МІНІСТЕРСТОВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри КІТ
_____ А. С. Савченко
«____» _____ 2020 р.

ДИПЛОМНИЙ РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»
ЗА НАПРЯМОМ 122 «КОМП'ЮТЕРНІ НАУКИ»

Тема: «Метод діагностики розподіленої мережної системи»

Виконавець: студент УС-211М Кульбіт Дмитро Сергійович
(студент, група, призвіще, ім'я, по батькові)

Керівник: доктор технічних наук, професор ВІНОГРАДОВ Микола Анатолійович
(науковій ступень, вчене звання, призвіще, ім'я, по батькові)

Нормоконтролер _____ Райчев І. Е.
(підпис) (П.І.Б.)

КИЇВ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних інформаційних технологій
Освітній ступінь **Магістр**
Напрям 122 Комп'ютерні науки
(шифр, найменування)

ЗАТВЕРДЖУЮ
Завідувач випускової кафедри

_____А. С. Савченко
« » _____2020 р.

ЗАВДАННЯ
на виконання дипломної роботи студента
Кульбіта Дмитра Сергійовича
(прізвище, ім'я, по батькові)

1.Тема роботи: «Методи автоматизації виробництва меблів » затверджена наказом ректора № 567/ст. від 16.03.2020р.

2.Термін виконання роботи: з 06.05.2020 по 28.06.2020р.

3. Вихідні дані до роботи: розробка ПО на базі чпу верстатату ROVER A 3.30 ATS для створення корпусних меблів.

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці): вступ, аналітичний огляд і постановка завдання, web додаток для створення корпусних меблів, висновок.

5. Перелік обов'язкового матеріалу: огляд загальної структури **G-код систем,** архитектра верстату ROVER A 3.30 ATS, схема загального алгоритму роботи **G-код систем.**

6. КАЛЕНДАРНИЙ ПЛАН

	Етапи виконання дипломного проекту	Термін виконання етапів	Примітка
1	Аналіз літератури та джерел за темою дипломної роботи.		
2	Розробка та затвердження плану дипломної роботи.		
3	Проведення консультації з науковим керівником щодо створення першого розділу.		
4	Аналітичний огляд і постановка задачі.		
5	Порівняльний аналіз існуючих **G-код систем**		
6	Створення web додатку для для проектування меблевих, корпусних виробів		
7	Висновки та оформлення пояснювальної записки дипломної роботи.		
8	Підписання необхідних документів у встановленому порядку.	27.01.2020	
9	Підготовка до захисту та попередній захист дипломного роботи на випусковій кафедрі дипломної роботи	28.01.2020	

7. Дата видачі завдання: 06.05.2020 р.

Керівник дипломної роботи _____ Віноградов М.А.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Іванова О.А
(підпис випускника) (П.І.Б.)

Реферат

Пояснювальна записка до дипломного проекту «Методи автоматизації виробництва меблів»

Ключові слова: G-КОД СИСТЕМИ, ЧПУ ВЕРСТАТ, КОРПУСНІ МЕБЛІ, ROVER A 3.30 ATS, САПР МЕБЛІВ .

Об’єкт дослідження: автоматизація виробництва меблів.

Предмет дослідження: ЧПУ верстат ROVER A 3.30 ATS

Мета роботи: розробка web додатку для проектування корпусних меблів на базі ЧПУ верстату ROVER A 3.30 ATS.

Методи дослідження, технічні та програмні засоби: розробка програмних модулів, обробка літературних джерел.

Отримані результати та їх новизна:

Матеріали проекту можуть бути використані автоматизації меблевого виробництва.

Зміст

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

ІС – Інформаційна система

ВСТУП

Сучасні види послуг в світі та зокрема в Україні продовжують удосконалюватися. З розвитком науково-технічного прогресу постійно з'являються нові послуги. Кур'єрські послуги займають помітне місце у виробничій та соціальній інфраструктурі, оскільки є її невід'ємною частиною. У сфері соціального забезпечення населення послуги з експрес-доставки дають можливість користуватися надійними засобами зв'язку для пересилання різного роду відправлень, отримання і обміну інформацією. У сфері суспільного виробництва товарів кур'єрські послуги забезпечують взаємозв'язок між учасниками виробничих операцій та доведення продукту до споживача. У сфері управління послуги забезпечують доставку різного роду документів управлінських структур, як до юридичних, так і до фізичних осіб.

Експрес-перевезення – це діяльність, пов'язана з наданням послуг і робіт, що забезпечують вивезення, транспортування і доставку за схемою «від дверей до дверей» документів та вантажів в строго обмежені за часом терміни. Звільняючи вантажовласника від всіх турбот з організації перевезення, вибором виду транспорту і схем доставки вантажу експрес-перевізник несе перед ним відповідальність за виконання перевезення в цілому і окремо за вантаж з моменту його прийняття в своє розпорядження і до моменту видачі одержувачу.

Основні відмінності кур'єрської служби наступні:

• кур’єри оперативно забирають у відправника посилку. Цей строк залежить від терміновості доставки, на яку розраховує відправник і становить зазвичай від 1 години до кінця наступного дня. У випадку звичайної пошти, відправнику самому довелося б нести свій лист або бандероль в поштове відділення;

• під час доставки кур'єрськими службами використовується спеціальна фірмова упаковка, зазвичай у вигляді щільних конвертів або поліетиленових

пакетів, яка оберігає листи та іншу кореспонденцію від намокання, забруднення та псування;

• час доставки кур'єрської служби чітко обумовлений із замовником і тому несвоєчасність доставки вкрай рідкісна;

• на всіх етапах доставки кур'єри збирають з посадових осіб підписи в документах, що підвищує рівень відповідальності за вантаж;

• після здійснення доставки кур'єр зазвичай відразу зв'язується з представником своєї кур'єрської служби і повідомляє про це. Відразу ж після цього представник компанії зв'язується із замовником і сповіщає про те, куди доставлена кореспонденція, коли та хто прийняв.

Основними споживачами послуг є компанії різних сфер бізнесу. Згідно з даними досліджень, найбільш важливими факторами при замовленні пересилання стають: вартість послуг, терміни доставки, досвід і репутація фірми-виконавця. Послугами користуються в основному корпоративні клієнти. Ними стають банки, страхові компанії, автовиробники, рекламні фірми, виробники електроніки, видавничий бізнес, фарміндустрія.

В умовах конкуренції на ринку транспортних послуг у зв'язку з виникненням безлічі дрібних приватних компаній та активним освоєнням східного напряму перевезень іноземцями у поєднанні з жорсткою податковою політикою і подорожчанням ресурсів кур'єрські служби одними з перших відчули необхідність впровадження інформаційних технологій в управління виробничими процесами. Ефективна діяльність транспортних компаній вже неможлива без широкого використання інформаційних технологій і персональних комп'ютерів. З'являється інформаційна надмірність, що гальмує ділові процеси з-за необхідності переробляти величезну кількість непотрібних даних. І ця проблема не менш важлива, ніж проблема інформаційної недостатності. Необхідна оптимізація інформаційних потоків в транспортно-логістичних системах, і головним її завданням є якість і доступність необхідної інформації для фахівців,

зручність її подання і використання для вирішення різних виробничих завдань.

Обіг інформації дедалі істотніше впливає на ефективність управління підприємством, його фінансові успіхи. Більш того, все частіше інформацію називають "стратегічною сировиною". У розвинених країнах Заходу витрати на інформацію вже перевищують витрати на енергетику. І ці витрати при здоровому, правильному підході дають плоди. Насамперед, впровадження комп'ютерного обліку і обробки даних істотно підвищує продуктивність праці у сфері документообігу. Сучасні інформаційні технології, побудовані на основі використання концепцій інформаційних сховищ та інтелектуальної обробки даних, сьогодні можуть забезпечувати віддачу в 1000%.

Побудова єдиного інформаційного простору логістичного ланцюга при дозволяє забезпечити необхідну в сучасних умовах швидкість, повноту і точність отримання потрібних для надання транспортної послуги відомостей.

Метою даної роботи є створення кросплатформеної інформаційної системи кур’єрської служби.

Для виконання цієї мети в роботі поставлені і вирішені наступні завдання:

Здійснено дослідження предметної галузі

Здійснено огляд сучасних інформаційних систем для кур’єрських служб

Досліджено поняття кросплатформеної системи

Досліджено технології розробки кросплатформених систем

Описано етапи розробки та інструменти, які використовувались у процесі роботи

Виявлено вимоги до технічного та програмного забезпечення

Описана робота з програмою користувачем

Практична значимість дипломної роботи полягає в тому, що розроблена інформаційна система може використовуватися для обліку та

супроводу замовлень у процесі діяльності будь-якої організації, що працює в сфері кур’єрської доставки.

Кур'єрська доставка - динамічний вид бізнесу, в якому головними факторами успіху є оперативність передачі інформації та злагодженість дій всіх учасників логістичного ланцюжка, тому одним із дієвих механізмів підвищення ефективності бізнесу кур'єрських компаній на сьогоднішній день є інформаційні системи управління.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД І ПОСТАНОВКА ЗАДАЧІ

1.1. Огляд сучасних технологій виробництва меблі

Залежно від експлуатаційного призначення меблі можуть бути:

- побутової, тобто для використання в житлових приміщеннях;
- для громадського користування;
- для розміщення в адміністративно-громадських будівлях;
- для комплектації громадського та приватного транспорту.

Побутові меблі включає предмети, які використовуються на кухні, в житлових кімнатах, ванних, на терасах, в передпокоях, в заміських котеджах або на дачах і так далі. Це можуть бути столами, стільцями, диванами, кріслами, кріслами-ліжками, матрацами, тумбами, кухонними гарнітурами, диванами-ліжками та іншими виробами меблевого виробництва.

Відповідно до функціонального призначення вся існуюча меблі підрозділяється на ту, яка використовується для:

- сну;
- відпочинку (сидіння);
- зберігання речей і різних предметів побуту;
- прийому їжі;
- виконання будь-яких робіт.

Розглянемо функціональне призначення на прикладі столу, який в залежності від виконуваної завдання і застосовуваної технології виробництва меблів може бути кухонним, обіднім, письмовим, журнальним, робочим, купейним, креслярським, хірургічним, комп'ютерним і так далі.

Стіл буде відповідати певним конструктивним рішенням, габаритам і формі, виготовлятися з певного виду матеріалів - всі ці критерії залежать саме від функціонального призначення будь-якого предмета меблів.

Залежно від конструктивно-технологічних особливостей сучасні меблі буває:

- збірно-розбірна;
- корпусні;
- трансформована;
- нерозбірна;
- вбудована;
- навісна;
- секційна;
- плетені;
- гнута;
- гнуто-клеєна.

Технологія виробництва меблів впливає на її характер. Для її виготовлення можуть використовуватися різні матеріали.

Дерев'яні меблі може бути:

- гнуті;
- гнуто-клеєні;
- столярні;
- плетені;
- пресовані.

Меблі, виконана з полімерних матеріалів, може бути:

- литі;
- формовані;
- склеєні;
- пресовані.

Металеві меблі буває:

- литі;

- штамповані;

- зварені;

- оснащені металевими каркасами.

Подібна класифікація технології виробництва меблів залежить від вимог, які до неї висуваються. Вони бувають функціональними, конструктивними, техніко-економічними і естетичними.

Перша група вимог впливає на її функціональні характеристики, особливості проектування та виробництва, саме з їх допомогою забезпечується максимальний рівень комфорту споживачів, а також задоволення його естетичних уподобань і відповідність загальноприйнятим гігієнічним, фізіологічним і психологічним нормам і потребам.

Що стосується конструктивних вимог, то їх дотримання в процесі проектування і виробництва меблів необхідно для того, щоб готові предмети не тільки відповідали модним тенденціям, але і володіли стійкістю, міцністю в процесі експлуатації, надійністю, довговічністю і простотою при повсякденному використанні.

Техніко-економічні вимоги означають, що вся виготовляється меблі, незалежно від технології її виробництва, повинна відповідати існуючим вимогам технічних регламентів і державних стандартів, що відносяться до її виробництва, включаючи особливості виготовлення, норми витрачання матеріалів і уніфікацію вузлів і деталей готових виробів.

Сучасні виробники пропонують нашій увазі найширший асортимент моделей предметів меблів, які можуть мати різні конструкторські втілення, різна кількість елементів і деталей, виготовлятися з використанням найрізноманітніших матеріалів, а також технологій виробництва.

На рівень затребуваності того чи іншого виду меблів впливає, в першу чергу, її функціональне призначення, потім - зовнішній вигляд і рівень якості. Що ж стосується характеру виробництва, то можна говорити про індивідуальне, а також про серійне або масове підходах.

