та торгівельні майданчики.

• Сайти повідомлень та знайомств.

• Соціальні мережі.

• Незвичайні/нестандартні, технічно важкі проекти.

• Сервіси та SaaS-проекти.

Встановлення Rails:

```
lecik@ruby-2.5.1:~/diploma$ gem install rails
Fetching: concurrent-ruby-1.0.5.gem (100%)
Successfully installed concurrent-ruby-1.0.5
Fetching: i18n-1.1.1.gem (100%)
Successfully installed i18n-1.1.1
Fetching: thread_safe-0.3.6.gem (100%)
Successfully installed thread_safe-0.3.6
Fetching: tzinfo-1.2.5.gem (100%)
Successfully installed tzinfo-1.2.5
Fetching: activesupport-5.2.1.gem (100%)
Successfully installed activesupport-5.2.1
Fetching: rack-2.0.5.gem (100%)
Successfully installed rack-2.0.5
Fetching: rack-test-1.1.0.gem (100%)
Successfully installed rack-test-1.1.0
Fetching: mini_portile2-2.3.0.gem (100%)
Successfully installed mini_portile2-2.3.0
Fetching: nokogiri-1.8.5.gem (100%)
Building native extensions. This could take a while...
Successfully installed nokogiri-1.8.5
Fetching: crass-1.0.4.gem (100%)
Successfully installed crass-1.0.4
Fetching: loofah-2.2.3.gem (100%)
Successfully installed loofah-2.2.3
Fetching: rails-html-sanitizer-1.0.4.gem (100%)
Successfully installed rails-html-sanitizer-1.0.4
Fetching: rails-dom-testing-2.0.3.gem (100%)
Successfully installed rails-dom-testing-2.0.3
Fetching: builder-3.2.3.gem (100%)
Successfully installed builder-3.2.3
Fetching: erubi-1.7.1.gem (100%)
Successfully installed erubi-1.7.1
Fetching: actionview-5.2.1.gem (100%)
Successfully installed actionview-5.2.1
```

Рисунок 6. Встановлення додатку Ruby on Rails

Після встановлення додатку потрібно згенерувати шаблон для нашого проекту:

```
lecik@ruby-2.5.1:~/diploma$ rails new enterprise_config_system
      create
      create  README.md
      create  Rakefile
      create  .ruby-version
      create  config.ru
      create  .gitignore
      create  Gemfile
         run  git init from "."
Initialized empty Git repository in /home/lecik/diploma/enterprise_config_system/.git/
      create  package.json
      create  app
      create  app/assets/config/manifest.js
      create  app/assets/javascripts/application.js
      create  app/assets/javascripts/cable.js
      create  app/assets/stylesheets/application.css
      create  app/channels/application_cable/channel.rb
      create  app/channels/application_cable/connection.rb
      create  app/controllers/application_controller.rb
      create  app/helpers/application_helper.rb
      create  app/jobs/application_job.rb
      create  app/mailers/application_mailer.rb
      create  app/models/application_record.rb
      create  app/views/layouts/application.html.erb
      create  app/views/layouts/mailer.html.erb
      create  app/views/layouts/mailer.text.erb
      create  app/assets/images/.keep
      create  app/assets/javascripts/channels
      create  app/assets/javascripts/channels/.keep
      create  app/controllers/concerns/.keep
      create  app/models/concerns/.keep
      create  bin
      create  bin/bundle
      create  bin/rails
      create  bin/rake
```

Рисунок 7. Генерація базового шаблон

3.3 Проектування бази даних

База даних має підтримувати мову структурованих запитів SQL для можливості маніпулювання даних в контексті роботи системи, також підтримка функції backup, для відновлення даних у базі даних, якщо дані будуть пошкодженні внаслідок будь-яких подій.

На даний часи у вільному доступі є багатий вибір SQL баз даних. Наприклад: MySQL, PostgreSQL, Oracle, MS SQL. Кожна з цих баз даних має майже ідентичний набір функціональних можливостей, але для системи в контексті проекту найбільш підходяща є база даних PostgreSQL. Вона має нативну інтеграцію з Ruby on Rails. [12]

Переваги:

- Незалежність від конкретної СУБД

Не зважаючи на наявність діалектів і відмінностей в синтаксисі, більшість текстів SQL-запитів, що містять, DDL і DML, можуть бути досить легко перенесені з однієї СУБД в іншу. Існують системи, розробники яких спочатку орієнтувалися на застосування щонайменше кількох СУБД (наприклад: система електронного документообігу Documentum може працювати як з Oracle, так і з Microsoft SQL Server та IBM DB2). Природно, що при застосуванні деяких специфічних для реалізації можливостей, такого рівня перенесення дуже важко досягти.

- Наявність стандартів

Наявність стандартів і наборів тестів для виявлення сумісності та відповідності конкретній реалізації SQL загальноприйнятому стандарту тільки сприяє «стабілізації» мови. Щоправда, слід звернути увагу на той факт, що сам по собі стандарт місцями занадто формалізований і має завеликі розміри, наприклад, Core-частина стандарту SQL:2003 містить понад 1300 сторінок тексту.

- Декларативність

За допомогою SQL програміст описує лише дані, які потрібно витягнути або модифікувати. Те, яким чином це зробити, вирішує СУБД безпосередньо при обробці SQL-запиту. Не слід вважати, що це повністю універсальний принцип — програміст описує набір даних для вибірки або модифікації, проте йому при цьому корисно уявляти, як СУБД інтерпретуватиме текст його запиту. Такі моменти стають особливо критичними при роботі з великими базами даних та зі складними запитами — чим складніше сконструйований запит, тим більше варіантів написання (різних за швидкістю виконання, але тих самих за набором даних) він припускає.

Недоліки

Невідповідність реляційної моделі даних

Творець реляційної моделі даних Едгар Кодд, Крістофер Дейт та їхні прихильники вказують на те, що SQL не є істинно реляційною мовою. Зокрема, вони привертають увагу до таких проблем SQL:

- Рядки, що повторюються
- Невизначені значення (null)
- Явна вказівка порядку стовпчиків зліва направо
- Стовпчики без імені та імена стовпчиків, що дублюються
- Відсутність підтримки властивості «=»
- Використання вказівників
- Значна надлишковість

Складність

Хоча мову SQL було початково заплановано як засіб роботи кінцевого користувача, врешті-решт вона стала настільки складною, що перетворилася на інструмент програміста.

Відступи від стандартів

Не зважаючи на наявність міжнародного стандарту ANSI SQL-92, багато компаній, що займаються розробкою СУБД (наприклад, Oracle, Sybase, Microsoft, MySQL), вносять зміни до мови SQL, вживаної в розроблених ними СУБД. Цим вони створюють передумови відступу від стандарту. Завдяки такій діяльності для кожної конкретної СУБД з'являються специфічні діалекти мови SQL.

Складність роботи з ієрархічними структурами

Раніше SQL не пропонувала стандартного способу маніпуляції деревовидними структурами. Деякі постачальники СУБД запропонували свої рішення. Для прикладу, Oracle використовує вираз CONNECT BY. В наш час як стандарт прийнята рекурсивна конструкція WITH.

Наша система зможе надавати такий функціонал:

- Облік співробітників та обладнання
- Кожен співробітник матиме роль яка надає доступ до певного обладнання
- Підприємство має кабінети, кожен з кабінетів має певний набір обладнання, співробітників та підключенні загальні системи підприємства
- Облік систем та моніторінг параметрів, зміна налаштувань систем.

ERD Модель для проектування

Модель «сутність-зв'язок» (ER-модель) (англ. Entity-relationship model або entity-relationship diagram) — модель даних, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків. ER-модель — це мета-модель даних, тобто засіб опису моделей даних. Існує ряд моделей для представлення знань, але одним з найзручніших інструментів уніфікованого представлення даних, незалежного від програмного забезпечення що його реалізує, є модель «сутність-зв'язок». Важливим є той факт, що з моделі «сутність-зв'язок» можуть бути породжені всі існуючі моделі даних (ієрархічна, мережева, реляційна, об'єктна), тому вона є найзагальнішою. [8]

Модель сутність-зв'язок є результатом систематичного процесу, який описує та визначає деяку предметну область. Вона не визначає сам процес, а лише візуалізує його. Дані представлені у вигляді компонентів (сутностей), які пов'язані між собою певними зв'язками, які виражають залежності і вимоги між ними, такі як: одна будівля може бути розділена на нуль або більше квартир, але одна квартира може бути розташована лише в одній будівлі. Сутності можуть мати різні властивості (атрибути), які характеризують їх. Діаграми, створені для представлення цих сутностей, атрибутів і зв'язків графічно, називають сутність-зв'язок діаграмами.

ER-модель зазвичай реалізується в вигляді баз даних. У разі реляційної бази даних, в якій зберігаються дані в таблицях, кожен рядок кожної таблиці являє собою один екземпляр сутності. Деякі поля даних в цих таблицях вказують на індекси в інших таблицях. Такі поля є покажчиками фізичної реалізації зв'язків між сутностями.

Коли ми говоримо про сутність, ми зазвичай говоримо про деякий аспект реального світу, який можна виділити поміж інших аспектів. Сутність — це збірне поняття, деяка абстракція реально існуючого об'єкта, процесу, явища чи деякого уявлення про об'єкт. Хоча термін сутність найбільш вживаний, потрібно розрізняти поняття типу сутності та екземпляру сутності. Поняття тип сутності відноситься до набору однорідних особистостей, предметів, подій або ідей, виступаючих як ціле. Екземпляр сутності відноситься до конкретної речі в наборі. Наприклад, типом сутності може бути МІСТО, а екземпляром — Київ, Львів і т. д.