Технологія виробництва меблів при індивідуальному підході має на увазі, що предмети інтер'єру випускаються з обмеженням по кількості, відповідно до спеціального замовленням, повторний випуск виробів не передбачено. На підставі цього принципу працюють майстерні, спеціалізацією яких є виготовлення висококласної меблів та інших виробів за індивідуальним вимогам.

Серійна технологія виробництва меблів відрізняється виготовленням великих партій, їх повторним випуском відповідно до заздалегідь позначених виробничим планом. За таким принципом працює більшість сучасних підприємств, що займаються виготовленням предметів обстановки. Відповідно до кількості виробів в кожній серії можна говорити про дрібне, середньому і великому серійному виробництві меблів.

Що ж стосується масового характеру технології виробництва меблів, то мова йде про великі партії виробів, що не піддаються будь-яким конструктивним змінам протягом тривалого часу. Таких компаній спеціалізуються на масовому виготовленні незначного переліку товарів, що мають підвищений попит у споживачів.

На технологію виробництва меблів впливають конструкторські особливості всіх видів продукції, що випускається. Вона може бути різною в залежності від окремих стадій виробництва, але в той же час відповідної загальним принципам, які стосуються обробки деревини.

Першим етапом технології виробництва дерев'яних меблів є розкрій пиломатеріалів, рівень вологості яких не перевищує 2%, на чорнові заготовки. Наступний етап полягає в механічній обробці заготовок, трансформує їх у готові деталі необхідного розміру.

Аналогічна технологія виробництва застосовується для виготовлення меблів з пресованої деревини, гнутих, гнуто-клеєних і клеєних деревних матеріалів. Останній етап виробництва передбачає нанесення на оброблені заготовки лакофарбового покриття.

Корпусних меблями називають предмети, що володіють ящикової конструкцією, призначені для розміщення уздовж стін. Дана категорія представлена столами, стелажами, шафами, тумбами, стінками і іншими видами обстановки, в складі яких є окремі жорсткі частини.

При виробництві корпусних меблів повинні дотримуватися вимоги наступних державних стандартів:

ДСТУ ГОСТ 16371:2016 (ГОСТ 16371-2014, IDT) Меблі. Загальні технічні умови.

Залежно від довжини виробничого процесу, технологію виробництва корпусних меблів можна поділити на такі варіанти:

Повний технологічний процес має на увазі всі стадії виробництва, починаючи з виготовлення матеріалів для корпусу (це може бути ДСП, МДФ, меблевий щит) і закінчуючи складанням готових виробів. Такий варіант є оптимальним для масових і серійних виробництв, оскільки завдяки йому суттєво знижується собівартість матеріалів, проте для малого бізнесу він не підходить із-за високих витрат.

Середній цикл передбачає виробництво меблів з готових листів ДСП, ДВП, МДФ, тобто в даному випадку мова йде тільки про розкрої матеріалів і збірки готової продукції.

Короткий процес полягає у виробництві корпусних меблів на основі вже розкроєних на замовлення листів ДСП, ЛДСП, МДФ, тобто цей варіант характеризується виключно складанням готових меблів.

Технологія виробництва будь-якого виду корпусних меблів включає в себе п'ять основних етапів:

- Складання проекту готового виробу в різних площинах.
- Розкрій необхідних матеріалів для деталей майбутніх меблів.
- Висвердлювання гнізд для кріплень.
- Облицювання обрізних кромок (для цього використовується ламінована кромка, шпон, плівка ПВХ).

- Збірка готового виробу.

На детальність опису технології виробництва меблів впливає автоматизація виробництва і процентне співвідношення використання ручного та механізованого праці. Найпрогресивніше (і, отже, дороге), виробництво передбачає застосування автоматизованих верстатів з ЧПУ. У цьому випадку завданням оператора є тільки внесення розмірних даних в спеціалізовану комп'ютерну програму, конструювання бажаного вироби і запуск машини.

Технологія виробництва меблів в даному випадку виглядає наступним чином:

1. Після того як ескіз розроблений і затверджений замовником, за допомогою спеціальної програми, яка встановлюється на звичайний ноутбук, необхідно створити модель майбутнього виробу.

2. Закріпити на верстаті плиту матеріалу, з якого має бути виготовлений виріб; машина самостійно розпилює її на окремі деталі на підставі карт розкрою.

Виробництво меблів з ДВП передбачає на цьому закінчення підготовчих робіт і початок складання деталей. Якщо ж для виробництва меблів використовуються такі матеріали, як ДСП або ЛДСП, то розпиляні краю чорнових заготовок повинні бути в обов'язковому порядку механічно оброблені.

3. Елементи меблів з ДСП відправляються на крайколичкувальний верстат, на якому за допомогою клею і притискного преса зрізи плит облицьовують, використовуючи для цього ламіновану кромку, плівку ПВХ, меламін або інші кромочні матеріали.

4. Можливі два варіанти виконання отворів для кріплень, які залежать від комплектації верстата:

- напівавтоматичний, якщо мова йде про присадний верстаті;

- ручний, при якому отвору роблять перфораторами та електродрилі, користуючись кресленнями зі схемами присадки.

5. Після закінчення присадки отворів краю заготовок шліфуються (для згладжування, зняття кромки в висоту і довжину), а потім відправляються на складання.

6. Під час контрольного складання з використанням ручного інструменту, виявляються і усуваються недоліки і нестиковки. Потім готові вироби розбирають (якщо є така необхідність), упаковують і відправляють на склади готової продукції.

1.2. Огляд методів автоматизації меблевих підприємств

Специфіка меблевої галузі та історичні особливості її розвитку в Україні визначають специфічні умови і вимоги, які необхідно враховувати при випуску нових меблів. Більшість меблевих підприємств відносяться до малого бізнесу, для якого поділ процесу створення нових виробів на конструювання, технологічну підготовку виробництва, планово-економічні розрахунки та інше чітко не простежується. Виконання багатьох проектних, технологічних операцій поєднується і за часом, і за виконавцями. Це вимагає певної універсальності, поєднання в них елементів автоматизації виробництва [1].

На ринку програмних продуктів для автоматизації конструкторського і технологічного забезпечення виділився окремий напрямок, орієнтований на меблеві підприємства [2, 3].

У процесі комплексної автоматизації меблевого виробництва необхідно подолати кілька досить поширених помилок, а також вирішити ряд завдань, загальних для будь-якого підприємства. Тільки в цьому випадку можна розраховувати на успішне завершення проекту автоматизації випуску нових меблів.

Одна з помилок є наслідком уявної простоти виробництва меблевих виробів і полягає в непомірно завищених очікуваних показниках зростання ефективності виробництва. Коли ж це не спостерігається, то увага до автоматизації слабшає або доходить до необґрунтованих кадрових та

організаційних висновків. Це ще більш ускладнює ситуацію і в підсумку зводить до нуля всі зусилля, що робляться.

Керівництву меблевих підприємств слід чітко уявляти, що сьогодні і в осяжному майбутньому будь-яка автоматизована система являє собою лише інструмент в руках фахівців, за допомогою якого вони можуть (або не можуть) вирішити тільки одну задачу - підвищити продуктивність і ефективність своєї праці [1].

Підприємства часто очікують від автоматизації необґрунтованих, завищених результатів.

Як і будь-які автоматизовані системи, ЧПУ виробів меблів припускають ефективне вирішення низки і кадрових завдань. Кінцева ж мета автоматизації полягає в підвищенні ефективності виконання технологічних процесів і оптимізації інформаційних зв'язків між ними. Виконання кожної технологічної операції на будь-якому рівні передбачає отримання вхідної інформації, її обробку та передачу вихідної інформації для виконання наступних операцій. Подібна схема універсальна. Вона визначається самим фактом існування автоматизованих технологій.

Впровадження систем ЧПУ для виробництва меблів дозволяє підвищити продуктвиність і якість реалізації обробки і передачі інформації. Зростання ефективності роботи підприємства визначається його існуючою структурою. Технологічний перехід на новий рівень роботи, а саме це і передбачає впровадження систем автоматизації, неможливий без кардинальної реконструкції організаційної, технологічної структури підприємства. Це завдання вирішується шляхом проведення необхідних досліджень із залученням незалежних спеціалістів.

На підставі аналізу та узагальнення досвіду успішного впровадження комплексних систем автоматизації на меблевих підприємствах доцільно використовувати структуру інформаційних потоків виробництва, яка дозволяє отримати максимальну віддачу від автоматизації (рис. 1.1). Суцільними лініями на ній показані інформаційні потоки, документація. А

пунктирними - контролюючі інформаційні потоки, які хоча і не припускають формування окремих документів, але, тим не менш, відіграють суттєву роль [1].

Рис. 1.1 Структура виробничих потужностей для високотехнологічних інформаційних потоків автоматизованого управління виробництвом меблевого підприємства.

1.3. Висновки розділу

Значення меблевої промисловості в господарському комплексі України є великим.Особливо в нашому повсягденному житті широко використовується продукція цієї промисловості. Насамперед меблі забезпечують нам якнайкращий комфорт. Кожний тип меблів має свою функціональність. Ми не можемо уявити наше життя без офісних, журнальних столів чи столів

побутового призначення, стінок, тумбів, кухонних меблів, диванів, м"яких крісел, стільців, торгового обладнання чи інших функціональних меблів. Меблі забезпечують умови нашого життя, а також призначаються для дизайну житла, який з кожним роком покращується.

Жодна організація не може обійтися без меблевої продукції. Офіси потрібно устаткувати офісними столами, де можна зберігати документи, цінні папері, архіви, контракти, бланки та інше; шафами;стінками; офісними кріслами; меблями для комп"юторів і т. д. Приміщення, де проводиться торгівля не може обійтися без торгового обладнання, яке функціонує для якнайкращого оформлення вітрини для кращого продажу продукції[2;ст.4].

З давних давен, ще тоді, коли не існувало меблевої промисловості, люди намагалися оформити своє житло устаткуванням, яке б допомагало створити кращі умови життя. І з часом виникла меблева промисловість, яка щороку розвивається і до сьогоднішого дня.

Отож на сьогоднішній день кожен може придбати меблі на любий смак та за любою ціною.

Отже, можна зробити висновок, що меблева промисловість потрібна і вона стала невід"ємною частиною життя людини.І потрібно зазначити, що меблева промисловість розвивається , так як і повинна би розвиватися разом із зростанням людських потреб, щоб їх задовільнити. Але Україна не є досить розвиненою країною і тому нам потрібно більше вдосконалити виробництво меблів для досягнення вищого рівня розвитку країни та покращити умови життя.

РОЗДІЛ 2 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ КЛІЄНТ-СЕРВЕРНИХ СИСТЕМ

2.1. Огляд архітектури клієнт-серверних систем

Сучасні тенденції розвитку інформаційних систем у великій мірі зумовлені необхідністю підвищення швидкості доступу кінцевого користувача - фахівця до потрібної інформації. Сьогодні інформаційні системи у фінансових установах засновані на концепції відкритих систем невід'ємною складовою частиною яких є технологія клієнт/ сервер. Слід зазначити, що свого часу централізовану обробку даних змінила обробка даних в режимі розподілу часу на центральному комп'ютері. Згодом процедури обробки та доступу до даних почали розприділятися між робочими

станціями, що зв'язані з центральним комп'ютером. Тому початково поняття клієнт/сервер означало використання персональних комп'ютерів клієнтів, об'єднаних локальними чи глобальними мережами з центральним комп'ютером або сервером рис. 1.

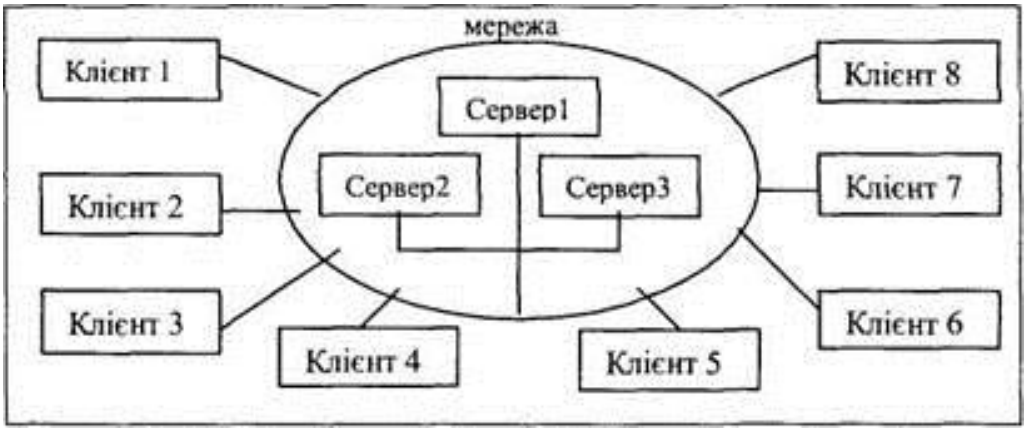


Рис. Модель технології клієнт/сервер.

Клієнт - це робоча станція користувача, що виконує функції взаємодії з користувачем, здійснює необхідні обчислення і забезпечує приєднання до віддалених ресурсів баз даних, засобів їх обробки та управління ними.

З технічної сторони в якості клієнта можуть бути використані звичайні персональні комп'ютери різних фірм, як наприклад IBM, HP, Compaq, Dell тощо, до яких не пред'являються специфічні вимоги.Основні функції, які мають забезпечити дані технічні засоби - це обслуговування введення-виведення інформації користувача, локальне збереження особистих даних, використання периферійного устаткування та управління взаємодією з віддаленими ресурсами серверів.

Програмне забезпечення клієнта може використовувати запуск програм на віддалених робочих місцях чи серверах. Це дозволяє не тільки зберігати фрагменти (модулі) прикладних програм на різних комп'ютерах для розподілу їх між користувачами, але й виконувати їх на різних комп'ютерах. Робоча станція клієнта може не мати жорсткого диска з якого забезпечується завантаження системних програм. В цьому випадку використовується віддалене завантаження операційної системи з віддаленого сервера в операційну пам'ять комп'ютера з використанням мережі передачі даних. Програмне забезпечення клієнта може надавати користувачу сервіси

прямого мережевного обміну з допомогою відомих протоколів: TCPЯP, IPX, NetBios, Ethernet, X.25, тощо.

Користувач має програмні засоби для розробки прикладних програм за технологією клієнт/сервер та засоби взаємодії з віддаленими базами даних. Сервери баз даних зберігають та оброблюють дані, приймаючи від користувача запити та повертаючи йому результат обробки.

Сервер - це одно або багатопроцесорний комп 'ютер з розділеними пам 'яттю, обробкою даних, засобами телекомунікації і засобами управління устаткуванням.

Сучасні мережі передачі даних включають велику кількість кінцевих користувачів, тобто клієнтів і велику кількість комп'ютерів- серверів, що забезпечують їх роботу. Сервери розрізняються за функціями обслуговування клієнтів: файл-сервери, сервери баз даних, комунікаційні сервери, обчислювальні сервери тощо.

В якості серверів використовуються більш потужні комп'ютери з розділеною пам'яттю та периферією. Системне програмне забезпечення як правило вибирається з багатозадачних операційних систем, таких як: UNIX, Windows-NT, Windows-2000, VMS, OS/2 тощо.

Сервер взаємодіє з клієнтом за допомогою механізму транзакцій.

Транзакція - це сеанс обміну даними між робочою станцією та сервером, при якому сервер відслідковує запити клієнта, ставить їх в чергу, виконує згідно з розписом, сповіщає клієнту про виконання та надає результат обробки.

Технологія клієнт/сервер забезпечує використання:

<=> корпоративного управління всіма ресурсами інформаційної системи, що доступні через мережі передачі даних;

^> розділення доступу до даних і програм між: робочими станціями і серверами, які з'єднані мережами передачі даних;

^> організації програмного забезпечення на основі концепції відкритих систем.

Таким чином за технологією клієнт/сервер користувач не керує ходом виконання поставленої проблеми, а система автоматично здійснює рішення поставленої задачі, оптимальним чином використовуючи ресурси технічних

засобів, баз даних та засобів телекомунікації. Це означає, що одна задача може вирішуватись багатьма робочими станціями та серверами, які розміщені в різних частинах міста, країни, світу, тобто є територіально розприділеними.

2.1.1. Архітектура клієнт-сервер

Детально про те, що таке технології та архітектура клієнт-сервер, і чому вони здатні допомогти при зростанні інформаційних систем на підприємстві, ми поговоримо нижче. Тут же ми коротко опишемо принципові архітектурні відмінності цієї системи від файл-серверної, яка на сьогоднішній день найбільш поширена в промисловості.

Передумови появи клієнт-серверної технології:

- число користувачів більше 10 - 15 осіб;
- висока вартість великих OEM та засобів розробки для них;
- вирішення завдань масштабу підприємства і управління підприємством в цілому;
- онлайн робота з віддаленими користувачами;
- критичне забезпечення цілісності інформації (банки тощо). В

основі клієнт-серверних технологій лежать дві ідеї:

- загальні для всіх користувачів дані на одному або декількох серверах;
- багато користувачів (клієнтів) на різних обчислювальних установках спільно (паралельно і одночасно) оброблюють загальні дані.

Тобто системи, засновані на цих технологіях, розподілені лише щодо користувачів, тому часто їх вважають видом багатокористувацьких систем.

Як і для файл-серверної архітектури, складовими компонентами клієнт-серверної архітектури є сервер, клієнтські місця і мережева інфраструктура. Однак, на відміну від попереднього випадку, сервер тут є вже не сервером файлів, а сервером баз даних або навіть сервером додатків. Таким чином, на сервер лягає не просто зберігання файлів, а підтримання бази даних в цілісному стані або, в разі сервера додатків, навіть виконання тієї чи іншої частини прикладної задачі. Природно, що вимоги до сервера при цьому можуть зростати в рази.

З іншого боку, те, що сервер володіє інформацією про характер збереженої бази даних, дозволяє набагато збільшити ефективність обробки.

Тому для багатьох завдань автоматизації навантаження на сервер за рахунок більш оптимального виконання операцій над даними в порівнянні з аналогічними файл-серверними додатками може навіть зменшитися.

Відповідно, спілкування між клієнтом і сервером відбувається не на рівні файлів, а на рівні обміну запитами. Клієнт передає серверу

високорівневі запити на отримання тієї чи іншої інформації або на її зміну, а сервер повертає клієнту результати виконання запитів. При цьому, на відміну від файл-серверної архітектури, доступ до даних не є прозорим для користувача програми. Тому, технологія розробки таких додатків принципово відмінна від локальних і файл-серверних систем.

На сучасному етапі мережеве забезпечення для архітектури клієнт- сервер аналогічно файл-серверному. Клієнт-серверні системи можуть будуватися з використанням тих же мережевих технологій і на тій же мережевій інфраструктурі. Більш того, як правило, на підприємстві мирно співіснують обидві ці архітектури. Це викликано двома основними причинами. По-перше, що на будь-якому підприємстві багато завдань, пов'язаних із зберіганням та обміном документами, які являють собою окремі файли, а для них архітектура файл-сервер оптимальна. По-друге, файл-серверні завдання в тому чи іншому обсязі майже завжди зберігаються в тому чи іншому обсязі та експлуатуються поряд зі створюваними клієнт-серверними додатками.

З точки зору мережевої взаємодії принципово новим у додатках клієнт-сервер є можливість переходу до використання глобальної мережі. Швидкість обміну в такій мережі може бути на порядок нижче, а відстані між робочими місцями можуть досягати кілометрів і навіть десятків кілометрів. І все це працює, оскільки в додатках клієнт-сервер обсяг переданої інформації може бути радикально скорочений за рахунок використання високорівневих запитів до даних. Але головною зміною, яка може бути здійснена промисловим підприємством при переході до клієнт-серверної архітектури - це якісний стрибок у масштабах завдань, що вирішуються при комп'ютеризації, в тому, наскільки комплексно автоматизується керування виробництвом, в рівні цілісності та достовірності даних, що зберігаються в інформаційній системі.

Слід зазначити, що поява клієнт-серверних додатків на підприємстві -

процес важкий і часто болючий. Причому, крім вирішення технічних та фінансових проблем, при цьому переході дуже важливо здійснити зміну самого рівня управління, що тягне за собою величезну організаційну роботу.

2.1.2.Моделі взаємодії клієнт-сервер

У своєму розвитку технології «клієнт-сервер» пройшли кілька етапів, тому є різні моделі технології. Їх реалізація заснована на поділі структури СКБД на три компоненти:

- введення і відображення даних (інтерфейс з користувачем);
- прикладний компонент (запити, події, правила, процедури та функції, які характерні для даної предметної області);
- функції керування ресурсами (файловою системою, базою даних тощо).

Тому, в будь-якому додатку виділяються наступні компоненти:

- компонент подання даних;
- прикладний компонент;
- компонент керування ресурсом.

Зв'язок між компонентами здійснюється за певними правилами, які називають «протокол взаємодії».

Існують різні класифікації, але однією з найпоширеніших є використання чотирьох моделей технології «Клієнт-сервер»:

1. Модель файлового серверу (File Server - FS) (рис. 2, а).
2. Модель віддаленого доступу до даних (Remote Data Access - RDA) (рис. 2, б).
3. Модель сервера БД (Data Base Server - DBS) (рис. 2, в).
4. Модель сервера додатків (Application Server - AS) (рис. 2, г).

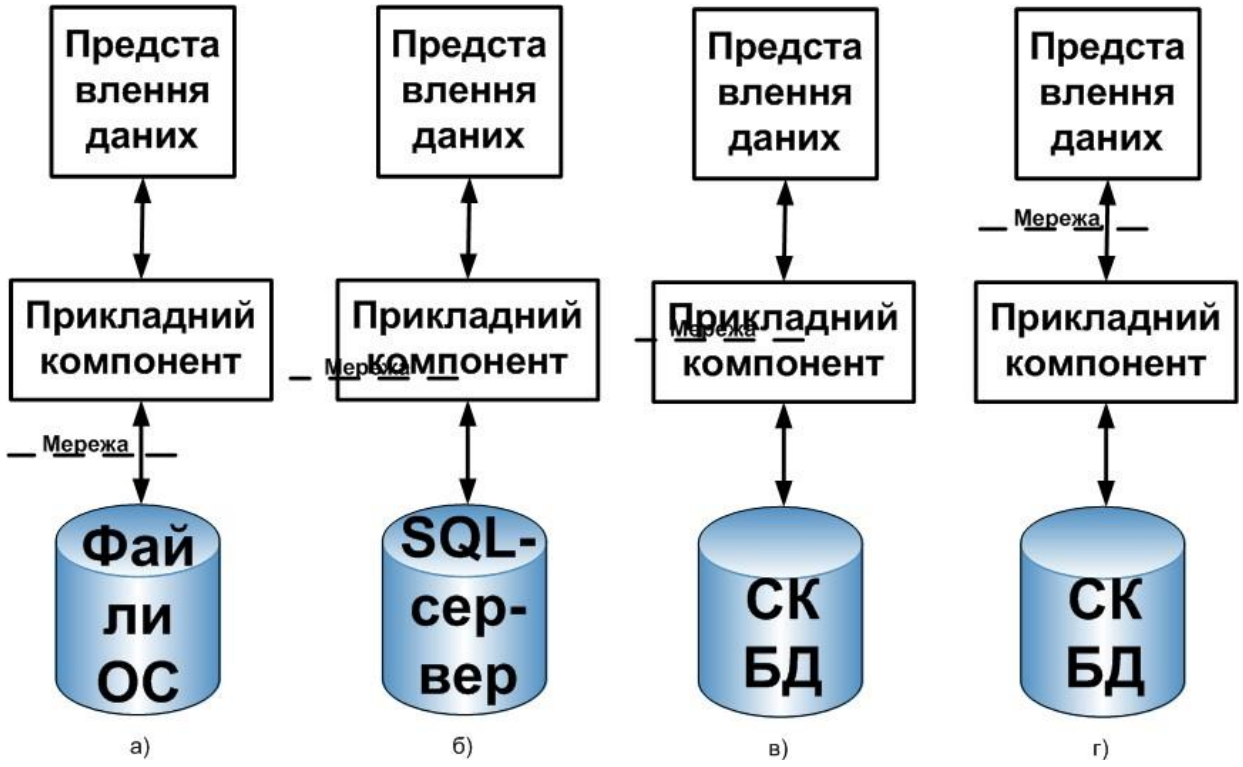


Рис. Моделі технології «Клієнт-сервер»: а) FS; б) RDA; в) DBS; г) AS

Історично першою з'явилася модель розподіленого представлення даних, яка реалізовувалася на універсальній ЕОМ з підключеними до неї неінтелектуальними терміналами. Управління даними та взаємодія з користувачем при цьому об'єднувалися в одній програмі, на термінал передавалася тільки «картинка», сформована на центральному комп'ютері.