Виділяють три види сутностей: стрижнева, асоціативна (асоціація) і характеристична (характеристика):

- Стрижнева (сильна) сутність — незалежна від інших сутність. Стрижнева сутність не може бути асоціацією, характеристикою чи позначенням.
- Асоціативна сутність (або асоціація) виражає собою зв'язок «багато до багатьох» між двома сутностями. Є цілком самостійною сутністю. Наприклад, між сутностями ЧОЛОВІК і ЖІНКА існує асоціативний зв'язок, висловлюваний асоціативної сутністю ШЛЮБ.
- Характеристичну сутність ще називають слабкою сутністю. Вона пов'язана з більш сильною сутністю зв'язками «один до багатьох» і «один до одного». Характеристична сутність описує або уточнює іншу сутність. Вона повністю залежить від неї і зникає зі зникненням останньої. Наприклад, сутність Зарплата

є характеристикою конкретних працівників підприємства і не може в такому контексті існувати самостійно — при видаленні екземпляра сутності Працівника повинні бути видалені і екземпляри сутності Зарплата, пов'язані з видаленим працівником.

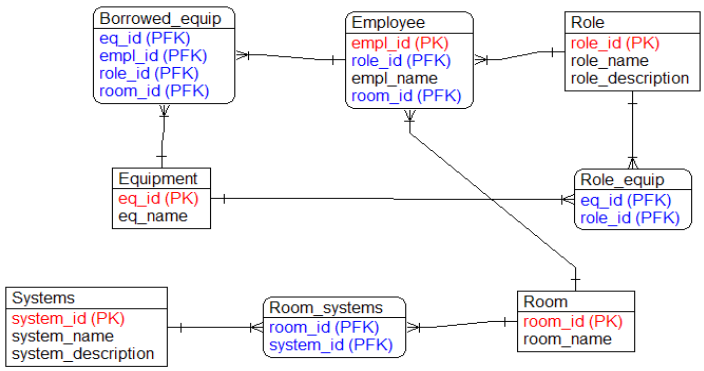
Позначення це така сутність, з якої інші сутності пов'язані за принципом «багато до одного» або «один до одного». Позначення, на відміну характеристики є самостійною сутністю. Наприклад, сутність Факультет позначає приналежність студента до даного підрозділу інституту, але є цілком самостійною. [9]

При моделюванні прийнято виражати (іменувати) сутність іменником або іменником з прикметником, що характеризує його, а зв'язок дієсловом, що поєднує два чи більше іменників.

Сутності та зв'язки можуть мати свої атрибути. Наприклад, сутність громадянин має атрибут номер паспорту, а зв'язок має між сутностями гравець та аккаунт володіє атрибутом останній вхід.

Кожна сутність (якщо це не слабка сутність) має мати мінімальний набір унікальних атрибутів, що зветься первинним ключем.

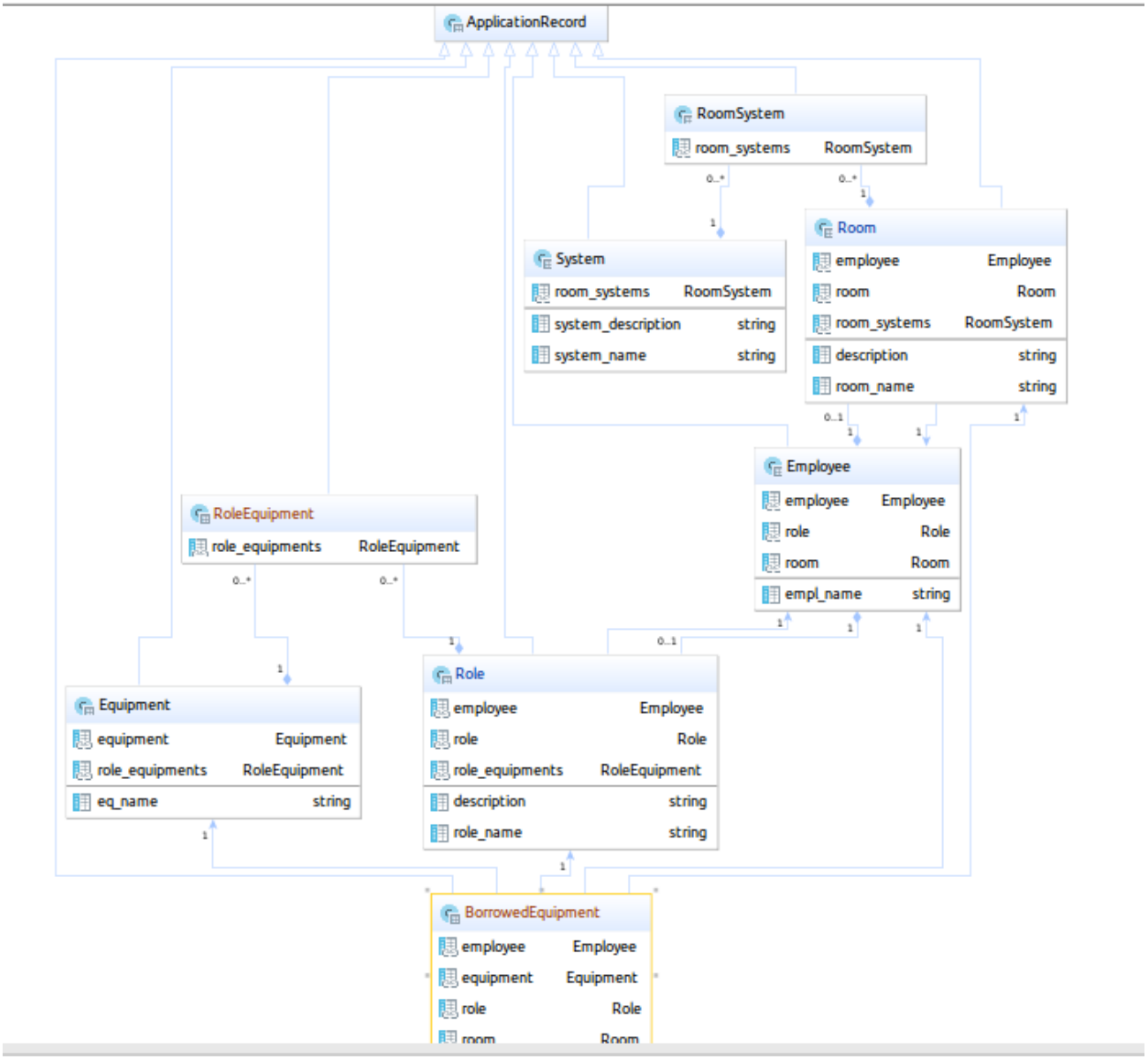
Розробимо Entity Relationship Diagram(ERD) для бази даних, яку проектуємо:



Діаграма 1. ERD – діаграма БД

Інструмент для проектування БД case studio дозволяє генерувати SQL- код для створення спроектованної БД.

Реалізація вище вказанної структури БД у Ruby on Rails:



Діаграма 2. Структура БД в додатку Rails

3.4 Проектування додатку конфігурації

Інструменти, що використовуються для розробки зовнішнього вигляду додатку.

Існує кілька інструментів і платформ (WordPress, Magento тощо), які можуть бути використані для розробки інтерфейсу веб-сайту, а також розуміння того, які інструменти найкраще підходять для конкретних завдань, відзначає різницю між розробкою зламаного сайту та добре розробленим , масштабованим сайтом.

Мова розмітки гіпертексту (HTML)

Мова розмітки гіпертексту (HTML) є основою будь-якого процесу розробки веб-сайту, без якого веб-сторінка не існує. Гіпертекст означає, що текст містить посилання, які називаються гіперпосиланнями, вбудованими в нього. Коли користувач натискає слово або фразу, що має гіперпосилання, вона принесе ще одну веб-сторінку. Мова розмітки означає, що текст можна перетворити на зображення, таблиці, посилання та інші представлення. Це HTML-код, який забезпечує загальну структуру того, як буде виглядати сайт. HTML був розроблений Тімом Бернерсом-Лі. Остання версія HTML називається HTML5 і була опублікована 28 жовтня 2014 року за рекомендацією W3. Ця версія містить нові ефективні способи обробки таких елементів, як відео та аудіо файли.

Каскадні таблиці стилів (CSS)

Каскадні таблиці стилів (CSS) контролює аспект презентації сайту і дозволяє вашому сайті мати свій унікальний вигляд. Це робиться за допомогою підтримки таблиць стилів, які сидять поверх інших правил стилю та запускаються на основі інших входів, таких як розмір екрана пристрою та роздільна здатність.

JavaScript

JavaScript є імперативною мовою програмування на основі подій (на відміну від декларативної мови моделі HTML), яка використовується для перетворення статичної HTML-сторінки в динамічний інтерфейс. Код JavaScript може використовувати модель документа об'єктів (DOM), надану стандартом HTML, для роботи з веб-сторінкою у відповідь на події, наприклад, користувальницький ввід.

Використовуючи техніку під назвою AJAX, код JavaScript також може активно завантажувати вміст з Інтернету (незалежно від оригінального пошуку HTML-сторінок), а також реагувати на подію на сервері, а також додавати справді динамічний характер до досвіду веб-сторінки.

WebAssembly

WebAssembly, що підтримується всіма основними браузерами (тобто від основних постачальників Google, Apple, Mozilla і Microsoft), є єдиною альтернативою JavaScript для роботи коду в веб-переглядачах (без допомоги плагінів, таких як Flash, Java або Silverlight; все це припиняється, оскільки браузери скидають підтримку плагінів). До його прийняття

існував asm.js (підмножина JavaScript, і таким чином строго працює у всіх браузерах), що також використовується як ціль компілятора з ефективною підтримкою в браузерах, таких як Internet Explorer 11; і для таких браузерів, які безпосередньо не підтримують WebAssembly, він може бути скомпільований для asm.js і тих веб-переглядачів, які підтримуються таким чином. Взагалі, програмісти не програмують безпосередньо в WebAssembly (або asm.js), але використовують мови, такі як Rust, C або C ++, або, теоретично, будь-яку мову, яка збирає її.