Потім, з появою персональних комп'ютерів (ПК) і локальних мереж, були реалізовані моделі доступу до віддаленої Бази Даних. Деякий час базової для мереж ПК була архітектура файлового сервера. При цьому один з комп'ютерів є файловим сервером, на клієнтах виконуються додатки, у яких сполучені компонент подання й прикладний компонент (СКБД і прикладна Програма). Протокол обміну при цьому представляє набір низькорівневих викликів операцій файлової системи. Така архітектура, реалізована, як правило, за допомогою персональних СКБД, має очевидні недоліки - високий мережевий трафік і відсутність уніфікованого доступу до ресурсів.

З появою перших спеціалізованих серверів баз даних з'явилася можливість іншої реалізації моделі доступу до віддаленої Бази Даних. У цьому

випадку ядро СКБД функціонує на сервері, протокол обміну забезпечується за допомогою мови SQL. Такий підхід у порівнянні з файловим сервером веде до зменшення завантаження мережі й уніфікації інтерфейсу «клієнт-сервер». Однак, мережевий трафік залишається досить високим, крім того, як і раніше неможливо задовільнити адміністрування додатків, оскільки в одній програмі сполучаються різні функції.

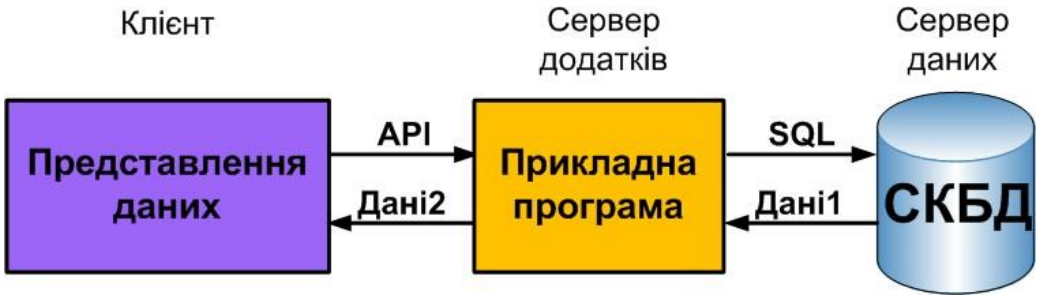
Пізніше була розроблена концепція активного сервера, який використовував механізм збережених процедур. Це дозволило частину прикладного компонента перенести на сервер (модель розподіленого додатку). Процедури зберігаються в словнику бази даних, розділяються між декількома клієнтами й виконуються на тому ж комп'ютері, що і SQL-сервер. Переваги такого підходу: можливо централізоване адміністрування прикладних функцій, значно знижується мережевий трафік (оскільки передаються не SQL-запити, а виклики збережених процедур). Недолік - обмеженість засобів розробки збережених процедур у порівнянні з мовами загального призначення (C і Pascal).

На практиці зараз звичайно використовуються змішані підходи:

- найпростіші прикладні функції виконуються збереженими процедурами на сервері;
- більш складні реалізуються на клієнті безпосередньо в прикладній програмі.

Зараз ряд постачальників комерційних СКБД оголосило про плани реалізації механізмів виконання збережених процедур з використанням мови Java. Це відповідає концепції «тонкого клієнта», функцією якого залишається тільки відображення даних (модель віддаленого подання даних).

Останнім часом також спостерігається тенденція до все більшого використання моделі розподіленого додатка. Характерною рисою таких додатків є логічний поділ додатка на дві та більше частини, кожна з яких може



виконуватися на окремому комп'ютері. Виділені частини додатка взаємодіють одна з одною, обмінюючись повідомленнями в заздалегідь погодженому форматі. У цьому випадку дволанкова архітектура клієнт- сервер стає триланкової (рис. 3), а в деяких випадках, вона може включати і більше ланок.

Рис. Триланкова архітектура

2.2. Методи розробки клієнт-серверних систем
2.2.1 Метод файлового серверу

В задачах обробки інформації, заснованих на системах баз даних, існують два варіанти розташування даних: локальний і віддалений. У першому випадку говорять про доступ до локальних даних, у другому - про доступ до віддалених даних. Локальні дані, як правило, розташовуються на жорсткому диску комп'ютера, на якому працює користувач, і знаходяться в монопольному керуванні цього користувача. Користувач при цьому працює автономно, не залежачи від інших користувачів та жодним чином не впливаючи на їх роботу. Дистанційні дані розташовуються поза комп'ютера користувача (користувачів) - на файловому сервері мережі або на спеціально виділеному для цих цілей комп'ютері.

Модель файлового серверу (File Server - FS) – це природне розширення персональних СКБД для підтримки багатокористувацького режиму і в цьому плані ще довго буде зберігати своє значення (рис. 4).

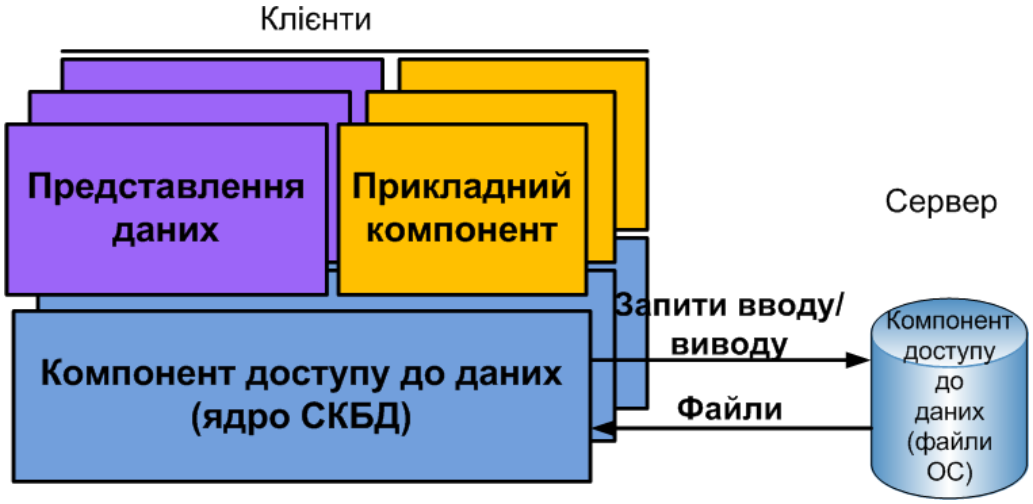


Рис. 4. Схема взаємодії FS-моделі

Особливості FS-моделі:

- всі основні компоненти розміщуються на клієнтському комп'ютері;
- модель характеризує не стільки спосіб створення ІС, скільки загальний спосіб взаємодії комп'ютерів в локальній мережі;
- один з комп'ютерів виділяється і визначається файловим сервером, тобто загальним сховищем будь-яких даних;
- сервер виконує чисто пасивну функцію;
- дуже проста та зрозуміла модель.

У стандартній файл-серверній архітектурі дані, розташовуючись на файл-сервері, є, по суті, пасивним джерелом. Вся відповідальність за їх отримання, обробку, а також за підтримку цілісності бази даних лежить на додатку, запущеному з робочої станції. При цьому, оскільки обробка даних здійснюється на робочій станції, по мережі переганяється вся необхідна для цієї обробки інформація, хоча обсяг даних, які цікавлять користувача, може бути менше в десятки разів. Наприклад, якщо користувача цікавлять всі працівники заданого підприємства, які беруть

участь у конкретному проекті, його додаток «отримає» спочатку всіх працівників і всі проекти з бази даних, і тільки після цього виконає необхідну вибірку.

Історично на персональних комп'ютерах використовувався саме цей підхід як більш простий в освоєнні. Однак великий обсяг даних, які переганяються по мережі, швидко «забиває» мережу вже при невеликому числі користувачів, істотно обмежуючи можливості зростання (масштабованості). Цей основний і самий істотний недолік змусив шукати способи зменшення навантаження на мережу.

Тут додатки виконуються на робочих станціях. Додаток включає модулі для організації діалогу з користувачем, бізнес-правила (транзакції), що забезпечують необхідну логіку обчислень, і ядро СКБД. Часто ядро СКБД в моделі файлового сервера не є вираженим і являє собою набір функцій, пов'язаних з іншими компонентами додатка. Додаток, включаючи і ядро СКБД, дублюється на різних робочих станціях. На файловому сервері зберігаються тільки файли бази даних (індекси, файли даних тощо) і деякі технологічні

файли (оверлейні файли, файли сортування тощо). Оператори SQL-звернення до СКБД, закодовані в прикладній програмі, обробляються ядром СКБД на робочій станції. СКБД організовує доступ до файлів бази даних для виконання оператора. По мережі передаються запити на читання/запис даних, індекси, проміжні та результуючі дані, блоки технологічних файлів.

На основі моделі файлового сервера функціонують такі популярні СКБД як FoxPro (Microsoft), dBase (Borland), CF-Clipper (Computer Associates International), Paradox (Borland) тощо. СКБД розглянутого класу коштують недорого, прості в установці та освоєнні. Також відсутні високі вимоги до продуктивності сервера та програмні компоненти СКБД не розподілені.

Але вони мають ряд істотних недоліків:

- системи, розроблені на базі цих СКБД, мають низьку продуктивність, оскільки всі проміжні дані передаються, як правило, по низькошвидкісний шині мережі, а прикладні програми і ядро СКБД виконуються на малопотужних робочих станціях;
- високий мережевий трафік;
- низька масштабованість;
- відсутність механізмів безпеки БД;
- ці СКБД не підтримують розподілену обробку.

В силу перерахованих недоліків модель файлового сервера практично не використовується в розподілених інформаційних системах.

2.2.2. Метод віддаленого доступу до даних

Модель віддаленого доступу до даних (Remote Data Access – RDA) – архітектура, яка заснована на обліку специфіки розміщення і фізичного маніпулювання даними у зовнішній пам'яті для реляційних СКБД (рис. 5).

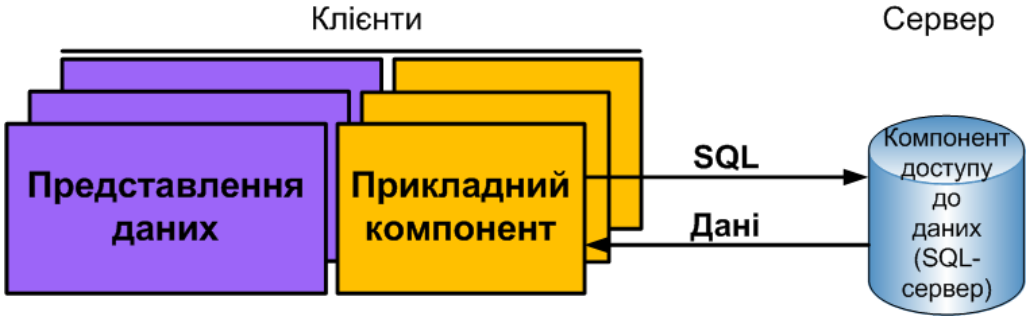
У RDA-моделі для обробки даних виділяється спеціальне ядро - так званий *SQL-сервер*, який приймає на себе функції обробки запитів користувачів, іменованих тепер клієнтами. Сервер баз даних являє собою програму, яка виконується, як правило, на потужному комп'ютері. Додатки-клієнти посилають з робочих станцій запити на вибірку (вставку, оновлення, видалення) даних. При цьому сервер виконує всю

«брудну» роботу з відбору даних, відправляючи клієнтові тільки необхідну «вичавлювання». Якщо наведений вище приклад перебудувати з урахуванням клієнт-серверної архітектури, то додаток- клієнт «отримає» від сервера в якості результату список тільки тих працівників, які беруть участь у заданому проекті, і не більше того! Такий підхід забезпечує вирішення трьох важливих завдань:

- зменшення навантаження на мережу;
- зменшення вимог до комп'ютерів-клієнтам;
- підвищення надійності та збереження логічної цілісності бази даних.

Тут додатки також виконуються, в основному, на робочих станціях. Додаток включає модулі для організації діалогу з користувачем і бізнес-правила (транзакції). Ядро СКБД є загальним для всіх робочих станцій і функціонує на сервері. Оператори звернення до СКБД (SQL-оператори), закодовані в транзакції, не виконуються на робочій станції, а пересилаються для обробки на сервер. Ядро СКБД транслює запит і виконує його, звертаючись для цього до індексів та інших проміжних даних. Назад на робочу станцію передаються тільки результати обробки оператора.

Рис. 5. Схема взаємодії RDA-моделі



У файлах БД на сервері знаходиться також і системний каталог БД. У числі іншого, в каталог БД поміщаються:

- відомості про зареєстрованих користувачів;
- привілеї користувачів;

На клієнтських комп’ютерах встановлюються частини СКБД, які реалізують:

- інтерфейсні функції;
- прикладні функції. Прикладний

компонент включає:

- бібліотеки запитів;
- процедури обробки даних.

Прикладний компонент повністю розміщується і виконується на клієнтській частині - формує SQL-інструкції, що направляються SQL- серверу.

SQL-сервер - спеціальний програмний компонент, який:

- орієнтований на інтерпретацію SQL-інструкцій;
- високошвидкісне виконання низькорівневих операцій з даними;
- приймає і координує SQL-інструкції від різних клієнтів;
- виконує SQL-запити;
- перевіряє та забезпечує виконання обмежень цілісності даних;
- направляє клієнтам результати обробки SQL-інструкцій - набори даних.

До переваг RDA-моделі слід віднести:

- зменшення завантаження мережі;
- SQL-сервер забезпечує виконання обмежень цілісності та безпеки даних;
- уніфікований інтерфейс взаємодії прикладної частини ІС із загальними даними;
- в рамках SQL така взаємодія реалізована через ODBC-протокол, та називається *інтероперабельністю*.

Недоліки RDA-моделі:

- високі вимоги до клієнтських комп'ютерів;
- великий обсяг технологічних змін;
- зміна структури управління та бізнес-процесів;
- значний, хоча і менший, ніж у моделі FS, мережевий трафік.

Наступним великим недоліком є великий обсяг власне технологічних змін, що виникають при спробі впровадження архітектури клієнт-сервер. Клієнт-серверна система вимагає іншого рівня технічної грамотності з боку, як співробітників інформаційних служб, так і кінцевих користувачів. Витрати на перепідготовку користувачів та експлуатаційного персоналу, перебудова структури

автоматизації підприємства становлять більшу частину айсберга, ніж ясно видимі прямі витрати на модернізацію апаратури, закупівлю та/або розробку

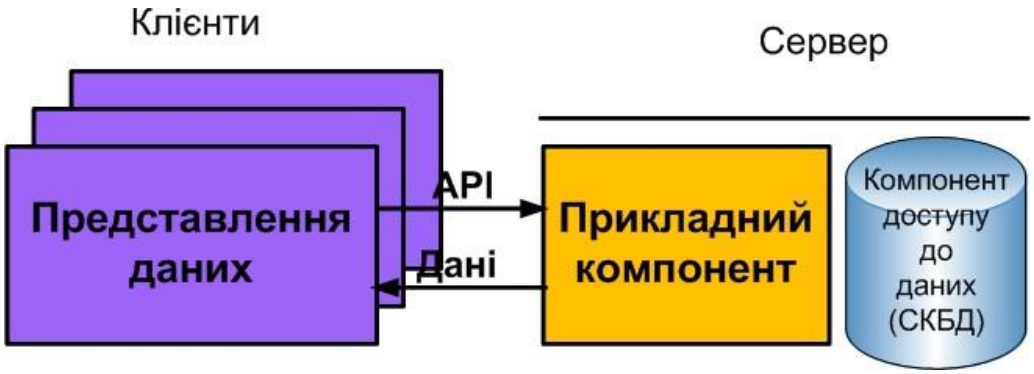
необхідного забезпечення.

І, нарешті, самим великим підводним каменем на шляху створення АС на підприємстві є необхідність міняти структуру управління і пов'язані з цим організаційні витрати. Спроба впровадити нові технологічні рішення нічого не змінюючи в суті автоматизованих бізнес- процесів може закінчитися безрезультатно. При цьому підприємство понесе прямі матеріальні збитки через великий обсяг дорогого апаратного та програмного забезпечення, лежачого мертвим вантажем, а також через відволікання співробітників від виконання основних службових обов'язків. У кращому випадку будуть впроваджені окремі ділянки клієнт-серверної системи, при цьому фактично нове програмне забезпечення буде використовуватися на старому ідейному рівні.

2.2.3.Модель сервера бази даних

Для поліпшення попередньої RDA-моделі в сучасних СКБД використовується *модель сервера баз даних (DataBase Server - DBS)*, в якій на сервері можуть запускатися так звані збережені процедури і тригери, які разом з ядром СКБД утворюють сервер бази даних (рис. 19.5.).

Рис. 6. Схема взаємодії DBS-моделі



До збережених процедур можна звертатися з додатків на робочих станціях. Це дозволяє скоротити розмір коду прикладної програми і зменшити потік SQL-операторів з робочої станції, так як групу необхідних SQL-запитів можна закодувати в збереженій процедурі. Тригери - це програми, які виконуються ядром СКБД перед або після оновлення (UPDATE, INSERT, DELETE) таблиці бази даних. Вони дозволяють автоматично підтримувати цілісність бази даних.

Модель сервера бази даних (БД) підтримують наступні СКБД: Oracle, DB2 (IBM), MS SQL Server (Microsoft), MySQL (Oracle), FireBird,

PostgreSQL, Sybase (SAP), тощо. Причому на перші чотири СКБД припадають понад 85% ринку. СКБД розглянутого класу мають наступні переваги:

- системи, мають високу продуктивність, тому що запити виконуються на високошвидкісних серверах;
- зниження мережевого трафіку, тому що по шині передаються тільки SQL-запити і результати з виконання;
- СКБД підтримують розподілену обробку;
- більш гнучка «настройка» на предметну область;
- забезпечення узгодженого стану даних;
- надійність зберігання і обробки даних;
- ефективна координація колективної роботи користувачів із загальними даними;
- в рамках цих СКБД пропонується велика кількість сервісних продуктів, що полегшують розробку додатків і створення розподіленої системи.

Дійсно, тепер сервер БД (які інколи мають назву SQL-сервер) повертає клієнтському додатку тільки «вичавлювання» того, що він переглянув в базі, а воно, в загальному випадку, дійсно становить малу частину від загального обсягу. Тому в мережі не спостерігається різкого збільшення навантаження при збільшенні кількості клієнтів. Клієнтські

ж додатки можуть виконуватися на менш потужних (в порівнянні з сервером) комп'ютерах завдяки тому, що їм практично не потрібно виконувати ніякої додаткової обробки отриманих від сервера результатів запиту (хоча, звичайно ж, це не заборонено). Побічним ефектом зменшення навантаження на мережу є підвищення швидкості виконання додатків клієнтів. Крім того, система легше масштабується - легше і дешевше замінити один сервер на більш потужний, ніж десятки робочих станцій.

Ці СКБД мають і недоліки:

- вони набагато дорожче СКБД попереднього класу, складні в освоєнні;
- для ефективної роботи цих СКБД потрібні високошвидкісні (а тому й дорогі) сервери та мережі.

Найбільш важливим результатом переходу в архітектуру клієнт- сервер є

гарантоване збереження логічної цілісності бази даних, тобто система стає більш стійкою і більш захищеною. Досягається це завдяки можливості перекласти турботу про збереження цілісності бази на сервер. Для цього «хороші» сервери мають великий набір вбудованих механізмів, що захищають систему від невірних дій клієнтів. Серед цих механізмів можна назвати такі як обмеження цілісності, декларативна цілісність посилань, тригери, віртуальні таблиці (подання), авторизація користувачів тощо.

2.2.4. Модель сервера додатків

Щоб рознести вимоги до обчислювальних ресурсів сервера у відношенні швидкодії і пам'яті за різними машинам, використовується *модель сервера додатків (Application Server AS)* (рис.7). AS-модель зберігає сильні сторони DBS-моделі.

Особливості AS-моделі:

- перенесення прикладного компонента AIC на спеціалізований сервер;
- на клієнтських машинах - тільки інтерфейсна частина системи;
- виклики функцій обробки даних спрямовуються на сервер додатків;
- низькорівневі операції з даними виконує SQL-сервер.

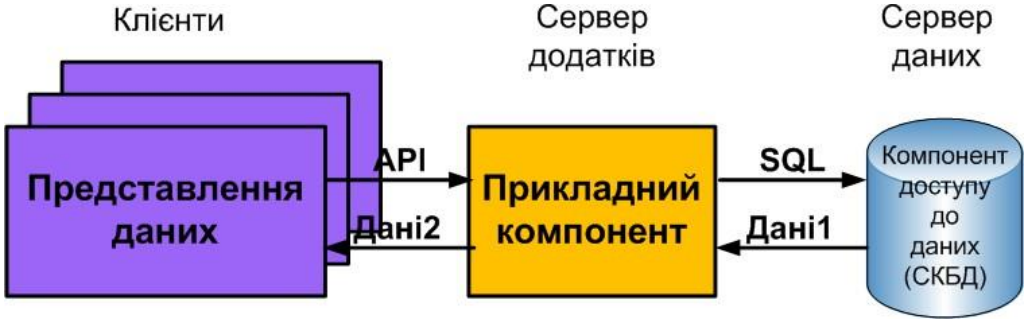


Рис. 7. Схема взаємодії AS-моделі

Використання цієї моделі дозволяє розвантажити робочі станції, тобто перейти до «тонких» клієнтів. Звичайно, сервер додатків можна організувати і за допомогою збережених процедур. Але для реалізації збережених процедур використовують мови високого рівня (наприклад, в Oracle - мову PL/SQL), тому програми виходять ресурсоємними. Причому можливості цих мов дуже широкі: від циклів до обробки даних

на рівні бітів. Збережені процедури також не підтримують розподілені додатки, тобто вони не забезпечують автоматичний запуск необхідної програми на іншому сервері.

Сучасні СКБД використовують наступні процедурні мови:

- Transact-SQL (MS SQL Server, Sybase);
- PL/SQL (Oracle);
- PSQL (FireBird);
- SQL PL (IBM DB2);
- JetSQL (MS Access);
- PL/pgSQL (PostgreSQL).

У тому випадку, коли інформаційна система об'єднує досить велику кількість різних інформаційних ресурсів і серверів додатків, постає питання про оптимальне управління всіма її компонентами. Для цього використовують програмні засоби, які часто називають*менеджерами транзакцій, моніторами транзакцій (Transaction Processing Monitor - TPM)*, і які розміщуються на сервері додатків. Також менеджер транзакцій і додатки можуть запускаютися на одному комп'ютері (хоча це не є обов'язковим), щоб зменшити потік SQL-запитів по мережі.

Транзакція - це послідовна сукупність операцій над даними (SQL-інструкцій), що має окреме смислове значення. Але часто в моніторах транзакцій поняття транзакції розширюється - в даному випадку це не атомарна дія над базою даних, а будь-яка дія в системі – видача повідомлення, запис в індексний файл, друк звіту тощо.

Для спілкування прикладної програми з монітором транзакцій використовується спеціалізований API (Application Program Interface - інтерфейс прикладного програмування), який реалізується у вигляді бібліотеки, що містить виклики основних функцій (встановити з'єднання, викликати певний сервіс тощо). Сервери додатків (сервіси) також створюються за допомогою цього API, кожному сервісу присвоюється унікальна назва. Монітор транзакцій, отримавши запит від прикладної програми, передає її виклик відповідному сервісу (якщо той не запущений, породжується необхідний процес), після обробки

запиту сервером додатків повертає результати клієнту. Для взаємодії моніторів транзакцій з серверами баз даних розроблений протокол XA. Наявність такого уніфікованого інтерфейсу дозволяє використовувати в рамках однієї програми кілька різних СКБД.

Монітор транзакцій усуває такі витрати спільної обробки:

- *втрачені зміни* - дві транзакції одночасно змінюють один об'єкт БД;
- *брудні дані* - одна транзакція змінює об'єкт, а друга читає дані з нього;
- *неповторювані читання* - одна читає об'єкт, а друга транзакція змінює його.

Для ізоляції транзакцій і подолання ситуацій неузгодженої обробки даних використовують *серіалізацію транзакцій* - виконання транзакцій таким чином, щоб результат їхньої спільного виконання був еквівалентний результату їх послідовного виконання.

Використання моніторів транзакцій у великих системах дає наступні переваги:

- концентрація всіх прикладних функцій на сервері додатків забезпечує значну незалежність як від реалізації інтерфейсу з користувачем, так і від конкретного способу керування ресурсами. При цьому також забезпечується централізоване адміністрування додатків, оскільки всі додатки знаходяться в одному місці, а не «розмазані» по мережі на клієнтських робочих місцях;
- монітор транзакцій в змозі сам запускати і зупиняти сервери додатків. В залежності від завантаження мережі та обчислювальних ресурсів він може перенести або скопіювати частину серверних процесів на інші вузли. Це забезпечує балансування навантаженням серверів;
- забезпечується динамічна конфігурація системи, тобто без її зупинки може бути доданий новий сервер ресурсів або сервер додатків;
- підвищується надійність системи, тому в разі збоїв сервер додатків може бути переміщений на резервний комп'ютер;
- з'являється можливість керування розподіленими базами даних. До недоліків AS-моделі слід віднести:
- необхідні додаткові потужності (може навіть окремий додатковий

сервер);

- підвищує трафік в мережі.