Цілі розвитку

Розробник інтерфейсу пам'ятає ці моменти, використовуючи наявні інструменти та методи для досягнення цього.

Доступність

З постійним розвитком для мобільних пристроїв, таких як смартфони та планшети, дизайнерам потрібно забезпечити належне їхнє розміщення веб-сторінок на всіх пристроях. Це можна зробити, створивши адаптивний веб-дизайн із використанням таблиць стилів в CSS.

Продуктивність

Цілі продуктивності в основному стосуються часу відтворення, маніпулювання HTML, CSS та JavaScript, щоб забезпечити швидке відкриття сайту.

Реалізація дизайну за допомогою вище зазначенних технологій:

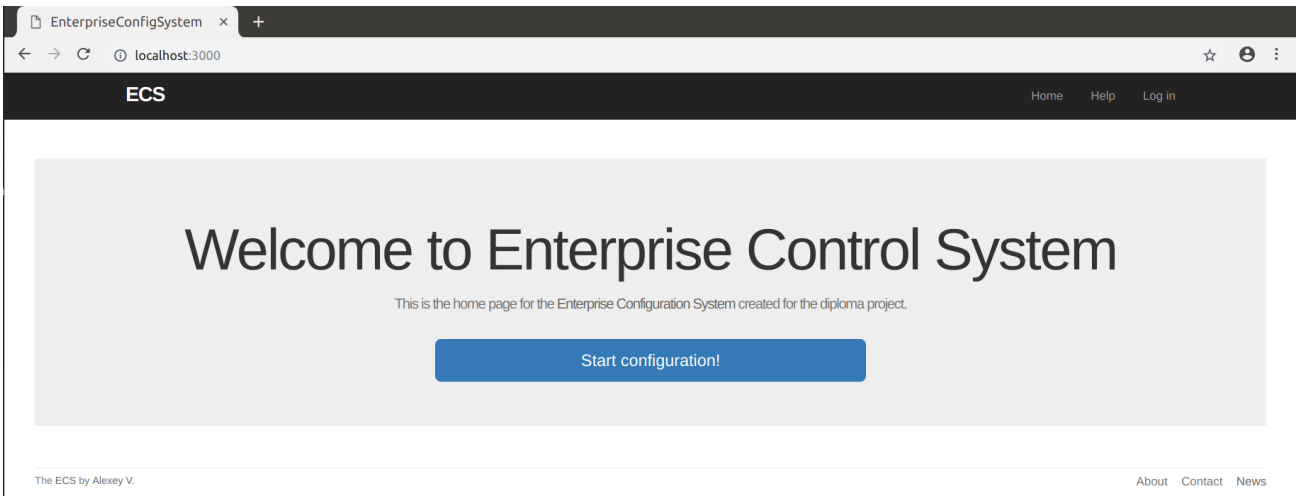


Рисунок 8. Домашня сторінка додатку

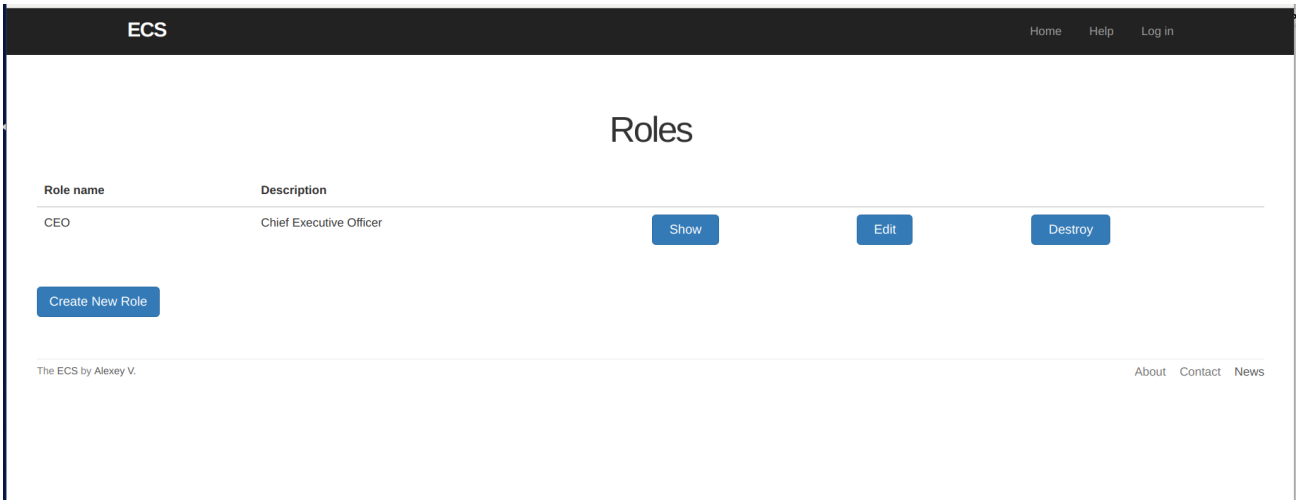


Рисунок 9. Ролі підприємства

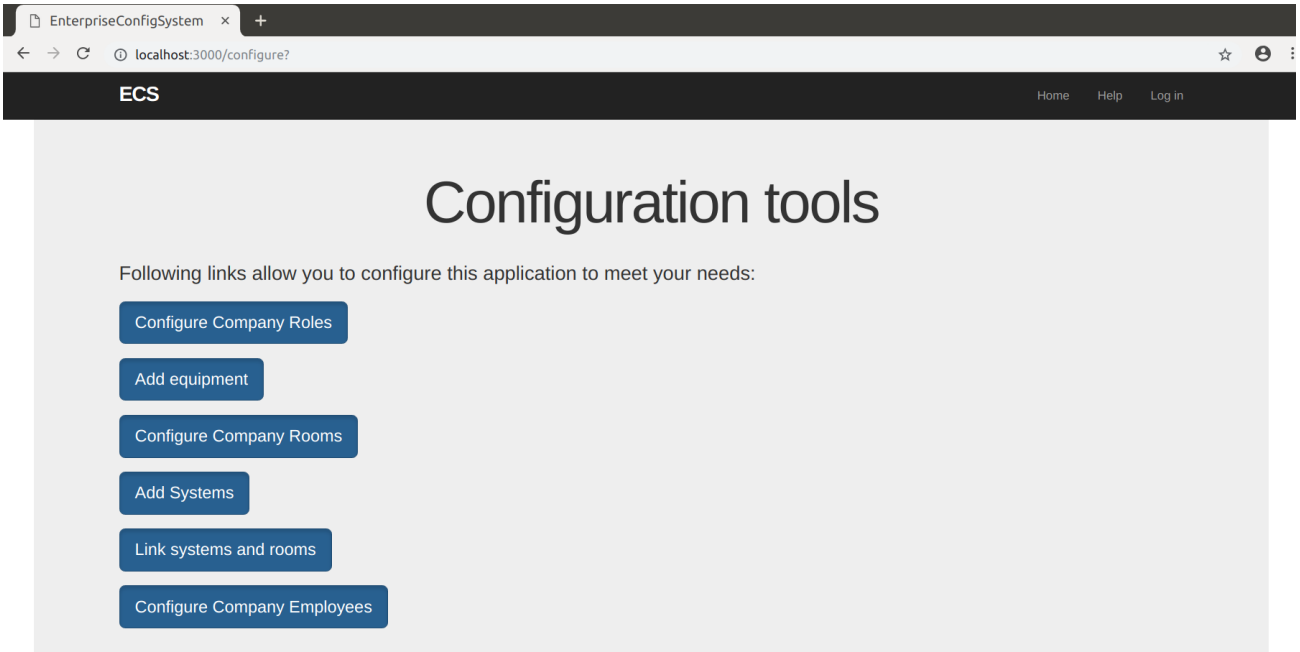


Рисунок 10. Конфігурація системи.

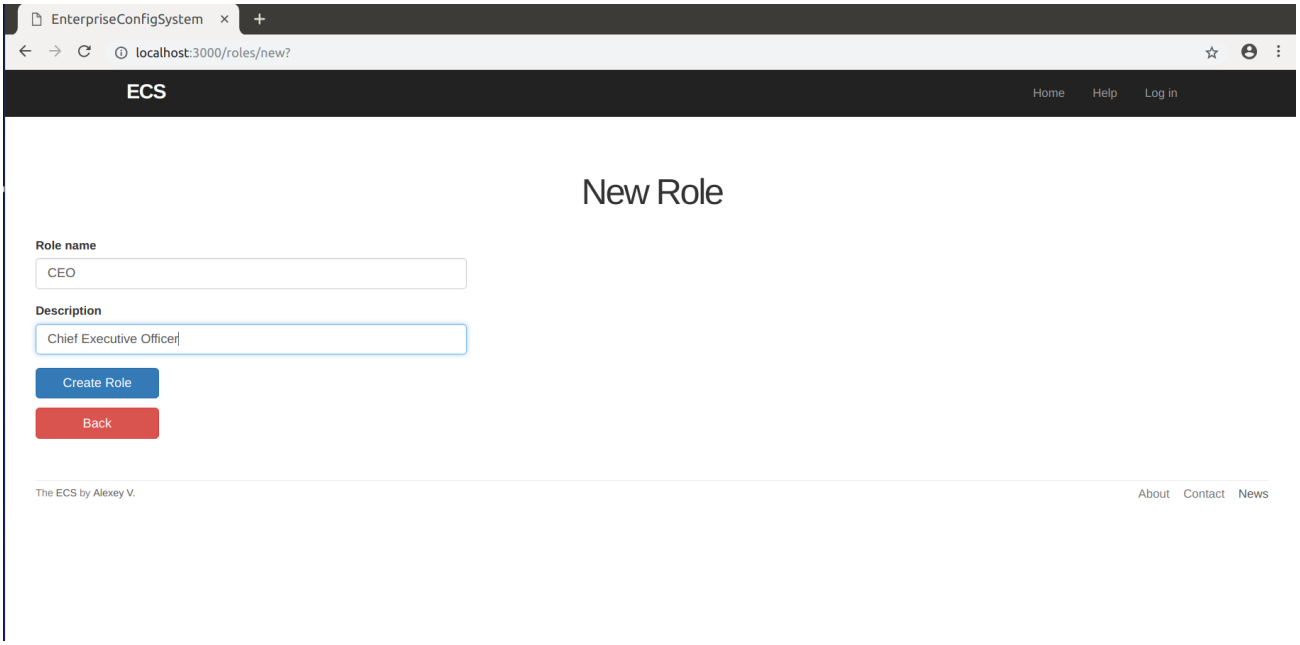


Рисунок 11. Створення нової ролі

Так само потрібно заповнити даними і інші таблиці. Після створених ролей для співробітників, можна додавати самих співробітників. Потім потрібно поповнити базу устаткування підприємства, а також додати всі кімнати, офіси, цехи та інши приміщення.