2.2.5.Відкриті системи

Перші інформаційні системи, з'явилися в середині XX-го століття. Вони ґрунтувалися на мейнфреймах компанії IBM, файлової системі OC/360, а згодом на ранніх СКБД типу IMS і IDMS (в СРСР на клонах відповідно ЄС ЕОМ, ОКА і КАМА). Ці системи прожили довге і корисне життя, багато хто з них до цих пір експлуатуються. Але з іншого боку, повна орієнтація на апаратні засоби і програмне забезпечення IBM породила серйозну проблему «успадкованих систем» (legacy systems). Проблема полягає в тому, що виробничий процес не дозволяє припинити або навіть призупинити використання морально застарілих систем, щоб перевести їх на нову технологію. Багато серйозних дослідників сьогодні зайняті намаганнями вирішити цю проблему. Серйозність проблеми успадкованих систем з очевидністю підтверджує, що інформаційні системи (ІС) і БД, які лежать в їх основі, є занадто відповідальними і дорогими продуктами, щоб можна було дозволити собі їх переробку при зміні апаратної платформи або навіть системного програмного забезпечення (головним чином ОС і СКБД). Для цього програмний продукт повинен мати властивості легкої переносимості з одної апаратно-програмної платформи на іншу. Це, однак, не означає, що при переносі не можуть знадобитися якісь зміни у вихідних текстах; головне, щоб такі зміни не означали корінної переробки системи.

Іншою природною вимогою до інформаційних систем є можливість їх розвитку за рахунок включення додаткових програмних та інформаційних компонентів.

Яким же чином можна одночасно задовольнити обидві ці вимоги вже на стадії проектування та розробки інформаційної системи? Відповіддю, який мені здається правильною, є наступна: необхідно слідувати принципам «Відкритих Систем» і відповідних міжнародних або фактичних загальновизнаних стандартів у всіх необхідних видах забезпечення.

Основною ідеєю підходу відкритих систем є спрощення комплексування інформаційно-обчислювальних систем за рахунок міжнародної та національної

стандартизації апаратних і програмних інтерфейсів. Головною спонукальною причиною розвитку концепції відкритих систем з'явився повсюдний перехід до використання комп'ютерних мереж і ті проблеми комплексування апаратно-програмних засобів, які викликав цей перехід. У зв'язку з бурхливим розвитком технологій глобальних комунікацій відкриті системи набувають ще більшого значення і масштабності. Прихильність концепції відкритих систем повинна забезпечити власникам ІС незалежність від конкретного постачальника. Орієнтуючись на продукцію компаній, які дотримуються стандартів відкритих систем, споживач, який набуває будь-який продукт такої компанії, не потрапляє до неї в рабство. Він може продовжити нарощування потужності своєї системи шляхом придбання продуктів будь-якої іншої компанії, що дотримується цих стандартів. Причому це стосується як апаратних, так і програмних засобів.

Практичною опорою системних і прикладних програмних засобів відкритих систем є стандартизована операційна система. В даний час такою системою є UNIX. Фірмам-постачальникам різних варіантів ОС UNIX в результаті тривалої роботи вдалося прийти до угоди про основні стандарти цієї операційної системи. Зараз всі поширені версії UNIX в основному сумісні по частині інтерфейсів, що надаються прикладним (а в більшості випадків і системним) програмістам. Незважаючи на появу нового претендента на стандарт системи Windows, саме UNIX залишиться основою відкритих систем у найближчі роки.

Технології та стандарти відкритих систем забезпечують реальну і перевірену практикою (в тому числі і в СНД) можливість виробництва системних і прикладних програмних засобів з властивостями мобільності (portability) та інтероперабельності (interoperability).

Властивість мобільності означає порівняльну простоту перенесення програмної системи в широкому спектрі апаратно-програмних засобів, відповідних стандартам. Інтероперабельність означає спрощення комплексування нових програмних систем на основі використання готових компонентів зі стандартними інтерфейсами. Під цим розуміється дотримання певних правил або залучення додаткових програмних засобів, що забезпечують можливість взаємодії незалежно розроблених програмних

модулів, підсистем або навіть функціонально завершених програмних систем.

Використання підходу відкритих систем вигідно і виробникам, і користувачам. Перш за все, відкриті системи забезпечують природне рішення проблеми поколінь апаратних і програмних засобів. Виробники таких засобів не змушують вирішувати всі проблеми заново; вони можуть, принаймні, тимчасово продовжувати кооперування, використовуючи існуючі компоненти.

Зауважимо, що при цьому виникає новий рівень конкуренції. Всі виробники зобов'язані забезпечити деяке стандартне середовище, але змушені добиватися його якомога кращої реалізації. Звичайно, стандарти не панацея: через якийсь час вони починають грати роль стримування прогресу, і тоді їх необхідно переглядати.

Перевагою для користувачів є те, що вони можуть поступово замінювати компоненти системи на більш досконалі, не втрачаючи працездатності системи. Зокрема, в цьому криється рішення проблеми поступового нарощування обчислювальних, інформаційних та інших потужностей комп'ютерної системи.

2.2.6. Міжнародні стандарти мов

Розглянемо трохи докладніше, які стандарти можна використовувати при розробці інформаційної системи в даний час. Сьогодні програмне забезпечення ІС пишеться на деякій мові

програмування, в неї вбудовуються оператори мови SQL і виклики бібліотечних функцій операційної системи.

Відповідно, перш за все, слід звертати увагу на ступінь стандартизованості використовуваної мови програмування. На сьогодняшній день прийняті міжнародні стандарти мов Fortran, Pascal, Ada, C, C++. Fortran, навіть у своєму найбільш розвиненому вигляді стандарту Fortran-95, не є мовою, відповідним для програмування інформаційних систем. У стандарт мови Pascal не включені засоби роздільної компіляції програм. Звичайно, в принципі можна оформити повний вихідний текст ІС у вигляді одного файлу, але навряд чи це розумно і практично. Мова Ada, взагалі кажучи, придатна для будь-яких цілей. На неї можна писати і інформаційні системи (що, до речі і роблять американські і деякі вітчизняні військові). Але аж надто вона громіздка.

На думку багатьох програмістів, найбільш зручний стандарт на сьогоднішній день існує для мови С. Досвід показує, що при грамотному використанні стандарту

С ANSI/ISO проблеми переносу програм, пов'язані з особливостями апаратури або компіляторів, практично не виникають (якщо враховувати наявні в самому стандарті рекомендації по створенню переносних програм). Декількох років виявилося достатньо, щоб забезпечити повну відповідність стандарту практично всіх індустріальних компіляторів мови С.

Дуже важливо, що в стандарті ANSI С специфіковані не тільки мова, але й базові бібліотеки стандартної системи програмування (зокрема, основні компоненти бібліотеки вводу/виводу). Наявність стандарту на бібліотечні функції та його суворе дотримання в реалізаціях в ряді випадків дозволяє створювати не дуже складні додатки, що переносяться не тільки між різними апаратними платформами, але і між різними операційними середовищами.

Був, нарешті, прийнятий стандарт С++. Мабуть, це означає (принаймні, на це можна сподіватися), що через кілька років можна буде говорити про мобільне програмування на С++ в тому ж сенсі, в якому можна говорити про це сьогодні по відношенню до С. Маючи на увазі взаємодії з базами даних, ми говоримо майже виключно про мову SQL. SQL з самого свого зародження був складною мовою зі змішаною декларативно-процедурною семантикою і не ідеальним синтаксисом. Тим не менш, саме SQL став єдиною практично використовуваною мовою реляційних баз даних, хоча були й інші кандидати, зокрема, мова системи Ingres Quel (QBE).

2.3. Технології розробки САПР меблів

2.3.1 Проектування та виготовлення виробів засобами програми САПР NX

В теперішній час багато підприємств в машинобудівній галузі зустрічаються із проблемами при пошуку нових ринків збуту продукції. Конкуренція приводить до підвищених вимог з боку замовників, як по строках виготовлення, так і по якості. Задача вирішується за рахунок впровадження на підприємстві систем автоматичного проектування (САПР).

Така ситуація вимагає відповідної підготовки випускників ВНЗ зі спеціальностей, пов'язаних з машинобудуванням. Ця тенденція враховується провідними вітчизняними й закордонними вищими навчальними закладами.

При впровадженні систем автоматизованого проектування в навчальний процес виникає проблема вибору програмного продукту, тому що сучасний ринок САПР пропонує широкий спектр програмних продуктів, орієнтованих на розв'язок глобальних або локальних завдань і на різні фінансові можливості покупця. Ці

продукти досить умовно можна класифікувати по рівнях:

— верхній рівень — багатофункціональні інтегровані системи з єдиною структурою даних і набором проблемно- орієнтованих додатків.

— середній рівень — група функціонально незалежних продуктів, що працюють із єдиною структурою даних.

— нижній рівень — сукупність програм, орієнтованих на формування конструкторської й технологічної документації. Ці програми, як правило, не зв'язані єдиною структурою даних.

Із САПР верхнього рівня зараз велике поширення, як у промисловості, так і у ВНЗ поширена програма NX (фірма Siemens), тому що програма використовує графічне ядро Parasolid (власна розробка), яке є стандартом для багатьох САПР різного рівня. Також Siemens надає виробництвам технічну підтримку, а навчальним закладам безкоштовні університетські ліцензії.

У багатьох ВНЗ ведеться навчання студентів програмі САПР NX. Зокрема, програма застосовується для розробки пристосувань, застосовуваних при виконанні технологічних операцій (обробка заготовок, складання виробів, контроль).

Традиційно виділяють два методи роботи зі збірками: «знизу-нагору»; «зверху-униз». При використанні концепції побудови складання «знизу-нагору» деталі й підбірки створюються як незалежні компоненти, і позиціонуються залежно від положення раніше доданих компонентів, або щодо обраної системи координат. Концепція роботи «зверху-униз» має на увазі створення та складання верхнього рівня послуг, що рухається униз по ієрархії, додаючи нові компоненти й підбірки.

При розробці пристосувань застосовувалася концепція «знизу- нагору» з використанням різних поєднань. У цьому випадку додавання компонентів відбувається незалежно один від одного. Метод роботи з використанням сполучень є найпоширенішим і часто найбільш ефективним при розробці обладнань і агрегатів. Особливо цей метод актуальний у випадках, коли необхідно проводити кінематичний аналіз конструкції, що створюється розрахунок розмірів, ланок, вузлів й у випадках, коли використовується безліч стандартних і запозичених компонентів.

Розробка пристосувань є трудомістким процесом. Однак сучасні системи автоматизованого проектування дозволяють знизити трудоємність основного процесу проектування за рахунок застосування такої опції як «Сімейство деталей». Дана опція дозволяє на базі однієї модифікації типової деталі пристосування створити, використовуючи вбудований доступ до табличного процесора, група деталей з різними розмірами й переліком елементів деталі. Розглянемо типові елементи пристосувань для обробки деталі типу «Корпус» (рис.) і

«Буксу» (рис.). та ін..

Рис. Пристосування для обробки деталей «Корпус»

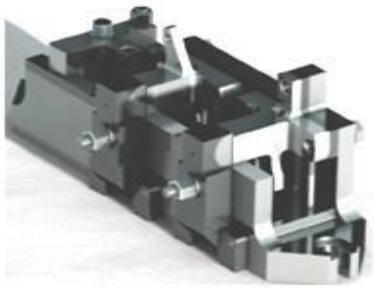


Рис. Пристосування для обробки деталей «Букса»

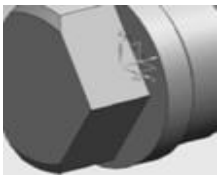


Рис. Деталь «Болт»

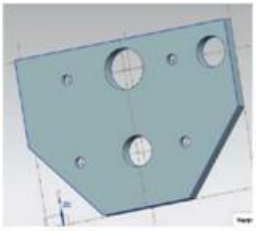


Рис. 6.

Рис. Деталь «Накладка»

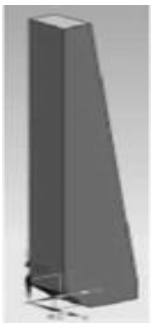


Рис. Деталь «Ромбічний палець»

Ромбічний палець — для фіксації певної орієнтації заготовки (рис. 7.10.).

Сімейство деталей формується на базі деталі- зразка з використанням вбудованої в них електронної таблиці для створення таблиці типових розмірів, що описує усе сімейство деталей. Завдяки цій опції можливе одержання нових деталей на основі уніфікації деталі, при зміні тільки необхідних параметрів уніфікованої деталі.

2.3.2. Загальний порядок побудови в програмі САПР NX

1. Створити деталь-зразок.