Заповнивши масиви даних, система отримує можливість слідкувати за потребами підприємства, кількістю устаткування для активних співробітників користувачів. Положення співробітників, а також положення устаткування теж можливо відслідити. А автоматизовані системи, які успішно інтегровані до системи управління підприємством, можуть бути налаштовані з центральної консолі без необхідності прямого втручання. Моніторинг параматрів теж можливо робити з центрального інтерфейсу користувача

3.5 Висновки

Сучасний ринок веб-додатків дає великий вибір технологій та можливостей реалізації своїх рішень. В рамках дипломного проекту для реалізації системи налаштування і управління підприємством був вибран досить відомий стек технологій Ruby on Rails.

Основна мова програмування в данній технологіх це Ruby, також даний стек має інтегрованні технології БД Active Record, який дозволяє взаємодіяти з різними провайдерами БД (MySQL, Postgresql) та створювати об’єктну реляційну модель в додатку. Для розробки дизайну Ruby on Rails має інтегрованні інструменти для роботи з html, css і javascript.

Розділ 4. Інтеграція системи прогнозування температури навколішнього середовища з системою управління і налашатування підприємства

4.1 Проектування інтеграціїї системи

Система буде складатися з декількох підсистем. Найбільш підходящою архітектурою для системи є клієнт-серверна архітектура, цей підхід дозволить розмежити систему на централізований сервер який отримує всі дані, зберігає їх у базі даних та використовує ці дані в алгоритмі машинного навчання. Клієнтська частина в свою чергу буде складатися з моніторингового пристрою(їв). Сервер зможе одночасно обробляти декілька пристроїв-клієнтів, та збирати достатньо статистичних даних для алгоритму машинного навчання, базуючись на різних локаціях або на різних вимірюваних величинах. Всі розрахунки буде виконувати сервер та виводити результати роботи штучної нейронної мережі у вигляді графіків на екран монітору.

Загальный вид системи можна описати такою схемою:

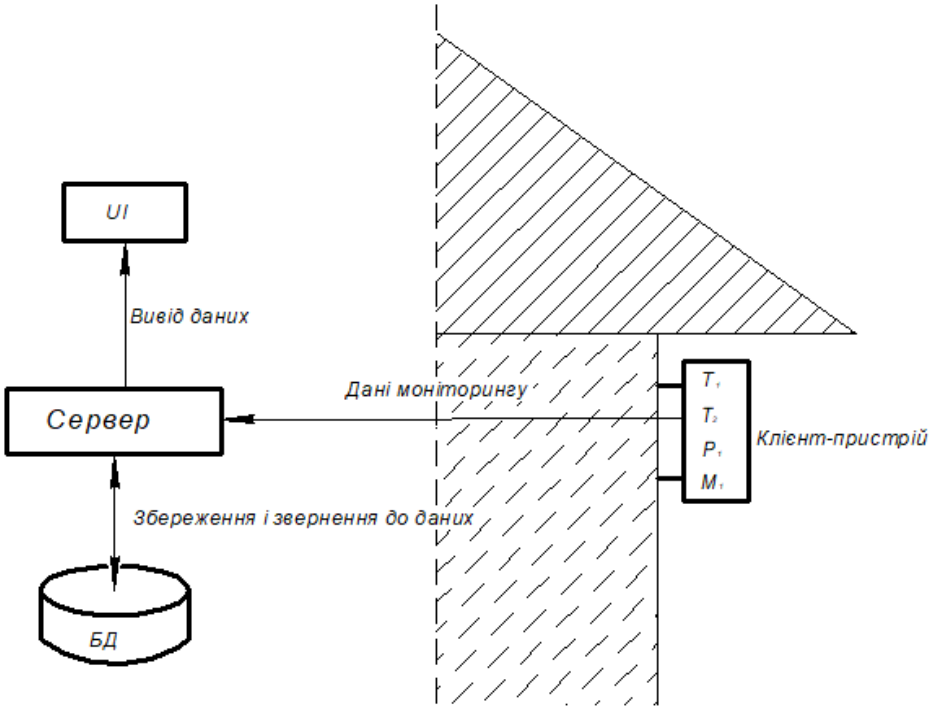


Схема 1. Схематичне зображення системи

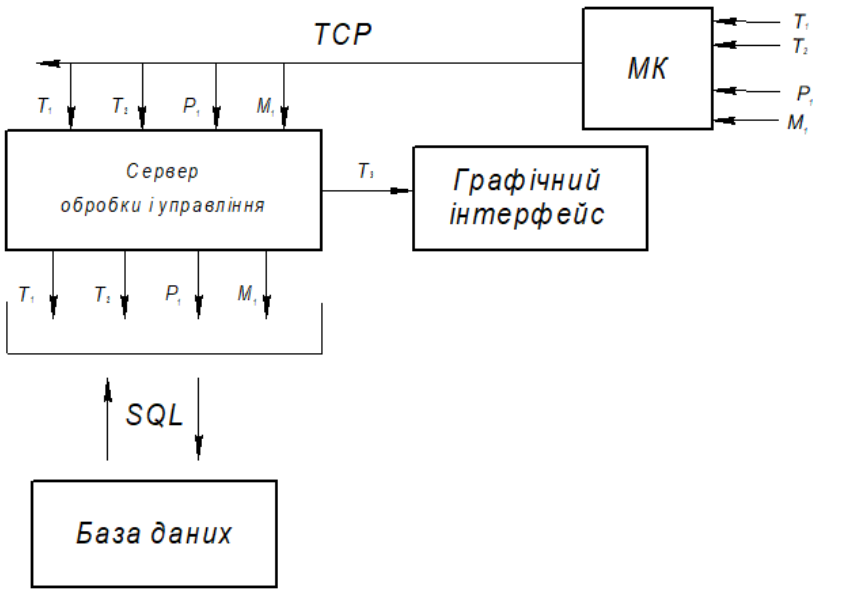
Серверна частина має покривати декілька аспектів системи: виділенний сервер для отримання результатів з пристроїв клієнтів, “спілкуватися” з базою даних, виконувати розрахунки для прогнозування відповідно до алгоритми штучної нейронної мережі.

Клієнтський пристрій має задоволняти таким критеріям: мати можливість збирати характеристики середи за допомогою давачів, інтерфейс для можливості організації підключення клієнта до сервера, можливість швидкого перепрограмування пристрою на інші види характеристик або зміни конфігурації з’єднань з сервером або внутрішніх компонентів.

База даних має підтримувати мову структурованих запитів SQL для можливості маніпулювання даних в контексті роботи системи, також підтримка функції backup, для відновлення даних у базі даних, якщо дані будуть пошкодженні внаслідок будь-яких подій.

Графічний інтерфейс користувача буде виводити результати роботи системи на екран монітору.

Побудуємо схему матеріально-інформаційних потоків системи:



T_1	Температура повітря давач 1
T_2	Температура повітря давач 2
P_1	Тиск повітря давач 2
M_1	Вологість повітря давач 1

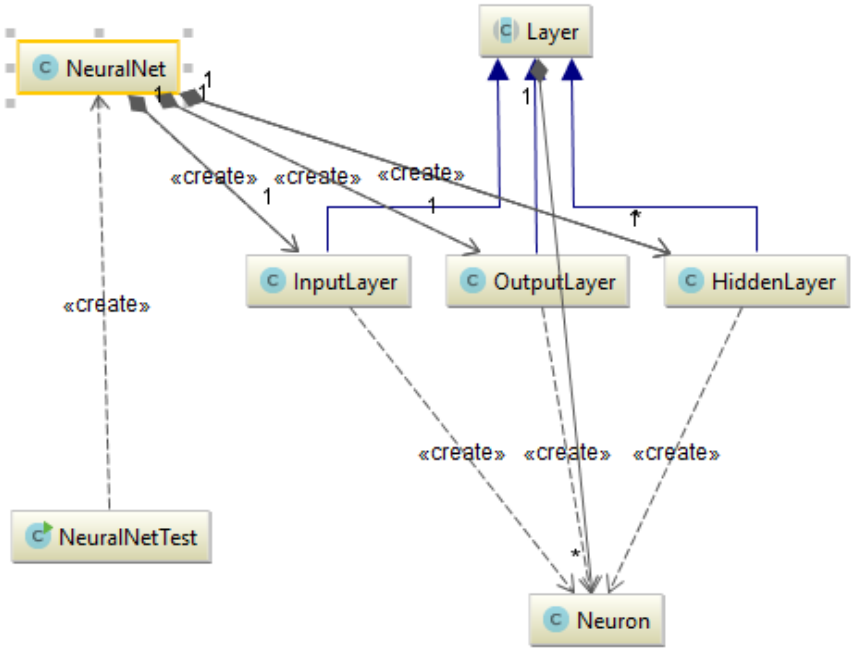
Схема 2. Схема інформаційно-матеріальних потоків

Як видно на схемі я вибрав такі параметри для навчання системи як: температура повітря від двох незалежних давачів, вологість та тиск повітря. Ці параметри були вибрані тому що давачі таких типів мають невисоку вартість та досить доступні і зрозумілі у використані.

Загальний цикл роботи системи можна описати такою послідовністю:

1. Пристрій або пристрої-клієнти збирають інформацію щодо середи в якій вони розташовані та відсилають всю інформацію серверу.
2. Сервер приймає всі дані з пристроїв клієнтів та зберігає їх у базі даних.
3. Коли даних достатньо для навчання штучної нейронної мережі, окремий процес запускає навчання мережі на більшості отриманих даних, а останні отримані виступають в ролі даних для тестування мережі для отримання прогнозуючих параметрів.

Для реалізації штучної нейронної мережі потрібно описати основні сутності мережі у вигляді классів в межах основної програми. Взаємозв'язок сутностей можна побачити на наступній UML-діаграммі:



Діаграма 3. UML діаграмма класів нейронної мережі

Після визначення основних сутностей потрібно визначити послідовність обробки даних. Відповідно до схем проектування у другому розділі система буде надсилати 4

параметри для обробки нейронною мережею. Це означає що наша система складатиметься з 3 прошарків:

1. Вхідний прошарок в якому буде чотири нейрони для кожного з параметрів
2. Внутрішній прошарок з чотирьма нейронами для визначення коефіцієнтів ваги для кожного з параметрів
3. Вихідний прошарок з одним вихідним нейроном для прогнозованного параметру.

Також програма потребує додаткових утилітарних класів для нормалізації даних, поділення даних на навчанльні і тестувальні, а також бібліотека для побудови графіків, наприклад використаємо jfreechart. Лістинг кінцевого виду програми наведений у додатку А.

4.2 Клієнтська частина системи прогнозування

Аппаратні засоби

Клієнт-пристрій буде побудован на базі мікроконтроллеру atmega328 у складі плати типу Arduino Uno. Вибір цієї плати обумовленний достатньою кількостю наявних портів входу виходу, загальною доступністю великої кількості готових бібліотек для роботи з платою та різноманітністю доступних модулів для розширення плати.

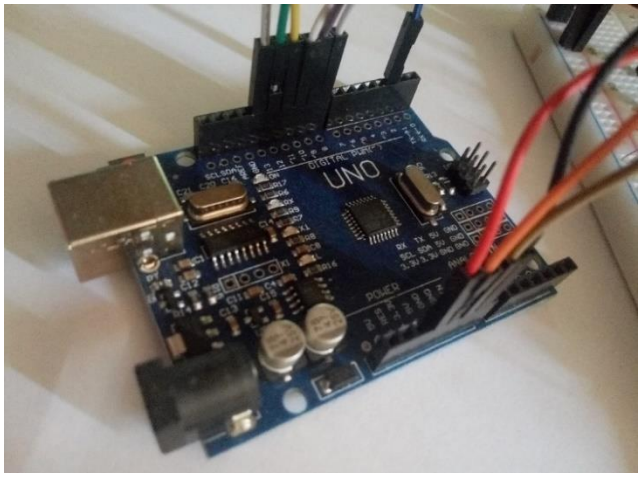


Рисунок 12. Плата Arduino Uno.

Характеристики:

- Мікроконтролер ATmega328
- робоча напруга 5 В
- Вхідна напруга (рекомендований) 7-12 В
- Вхідна напруга (граничне) 6-20 В
- Цифрові Входи / Виходи 14 (6 з яких можуть використовуватися як виходи ШІМ)
- аналогові входи 6
- Постійний струм через вхід / вихід 40 мА
- Постійний струм для виведення 3.3 В 50 мА
- Флеш пам `ять 32 Кб (ATmega328) з яких 0.5 КБ використовуються для завантажувача
- ОЗУ 2 КБ (ATmega328)
- EEPROM 1 Кб (ATmega328)
- Тактова частота 16 МГц

Живлення

Arduino Uno може отримувати живлення через підключення USB або від зовнішнього джерела живлення. Джерело живлення вибирається автоматично. Зовнішнє живлення (НЕ USB) може подаватися через перетворювач напруги AC / DC (блок живлення) або акумуляторною батареєю. Перетворювач напруги підключається за допомогою роз'єму 2.1 мм з центральним позитивним полюсом. Провід від батареї підключаються до висновків Gnd і Vin роз'єму живлення. Платформа може працювати при зовнішньому живленні від 6 В до 20 В. При напрузі живлення нижче 7 В, вихід 5V може видавати менше 5 В, при цьому платформа може працювати нестабільно. При використанні напруги вище 12 В регулятор напруги може перегрітися і пошкодити плату. Рекомендований діапазон від 7 В до 12 В.

Роз’єми живлення:

VIN. Вхід використовується для подачі живлення від зовнішнього джерела (за відсутності 5 В від роз'єму USB або іншого регульованого джерела живлення). Подача напруги живлення відбувається через даний роз’єм. 5V. Регульований джерело напруги, що використовується для живлення мікроконтролера і компонентів на платі. Живлення може подаватися від виведення VIN через регулятор напруги, або від роз'єму USB, або іншого регульованого джерела напруги 5 В.

3V3. Напруга на виводі 3.3 В генерується вбудованим регулятором на платі. Максимальне споживання струму 50 мА.

Входи і Виходи

Кожен з 14 цифрових виходів Uno може налаштований як вхід або вихід, використовуючи функції pinMode (), digitalWrite (), і digitalRead (),. Виходи працюють при напрузі 5 В. Кожен вихід має навантажувальний резистор (за замовчуванням відключений) 20-50 кОм і може пропускати до 40 мА. Деякі виходи мають особливі функції:

1. Послідовна шина: 0 (RX) і 1 (TX). Виходи використовуються для отримання (RX) і передачі (TX) даних TTL. Дані виходи підключені до відповідних виходів мікросхеми послідовної шини ATmega8U2 USB-to-TTL.

2. Зовнішнє переривання: 2 і 3. Дані виходи можуть бути налаштовані на виклик переривання або на молодшому значенні, або на передньому чи задньому фронті, або при зміні значення. Детальна інформація знаходиться в описі функції attachInterrupt ().

3. ШИМ: 3, 5, 6, 9, 10, і 11. Будь-який з виходів забезпечує ШИМ з роздільною здатністю 8 біт за допомогою функції analogWrite ().

4. SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). За допомогою даних виходів здійснюється зв'язок SPI, для чого використовується бібліотека SPI.

5. LED: 13. Вбудований світлодіод, підключений до цифрового виходу 13. Якщо значення на виведення має високий потенціал, то світлодіод горить.

На платформі Uno встановлені 6 аналогових входів (позначених як A0 .. A5), кожен дозволом 10 біт (тобто може приймати 1 024 різних значення). Стандартно виходи мають діапазон вимірювання до 5 В відносно землі, проте є можливість змінити верхню межу за

допомогою виведення AREF і функції analogReference (). Деякі виходи мають додаткові функції:

- I2C: 4 (SDA) і 5 (SCL). За допомогою виходів здійснюється зв'язок I2C (TWI), для створення якої використовується бібліотека Wire.

Додаткова пара виходів платформи:

- AREF. Опорна напруга для аналогових входів. Використовується з функцією analogReference ().
- Reset. Низький рівень сигналу на виводі перезавантажує мікроконтролер. Зазвичай застосовується для підключення кнопки перезавантаження на платні розширення, що закриває доступ до кнопки на самій платі Arduino.

Зв'язок

На платформі Arduino Uno встановлено кілька пристроїв для здійснення зв'язку з комп'ютером, іншими пристроями Arduino або микроконтроллерами. ATmega328 підтримують послідовний інтерфейс UART TTL (5 В), здійснюваний виходами 0 (RX) і 1 (TX). Встановлена на платі мікросхема ATmega8U2 направляє даний інтерфейс через USB, програми на стороні комп'ютера "спілкуються" з платою через віртуальний COM порт. Прошивка ATmega8U2 використовує стандартні драйвера USB COM, ніяких стороніх драйверів не потрібно, але на Windows для підключення потрібно файл ArduinoUNO.inf. Моніторинг послідовної шини (Serial Monitor) програми Arduino дозволяє посилати і отримувати текстові дані при підключенні до платформи. Світлодіоди RX і TX на платформі будуть мигати при передачі даних через мікросхему FTDI або USB підключення (але не при використанні послідовної передачі через виходи 0 і 1).

Бібліотекою SoftwareSerial можливо створити послідовну передачу даних через будь-який з цифрових виходів Uno. ATmega328 підтримує інтерфейси I2C (TWI) і SPI. В Arduino включена бібліотека Wire для зручності використання шини I2C.

Програмування

Платформа програмується за допомогою ПЗ Arduino. З меню Tools> Board вибирається «Arduino Uno» (згідно зі встановленим микроконтроллеру). Детальна інформація знаходиться в довіднику і інструкціях. Мікроконтролер ATmega328 поставляється з записаним завантажувачем, що полегшує запис нових програм без

використання зовнішніх программаторов. Зв'язок здійснюється оригінальним протоколом STK500. Є можливість не використовувати завантажувач і запрограмувати мікроконтролер через висновки ICSP (внутрішньосхемне програмування). Детальна інформація знаходиться в даній інструкції.

Давачі

Для отримання характеристик навколишнього середовища клієнт-пристрій має моніторити середу за допомогою давачів. Для навчання штучної нейронної мережі, я обрав такі характеристики середи: температура, вологість повітря, тиск повітря, індекс нагріву.

Для отримання температурних даних і даних вологості повітря я обрав цифровий давач DHT22, він має малу похибку порівняно з своїм попередником DHT11 і одразу покриває вимірювання температури, вологості і індексу нагріву.