2. Визначити параметри, які будуть використовуватись для визначення членів сімейства.

3. Створити й зберегти таблицю параметрів, у якій задані значення параметрів для всіх майбутніх членів сімейства:

4. Відкрити електронну таблицю з діалогом створення сімейства за допомогою команди Електронна таблиця сімейства

— Створити.

5. З електронної таблиці виконати команду Сімейство - деталей — Створити частини для створення частин в них.

Створення бази даних типових елементів пристосування - для обробки деталей на металорізальних верстатах у програмі **САПР NX** скорочує час проектування пристосувань, що веде до зменшення собівартості розробки пристосувань, а, отже, і собівартості продукції. Крім того, уже на стадії проектування можливе усунення помилок, невідповідностей, проведення кінематичного аналізу пристосувань, аналізу навантаження деталей методом кінцевих елементів.

2.4. Розробка проекту меблів за допомогою САПР

Проектувати вироби з деревини нині можна не тільки на папері, а й на екрані комп'ютера за допомогою спеціалізованих програм, які наочно демонструють кожну деталь майбутнього виробу. Сьогодні вчителі технологій і профільного навчання повинні володіти прогресивними методами проектування, бути

«озброєні» сучасними ефективними інформаційними технологіями, зокрема отримати навички роботи в одній із систем САПР. В цій роботі ми спробуємо показати ряд можливостей графічної програми PRO100 у проектуванні виробів з деревини на прикладі розробки проекту тумби.

Аналіз останніх досліджень і публікацій. Для того, щоб проектувати вироби з деревини на комп'ютері використовують як спеціалізовані (PRO 100, bCAD-Мебельщик, Базис- Конструктор-Мебельщик, KitchenDraw, Astra, Woody) так і універсальні (T-FLEX, Mechanical Desktop, AutoCAD, 3D Studio Max) програми.

За допомогою спеціалізованих програм є можливість швидко і, головне, якісно

створювати будь-які конструкції виробів з деревини, розраховувати їх попередню вартість безпосередньо на екрані ПК. До того ж, конструктор отримує наочні ілюстрації проекту, детальні складальні креслення, а також креслення окремих деталей.

Впровадження САПР у процес навчання студентів проектуванню виробів з конструкційних матеріалів, зокрема деревини, є актуальним у сьогоденні. Аналіз літературних джерел дозволив встановити, що в цьому напрямку працюють ряд науковців. Так, Ю.І. Рудін стверджує, що дизайн і конструювання виробів з деревини нерозривно пов'язані зі спеціалізованими комп'ютерними програмами, зокрема bCAD- Мебельщик [5]. А. Стариков у своїх працях розглядає автоматизоване конструювання виробів корпусних меблів на основі САПР «bCAD для Мебельщика», «Базис-Конструктор- Мебельщик» [6]. Досвід використання програмного комплексу T-FLEX для параметричного проектування в меблевому виробництві наведено у статті П.В. Перфільєва [3].

Невирішені частини проблеми. Науковцями висвітлюються різноманітні проблеми впровадження САПР у процес навчання студентів проектуванню виробів з деревини. Однак, питання розробки конструкторської і технологічної документації на вироби з деревини в САПР PRO100 з

покроковими сценаріями поетапного виконання побудов залишаються невирішеними.

Мета дослідження - розглянути послідовність розробки проекту тумби засобами графічної програми PRO 100 майбутніми вчителями технологій і профільного навчання.

«Системи автоматизованого проектування в деревообробній промисловості» [1], де базовим графічним пакетом є САПР PRO 100.

Програма PRO100 польського походження застосовується на всіх етапах процесу виробництва меблів, де споживач хоче осучаснити свою роботу, спираючись на досягнення комп'ютерної техніки. Програма може застосовуватися для проектування меблів «з нуля», для створення власної електронної бібліотеки, для планування постачання у виробництві, для аранжування інтер'єрів, або ж, нарешті, для надання сприяння в процесі безпосереднього продажу - на кожному з цих етапів доступна візуалізація, різні типи видів, оцінка і рапорти. Простота

обслуговування (більшість операцій можна виконати за допомогою миші), швидкість дії, а також постійна можливість введення змін в проекті, значно полегшує життя виробникам і продавцям виробів з деревини [3].

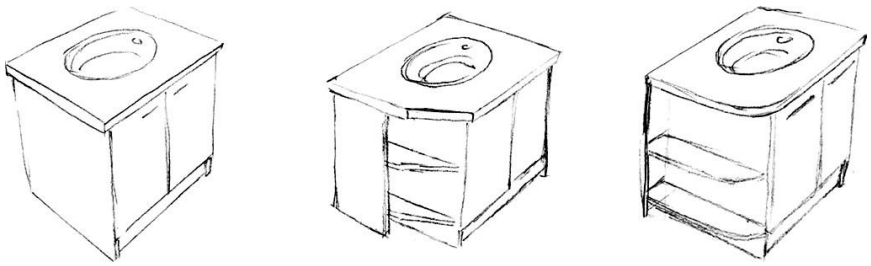
Створення нових виробів в деревообробній промисловості відбувається в такій послідовності: на основі аналізу продукції, що випускається, проектується нова, яка володіє більш високими естетичними, експлуатаційними або іншими властивостями, потім проводяться інженерні розрахунки і моделювання, технологічна підготовка виробництва, виготовлення і збут виробу. При цьому отримують замкнутий цикл, оскільки проектування нового виробу виконується на базі аналізу ринку і даних про ефективність, надійність і збут моделей, що випускаються.

Розглянемо приклад виконання лабораторно-практичної роботи «Розробка проекту тумби за допомогою САПР PRO 100». Мета роботи: навчитися проектувати різні типи тумб за допомогою САПР PRO 100 [1].

Згідно інструкції студентам ставляться такі завдання: опрацювати та законспектувати питання для контролю самопідготовки; підготувати ескізи тумби під мийку на кухню; затвердити ескіз тумби у викладача; спроектувати методом комбінування тумбу під мийку, згрупувати деталі об'єкту, надати відповідну текстуру елементам виробу; зберегти файл у базі бібліотеки «Мебель»; роздрукувати проект тумби під мийку; подати письмовий звіт за результатами виконаної роботи.

Завдання були визначенні з урахування особливості побудови тумби під мийку, конструкція якої має свої особливості. Вона полягає у виконанні вправ на конструювання отвору у кришці під мийку.

Ознайомившись зі специфікою конструкцій тумби студенти під час самостійної роботи виконують ескізи тумб (рис. 1).



а) б)

Рис. Ескізи тумб (авторська розробка)

Для проектування із запропонованих ескізів було обрано тумбу (рис. 7.1, в), яка скрадатиметься з двох частин: основна - тумба і додаткова - секція з поличками. В процесі деталізації було обрано такі параметри тумби: основний матеріал ДСП товщиною 16 мм; кришка ламінована ДСП товщиною 28 мм та габаритами 900х600; 2 бокові стінки - 695х510; 2 дверцята - 705х347; 2 перемички - 668х80; дно 1 шт. - 668х510; цокольна планка 1 шт. - 700х105 та 4 кутові задні елементи для підсилення конструкції з ДВП. Секція з поличками скрадатиметься з 3 поличок - 530х140, радіуси округлення на кінці кришки і поличках Я130; 1 бокової стінки - 810х533 і задньої стінки для утримання бокових поличок - 711х140. Утримуватиметься сама тумба на 4 ніжках, які регулюють по висоті.

Процедура позиціонування в процесі проектування може виявити і візуально показати недоліки в попередніх обрахунках, що дасть можливість виправити їх і в подальшому уникнути неточності у проекті конструкції.

Поетапність створення тумби:

1) створення бокових стінок та їх розміщення в просторі (позиціонування);
2) проектування дна тумби;
3) проектування двох перемичок, які з'єднують стінки тумби у верхній частині для надання жорсткості конструкції;
4) проектування ніжок;
5) розробка чотирьох кутових елементів тильної сторони тумби;
6) проектування цокольної планки;
7) створення дверцят тумби;
8) групування елементів тумби;
9) проектування кришки тумби (рис. 7.2); 10)створення отвору

для мийки в кришці;

11) імітація мийки з використанням бібліотеки готових об'єктів компонування окремої секції з поличками тумби згідно ескізу та попередніх розрахунків;

12) зміна і корективи кольору елементів конструкції, додавання ручок.

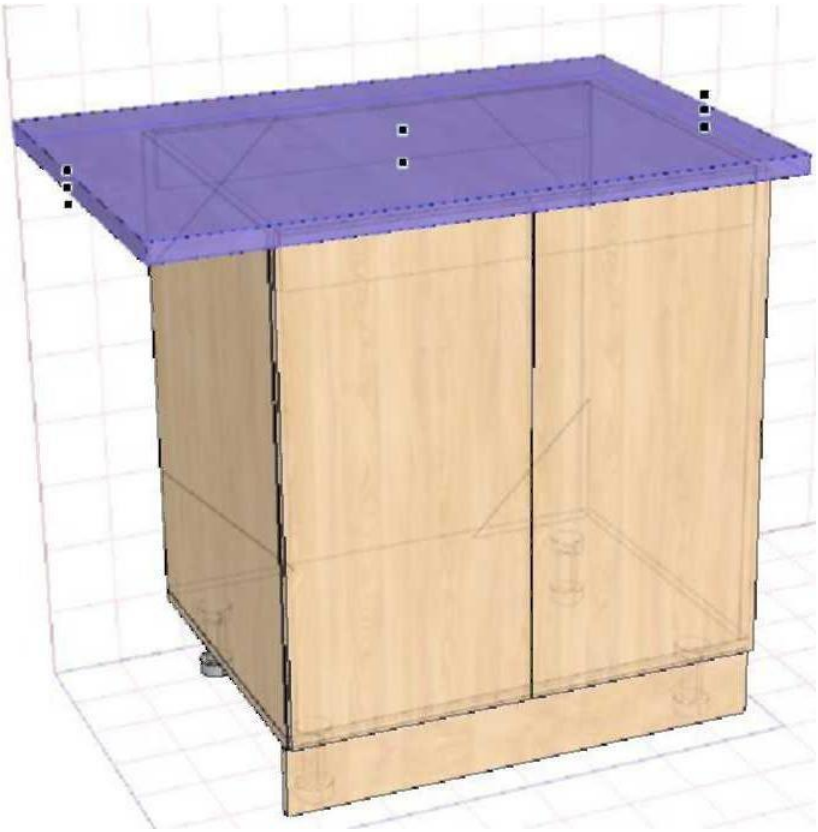


Рис. Проектування кришки тумби (авторська розробка)

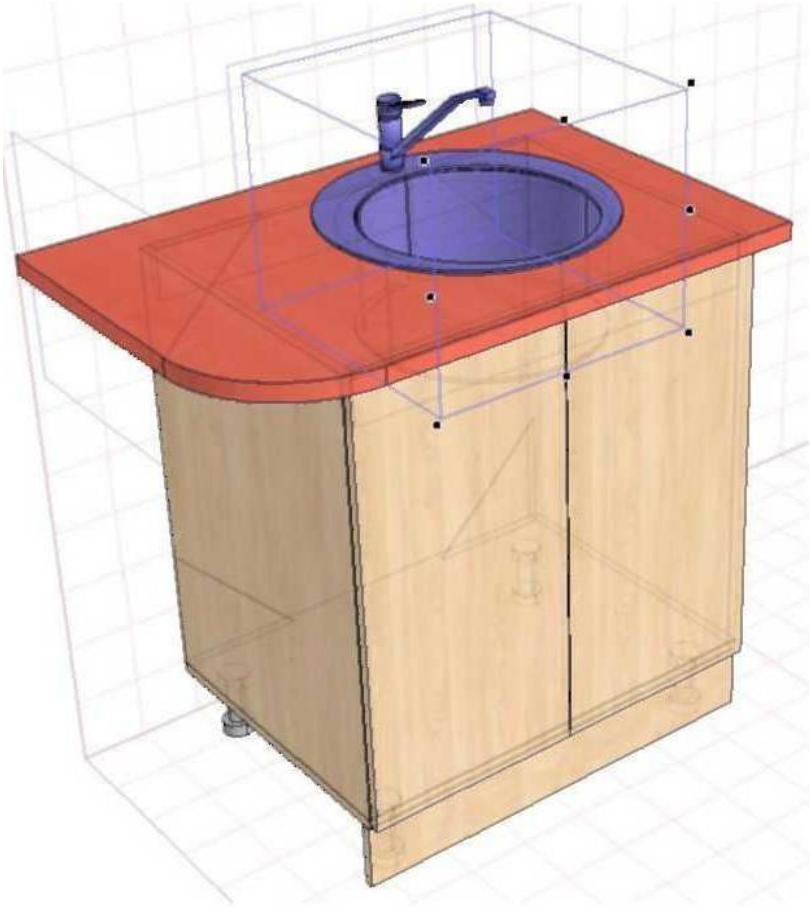


Рис. Компонування мийки у кришці тумби (авторська розробка)

Для кращої презентабельності проекту добавлено елемент освітлення та кольорове забарвлення стін і підлоги віртуального простору (рис. 7.4).