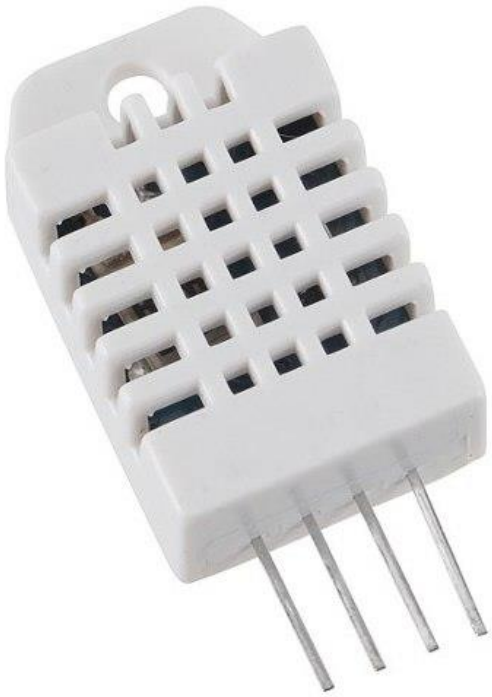


Рисунок 13. Давач DHT22.

Контакт з ліва на право:

1. VCC
2. DATA
3. Не використовується

4. GND

Характеристики давача температури і вологості DHT22:

- модель AM2303
- Джерело живлення 3.3-6V DC
- Вихідний сигнал цифрового
- Чутливий елемент вимір RH - полімерний конденсатор
- вимір температури - на базі чіпа DS18B20
- Вимірювання вологості 0-100% з похибкою ± 2%
- Збільшення похибки ± 0.5% / рік
- Вологість гистерезиса ± 0.3%
- Вимірювання температури -40 ... + 125 ° C, похибка ± 0.5 ° C
- Взаємозамінність повністю замінні
- Розміри 25.1 x 15.1 x 7.7 мм
- Вага 2.2 гр.

Для отримання тиску повітря обрано цифровий давач тиску bmp280 від фірми Bosch, також цей давач вимірює температури, тому для більш точніших даних система буде брати усереднене значення обох давачів bmp280 і dht22.

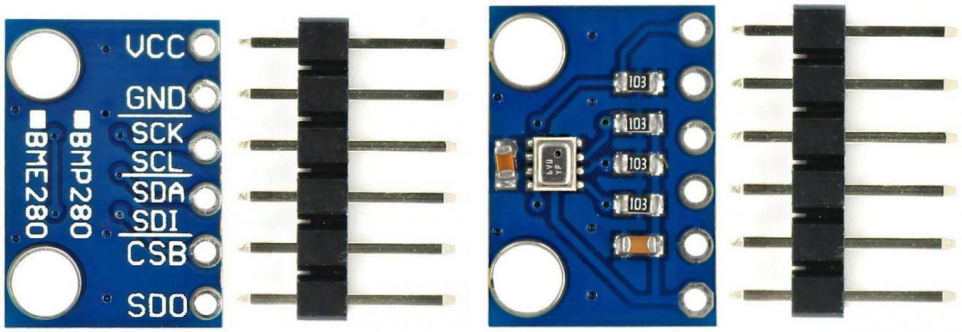


Рисунок 14. Давач bmp280

Модуль вимірювання атмосферного тиску на BMP-280. Цей датчик є покращеною версією датчика BMP180 і відрізняється від нього меншими розмірами, зниженим енергоспоживанням, високою точністю роботи і наявністю точної заводського калібрування і двома послідовними інтерфейсами: I2C і SPI.

Режими роботи:

- SLEEP - режим зниженого енергоспоживання
- FORCED - режим, аналогічний, режиму роботи датчиків BMP085 і BMP180. За командою контролера датчик виходить з режиму сну, проводить вимірювання, видає результати вимірювання контролера і переходить в режим зниженого енергоспоживання
- NORMAL - унікальний для цього датчика режим. Датчик самостійно прокидається, проводить вимірювання тиску і температури і засинає. Всі тимчасові параметри цього режиму програмуються незалежно. Зчитувати дані в цьому режимі можна в будь-який час.

Характеристики:

- Напруга живлення: від 1.71 В до 3.6 В
- Макс швидкість I2C інтерфейсу: 3.4 МГц
- Струм: 2.7мкА при частоті відліків в 1 Гц
- Інтерфейс: I2C, SPI (4 дроти), SPI (3 дроти)
- Калібрування: заводська
- Рівень шуму: до 0.2 Па (1.7 см) і 0.01 температури
- Діапазон вимірюваного тиску: від 300hPa до 1100hPa (9000 м до -500 м)
- Розмір: 21 мм х 18 мм

Також для зв’язку клієнта-пристрою з серверву потрібен модуль бездротового зв’язку. Для вирішення цієї задачі був обранний Wi-Fi модуль ESP8266 версія ESP-01, це найбільш

примітивна з версій лінійки цих пристроїв, але вона покриває всі потреби системи для зв’язку з сервером.

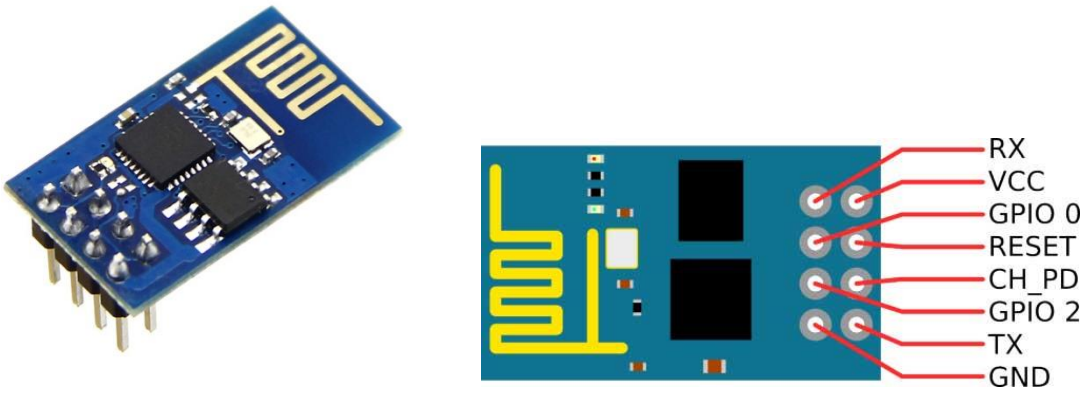


Рисунок 15. Wi-Fi модуль ESP8266 версія ESP-01

Мініатюрний WiFi модуль на базі новітньої мікросхеми ESP8266 з вбудованим стеком протоколу TCP / IP і управлінням AT-командами. Чіп створений для використання в розумних розетках, mesh-мережах, IP-камерах, бездротових сенсорах, що носиться електроніці і так далі.

Передбачено два варіанти використання чіпа: 1) у вигляді моста UART-WIFI, коли модуль на базі ESP8266 підключається до існуючого рішенням на базі будь-якого іншого мікроконтролера і управляється AT-командами, забезпечуючи зв'язок рішення з інфраструктурою Wi-Fi; 2) реалізуючи нове рішення, яке використовує сам чіп ESP8266 в якості керуючого мікроконтролера.

Характеристики:

- підтримка WiFi протоколів 802.11 b / g / n
- Wi-Fi Direct (P2P), soft-AP
- вбудований стек TCP / IP
- вбудований TR перемикач, balun, LNA, підсилювач потужності і відповідність мережі
- вбудований PLL, регулятори, і система управління живленням
- вихідна потужність +20.5 дБм в режимі 802.11b

- підтримка зовнішніх антен
- струм витоку в вимкненому стані до 10 мкА
- SDIO 2.0, SPI, UART
- STBC, 1x1 MIMO, 2x1 MIMO
- A-MPDU і A-MSDU aggregation і 0.4мс guard interval
- пробудження і посил пакетів до 22 мс
- споживання в режимі Standby до 1.0 мВт (DTIM3)
- розміри: 24.5x14 мм

Обрав всі необхідні елементи можна описати схему підключення і функціональну схему клієнта-пристрою:

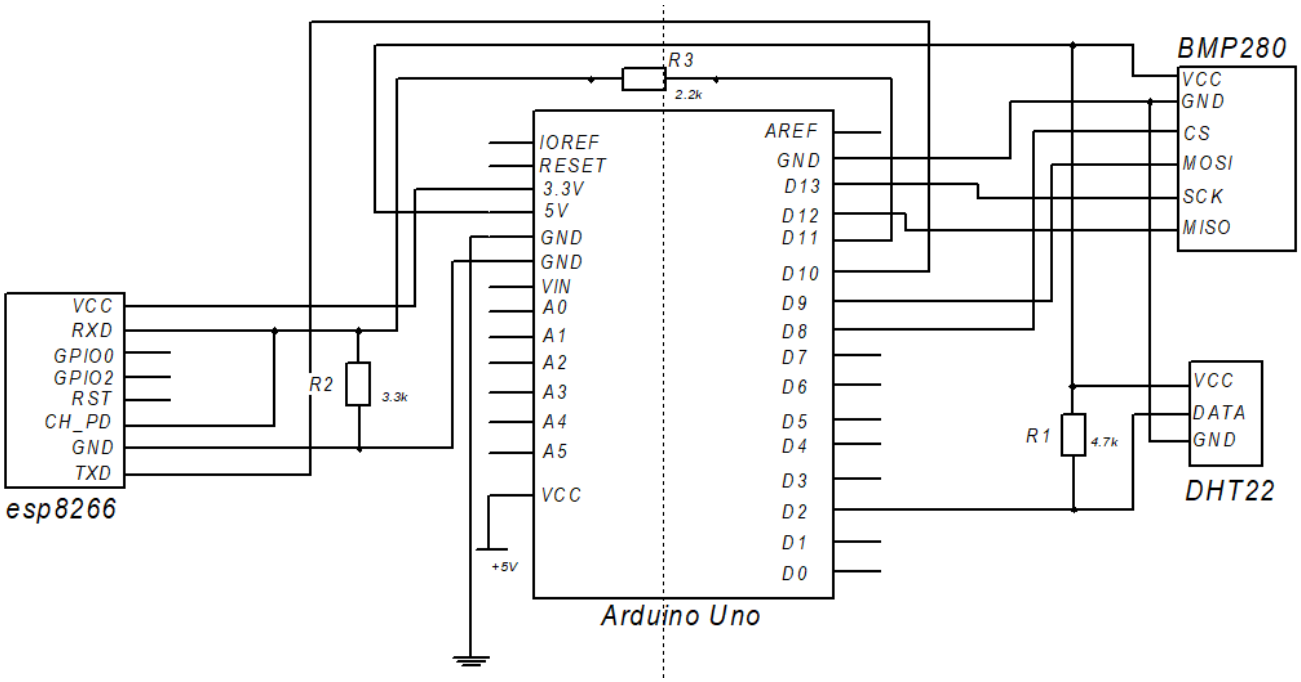


Схема 3. Схема підключення клієнта-пристрою

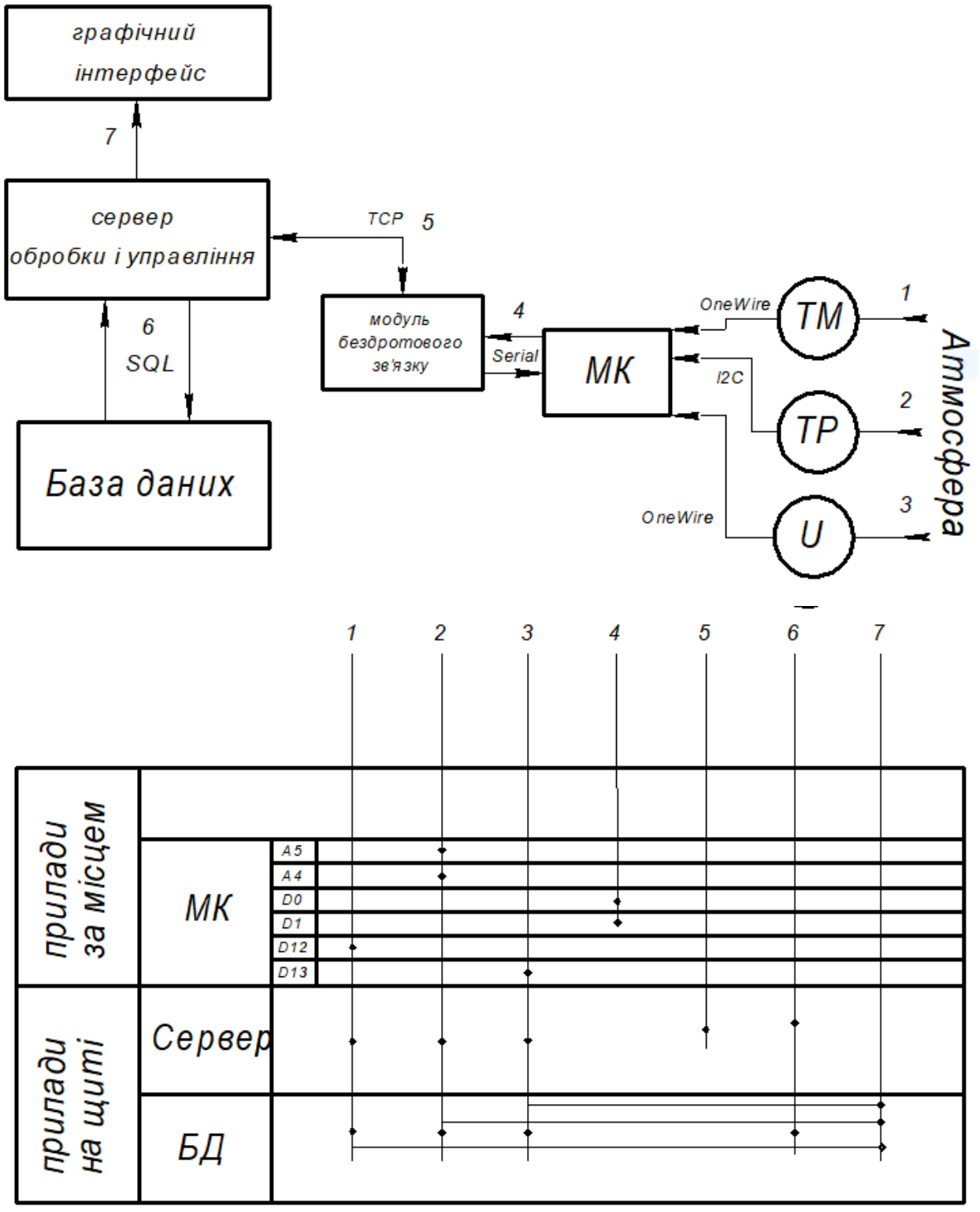


Схема 4. Функціональна схема клієнта-пристою

4.3 Програмна частина

Програмна частина с кладатиметься з коду на мові програмування С для плати Arduino, для цього використаємо ПЗ Arduino IDE та бібліотеки для обранних пристроїв:

- Software Serial – бібліотека для реалізації серійних портів за допомогою цифрових виходів плати.
- DHT lib – бібліотека для спрощення програмного коду взаємодії між платою та давачем серії DHT.
- ESP8266 lib – бібліотека для керування WI-FI модулем ESP8266.
- Adafruit_Sensor та Adafruit_BMP280 – бібліотеки для спрощення програмного коду для давача BMP280.
- ArduinoJson – бібліотека для агрегації результатів вимірювання давачей та пересилання даних до серверу у форматі JSON.

Загалом код клієнта розбитий на дві частини це ініціалізація та цикл роботи. Під час ініціалізації пристрій буде підключатися до давачів та модулю бездротового зв'язку і потім до самого серверу. В основному циклі пристрій буде зчитувати показники давачів, формувати JSON та відправляти на сервер для його обробки.

4.4 Графічне представлення налаштування системи та контроль

Для початку потрібно підключити систему прогнозування до основної системи, для цього скористаємося сторінкою налаштування системи:

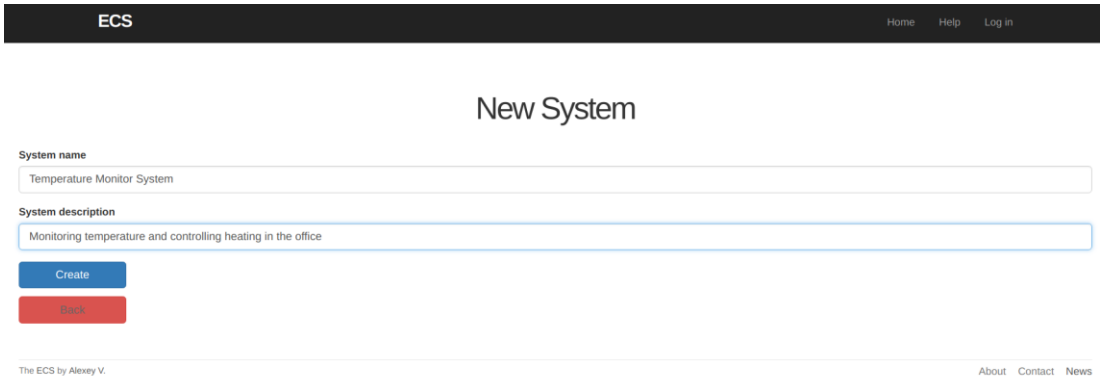


Рисунок 16. Підключення системи прогнозування

Для подальшого управління системою потрібно імлеметувати графічний інтерфейс для представлення даних і управління на основі цих данних.

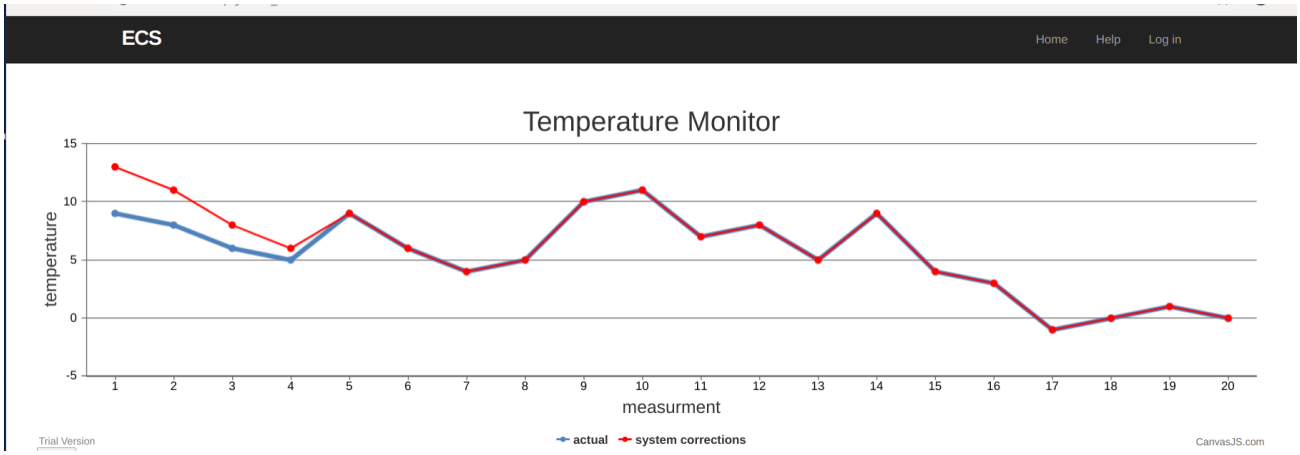


Рисунок 17. Графічне представлення даних

На данний момент система задає температуру відповідно до отриманої від системи прогнозування. Для розширення контролю пропонуються додати можливість додати зміщення температури від прогнозованної та константне задання температури:

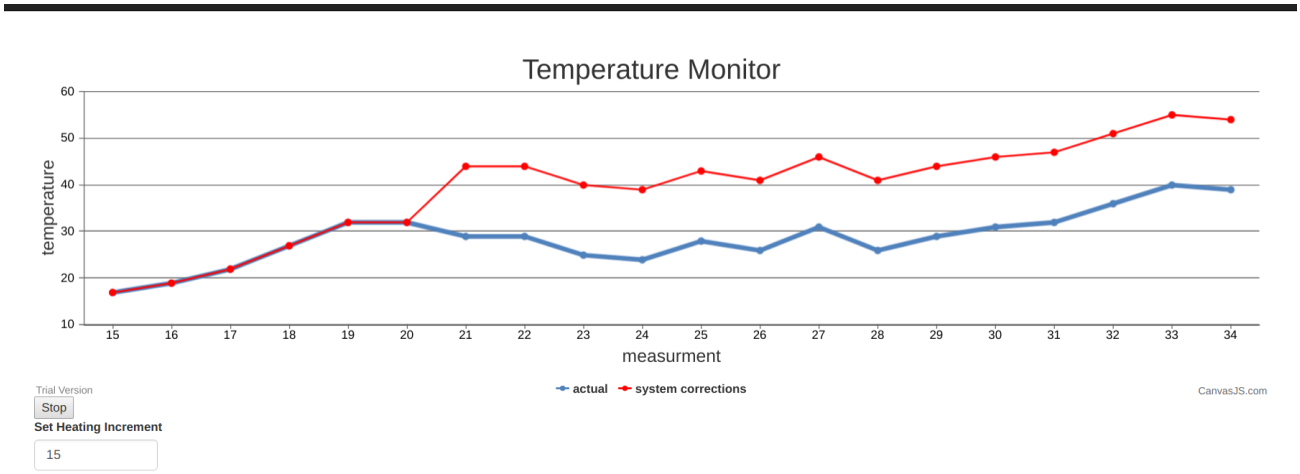


Рисунок 18. Реалізація контролю зміщення температури

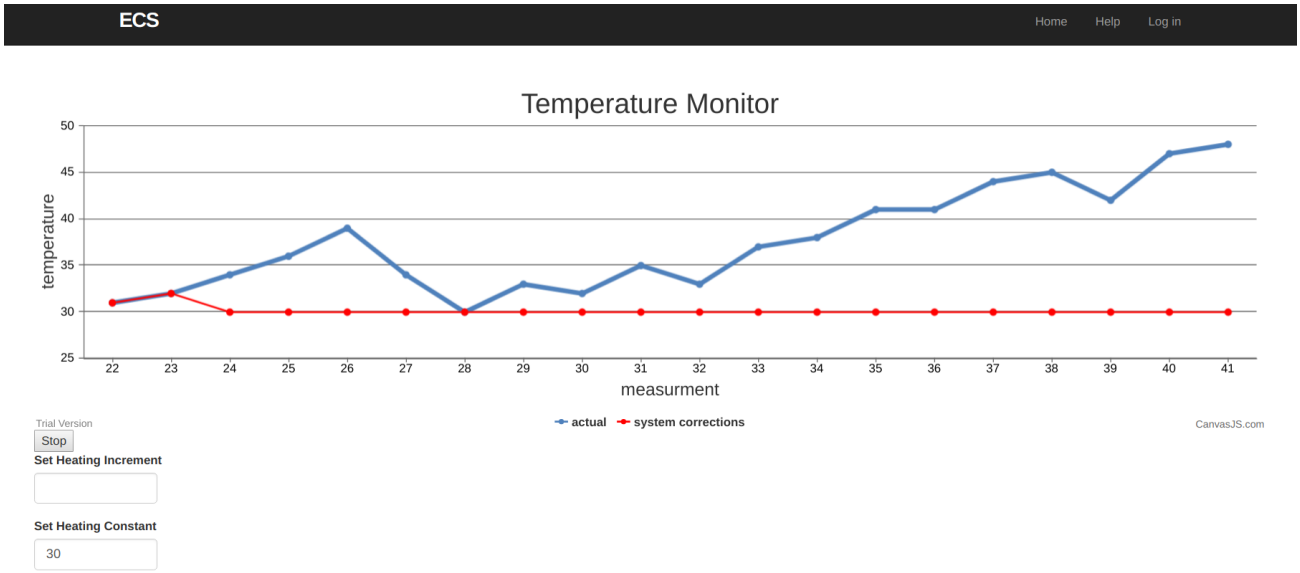


Рисунок 19. Реалізація контролю постійного значення

Як ми бачимо, тепер інтерфейс користувача дозволяє керувати температурою залежно від показань системи прогнозування так і задавати постійне значення. Кількість систем яку можна інтегрувати до системи налаштування і керування підприємстовм є теоритично необмежена.

4.5 Висновки

В даному розділі були розглянуто реалізацію серверної частини системи, клієнтської частини системи, аналіз роботи системи. Створення основних підсистем, серверної і клієнтської, було поділено на дві частини. Було розглянуто апаратні засоби реалізації систем, тобто складові елементи підсистем, обгрунтування вибору, взаємодія. Інша частина присвячена програмній оболонці кожної з підсистем та реалізація прогрманого коду для них.

ВИСНОВКИ

Дипломна робота присвячена актуальній задачі управління і налаштування підприємства за допомогою комп'ютерно інтегрованних систем. Ця задача вважається актуальною, тому що підрпиємства зростають у розмірах та запроваджують велику кількість автоматизованих систем за якими потрібен централізований моніторинг.

У результаті виконання даної дипломної роботи було розроблено систему, за допомогою якої можна дослідити повний цикл розробки і роботи системи управління і налаштування підприємства, починаючи з архітектури додатку закінчуючи інтеграцією додатку з іншими системами.

В контексті роботи використовувався досить широкий спектр технологічних інструментів. Були використані такі мови програмування як Ruby, Java та JavaScript. Використана база даних PostgreSQL. Наведені схеми системи, які створення під час проектування. Реалізований алгоритм прогнозування температури на базі машинного навчвання.

Згідно аналізу результатів роботи системи вона виконує свої функціїї управління підпорядкованими системами і управління персоналом та його устаткуванням. Також були виявленні моделі поведінки системи залежно від типу даних які надходять до неї у вигляді тренувального масиву.

У першому розділі описано про особливості та проблематики систем управління підприємства, були розглянуті існуючі аналоги таких видів систем. Також проаналізовані вітчизняні аналоги систем та зарубіжні моделі систем. Можна зробити висновок, що ринок має невеликий вибір таких систем, і більшість підприємств обирають відомі компанії такі як Oracle або Microsoft. Тому розробка легковісного додатку може відкрити можливості використання автоматизованих систем управління для середніх та малих підприємств.

У другому розділі більш детально описанні кроки для реалізації таких систем. Детально розглянуті види систем та сфера їх використання, інструменти за допомогою яких ці системи можливо реалізувати.

Третій розділ повністю присвячений створенню системи управління і налаштування підприємства. Детально описанні серверна та клієнтська частина системи, аппаратні та

програмні засоби для їх реалізації. Також проаналізовані результати роботи системи з різними видами даних. Зроблені висновки, що модель системи працює коректно.

У четвертому розділі розглянута реалізація інтеграції системи з системою прогнозування температури для автоматизації контролю температурного режиму на підприємстві. Наданий графічний інтерфейс системи та приклади роботи системи у декількох режимах.

СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Штучні нейронні мережі. [Електронний ресурс] — Режим доступу: http://www.victoria.lviv.ua/html/oio/html/theme5.htm. — Дата доступу: 1.10.18.
2. Прогнозування за допомогою нейронних мереж. [Електронний ресурс] — Режим доступу: http://wiki.tntu.edu.ua/Прогнозування_за_допомогою_нейронних_мереж. — Дата доступу: 30.10.18.
3. Что такое ERP система — Режим доступу: https://habr.com/company/trinion/blog/333018/ [Електронний ресурс] — Дата доступу: 2.10.18.
4. ERP-система (планування ресурсів підприємства) система — Режим доступу: https://pidruchniki.com/1171062647760/informatika/erp-sistema_planuvannya_resursiv_pidpriyemstva — Дата доступу: 3.10.18.
5. Что такое ERP система и для чего она нужна? [Електронний ресурс] — Режим доступу: https://sitis.com.ua/about/articles/chto_takoe_erp_sistema_i_dlya_chego_ona_nuzhna — Дата доступу: 5.10.18.
6. Клієнт-серверна архітектура та ролі серверів [Електронний ресурс] — Режим доступу: https://medium.com/@IvanZmerzlyi/— Дата доступу: 4.10.18.
7. Документація фреймворку Ruby on Rails [Електронний ресурс] — Режим доступу: https://rubyonrails.org/ ; Дата візиту – 3.10.18
8. Alan M. F. Souza, Fabio M. Soares Neural Network Programming with Java / Alan M. F. Souza // Fabio M. Soares California, 13 January, 2016 p. 244
9. Laurene V. Fausett Fundamentals of neural networks / Laurene V. Fausett December 9, 1994 p.449
10. Анджей Пегат Нечеткое моделирование и управление / Анджей Пегат. БИНОМ. Лаборатория знаний, Москва 2014. — 798 с.
11. Ameet Talwalkar, Afshin Rostamizadeh, Mehryar Mohri Foundations of Machine Learning / Ameet Talwalkar // Afshin Rostamizadeh // Mehryar Mohri, August 2014 p.432
12. Interactive Online SQL Training [Електронний ресурс] – Електронні дані. Режим доступу: http://www.sqlcourse.com/intro.html ; Дата доступу: 5.10.18.