Рис. 7.4. Проект кухонної тумби під мийку(авторська розробка)

Проект тумби можна роздрукувати в будь-якому виді, яких у PRO 100 є 7 (перспектива, аксонометрія, вид зверху, вид спереду, вид справа, вид ззаду, вид

зліва) за допомогою опції

«Друк».

Слід відмітити, що проекти розроблених виробів практично реалізуються студентами в процесі виконання лабораторно-практичних занять з дисципліни «Практикум з проектування та виготовлення виробів з деревини», яка читається паралельно з дисципліною «Системи автоматизованого проектування в деревообробній промисловості» [2] (рис.).

Рис. Кухонна тумба під мийку (авторська розробка

2.5. Висновки розділу

Досвід експлуатації САПР PRO 100 показав, що ця програма досить легко опановується користувачами. При цьому значно скорочується час на випуск креслярської продукції, помітно підвищується її якість. PRO 100 в очах майбутніх вчителів технологій і профільного навчання стає зручним і зрозумілим інструментом, який дозволяє полегшити та пришвидшити процес виконання традиційних навчальних операцій.

РОЗДІЛ 3 РОЗРОБКА ПРОЕКТУ

3.1. Створення проекту та засоби, які використовувались у процесі роботи

В ході розробки використовувалась мова програмування C#, платформа ASP.NET, стандартизована мова html та бібліотека vue.js.

Платформа ASP.NET. ASP.NET є єдиною моделлю для розробки веб-додатків з застосуванням мінімуму коду, яка містить служби, необхідні для побудови веб-додатків. ASP.NET є частиною платформи .NET Framework, а тому забезпечує доступ до класів цієї платформи. Програми можуть бути написані будь-якою мовою середовища CLR, включаючи Microsoft Visual Basic, C#, JScript .NET і J#. Ці мови дозволяють розробляти програми ASP.NET, які можуть використовувати всі переваги середовища CLR, типової безпеки, наслідування і т. д.

В ASP.NET входить [10]:

Платформа для розробки сторінки та елементів управління

Компілятор ASP.NET

Інфраструктура захисту даних

Можливості по управлінню станом

Конфігурація

Спостереження і налаштування продуктивності

Підтримка налагодження

Платформа веб-служб XML

Розширювана середа розміщення та управління життєвим циклом додатки

Розширювана середовище конструктора

Структура сторінок та елементів управління ASP.NET - структура програмування, яка виконується на веб-сервері для динамічного створення та відображення веб-сторінок ASP.NET. Веб-сторінки ASP.NET можна переглядати в будь-яких браузерах або клієнтських пристроях, ASP.NET відображає розмітки (HTML) в запрошуваному браузері. Як правило, можна

використовувати одну і ту ж сторінку для різних оглядачів, так як ASP.NET відображає відповідну розмітку для запитувача браузера. Однак можна розробляти веб-сторінки ASP.NET для певних оглядачів, наприклад Microsoft Internet Explorer 6, і використовувати широкі можливості конкретного браузера. ASP.NET підтримує елементи керування для мобільних пристроїв, наприклад таких пристроїв веб-доступу, як смартфони, портативні комп'ютери і PDA.

Веб-сторінки ASP.NET є повністю об'єктно-орієнтованими. На сторінках ASP.NET з елементами HTML можна працювати, використовуючи властивості, методи і події. Структура сторінок ASP.NET надає єдину модель відгуку на клієнтські події в коді, що виконується на сервері, тому реалізація поділу клієнта і сервера, що використовується у веб-додатках, не потрібна. Вона також автоматично обробляє стани сторінки та її елементи керування під час циклу обробки сторінки.

Структура сторінок та елементів управління ASP.NET також інкапсулює загальні функціональні можливості користувальницького інтерфейсу в зручні повторно використовувані елементи управління. Елементи управління, написані одного разу, можна використовувати в багатьох сторінках. Вони вбудовуються в веб-сторінки ASP.NET, на якій вони розміщуються під час відтворення.

Структура сторінок та елементів управління ASP.NET також надає можливості управління відображенням і поведінкою веб-вузла за допомогою тем і обкладинок. Можна визначити теми і обкладинки і потім застосувати їх на рівні сторінки або елементу управління.

Окрім тйього можна визначити головні сторінки, що дозволяють створити макет сторінки, який можна буде використовувати для всіх сторінок у додатку. Одна головна сторінка макету визначає і поведінку, яку можна використовувати для всіх сторінок (або групи сторінок) в додатку. Потім можна створити окремі сторінки, що включають зміст, пов'язаний зі сторінкою, яку слід відображати. Коли користувачі запитують сторінку

змісту, вихідна сторінка являє собою поєднання структури головної сторінки і змісту зі сторінки змісту.

Весь код ASP.NET компілюється, що дозволяє використовувати строгий контроль типів, оптимізації продуктивності, раннє зв'язування і інші переваги. Після компіляції коду середовище CLR компілює код ASP.NET в машинний код, тим самим забезпечуючи підвищення продуктивності [8].

В ASP.NET включений компілятор, що виконує компіляцію всіх компонентів програми, включаючи сторінки і елементи управління, в збірку, яку середа розміщення ASP.NET може використовувати в подальшому для обслуговування запитів користувача.

Крім можливостей захисту даних .NET, ASP.NET надає додаткову інфраструктуру для автентифікації та авторизації доступу користувачів, а також інших завдань безпеки. Можна виконувати автентифікацію за допомогою перевірки автентичності Windows, що надається службами IIS, або за допомогою власної бази даних користувача, використовуючи перевірку автентичності форм ASP.NET і членство ASP.NET. Також можна керувати перевіркою достовірності веб-додатка за допомогою груп Windows або власної бази даних ролей, використовуючи ролі ASP.NET. Ці схеми легко додати, видалити або замінити в залежності від вимог до додатка.

Додатки ASP.NET використовують систему конфігурації, що дозволяє визначати параметри конфігурації для веб-сервера, веб-сайту та окремих додатків. Параметри конфігурації можна застосовувати в момент першого розгортання додатків ASP.NET, а також у будь-який момент додавати або переглядати параметри конфігурації з мінімальним впливом на працюючі веб-додатки та сервери. Параметри конфігурації ASP.NET зберігаються у файлах XML. Так як ці XML-файли є текстовими ASCII-файлами, які можна читати і змінювати, вносити зміни в конфігурацію веб-додатка нескладно. Можна розширити схему конфігурації у відповідності зі своїми вподобаннями.

Мова програмування C#. Хоча платформа .NET і передбачає можливість використання різних мов програмування (наприклад, C++ та Visual Basic), спеціально для неї група програмістів фірми Microsoft розробила нову мову програмування, яка одержала назву C#. Ця мова дозволяє у повному обсязі скористатися можливостями, що надає технологія .NET. Фактично, C# являє собою гібрид різних мов програмування, від кожної з яких вона взяла найкраще [12].

Будучи спадкоємицею мови C++, мова C# використовує подібний до неї синтаксис, проте позбавлена неоднозначностей, які допускали компілятори C++. Багато спільного має також із мовою Java, яка створювалась спеціально як мова, що дозволяє реалізовувати застосування, які можуть використовуватись в різних операційних системах. Її створення як раз і було намаганням розв'язати так звану проблему «перенесення» комп'ютерних програм в інше операційне середовище. Ця можливість була реалізована за рахунок компіляції у два етапи. На першому етапі Java-компілятор створює машиннонезалежний так званий «байт-код». Цей байт-код виконується віртуальною машиною Java (JVM – Java Virtual Machine). Вона являє собою спеціальну операційну систему. Отже, програма мовою Java може бути виконана на будь-якій платформі, де реалізовано JVM. А оскільки така реалізація була відносно не складною, то JVM були достатньо швидко та успішно реалізовані у досить різноманітних програмних середовищах. Проте мова Java не вирішила проблему міжмовної взаємодії в програмах.

На відміну від Java, C# генерує код, що може бути використаний лише у середовищі виконання .NET – так званий керований код (managed code). Зокрема, це передбачає автоматичне керування пам'яттю та відсутність потреби працювати напряму із вказівниками на адреси у пам'яті, що дуже зменшує кількість ймовірних помилок при розробці та використанні програми.

Ще одна приємна риса, яка відрізняє мову C# – можливість одночасно створювати і програму, і документацію до неї у форматі XML. Для цього

треба лише використовувати спеціальний тип коментарів та після завершення роботи згенерувати окремий файл із документацією до програми (ця можливість вбудована у середовище розробки.)

Компонент .NET Microsoft Visual Studio – це інтегроване середовище розробки програмного забезпечення, яке забезпечує абсолютно комфортний інтерфейс для створення C#-програм.

C# створювалася як мова компонентного програмування, і в цьому одне з головних переваг мови, спрямоване на можливість повторного використання створених компонентів. З інших об'єктивних факторів відзначимо наступні:

C# створювався паралельно з каркасом Framework .Net і повною мірою враховує всі його можливості - як FCL, так й CLR;

C# є повністю об'єктно-орієнтованою мовою, де навіть типи, вбудовані в мову, представлені класами;

C# є потужною об'єктною мовою з можливостями спадкування й універсалізації;

C# є спадкоємцем мов C/C++, зберігаючи кращі риси цих популярних мов програмування;

завдяки каркасу Framework .Net, що стали надбудовою над операційною системою, програмісти C# одержують ті ж переваги роботи з віртуальною машиною, що й програмісти Java. Ефективність коду навіть підвищується, оскільки виконавче середовище CLR являє собою компілятор проміжної мови, у той час як віртуальна Java-машина є інтерпретатором байта-коду;

потужна бібліотека каркасів підтримує зручність побудови різних типів додатків на C#, дозволяючи легко будувати Web-служби, інші види компонентів, досить просто зберігати й одержувати інформацію з бази даних й інших сховищ даних;

реалізація, що сполучає побудову надійного й ефективного коду, є немаловажним чинником, що сприяє успіху C#.

HTML (HyperText Markup Language - мова розмітки гіпертекстових документів) - стандартна мова розмітки веб-сторінок в Інтернеті. Більшість веб-сторінок створюються за допомогою мови HTML (або XHTML). Документ HTML оброблюється браузером та відтворюється на екрані у звичному для людини вигляді.

HTML є похідною мовою від SGML, успадкувавши від неї визначення типу документа та ідеологію структурної розмітки тексту.

Попри те, що HTML - штучна комп'ютерна мова, вона не є мовою програмування.

HTML разом із каскадними таблицями стилів та вбудованими скриптами - це три основні технології побудови веб-сторінок.

HTML впроваджує засоби для:

створення структурованого документа шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше;

отримання інформації із Всесвітньої мережі через гіперпосилання;

створення інтерактивних форм;

включення зображень, звуку, відео, та інших об'єктів до тексту.

Розмітка в HTML складається з чотирьох основних компонентів: елементів (та їхніх атрибутів), базових типів даних, символьних мнемонік та декларації типу документа.

Елементи HTML. Елементи являють собою базові компоненти розмітки HTML. Кожен елемент має дві основні властивості: атрибути та зміст (контент). Існують певні настанови щодо кожного атрибута та контента елемента, які треба виконувати задля того, щоб HTML-документ був визнаний валідним.

У елемента є початковий тег, який має вигляд <element-name>, та кінцевий тег, який має вигляд </element-name>. Атрибути елемента записуються в початковому тегу одразу після назви елемента, контент елемента записується між його двома тегами. Наприклад: <element-name element-attribute="attribute-value">контент елемента</element-name>.

Деякі елементи, наприклад br, не містять контенту, тож і не мають кінцевого тега. Елемент може не мати початкового та кінцевого тега (наприклад, елемент head), проте він завжди буде представлений в документі. Нижче зазначені деякі типи елементів розмітки HTML.

Елементи структурної розмітки застосовуються задля опису семантики тексту, іншими словами ці елементи описують призначення тексту свого контенту. Вони не зазначають ніякого спеціального (візуального) відтворення тексту, проте більшість браузерів мають стандартні стилі форматування для кожного елемента. Для подальшого стилізування тексту рекомендується використовувати Каскадні таблиці стилів (CSS) [3].

Бібліотека jQuery. jQuery - це одна з найпопулярніших бібліотек JavaScript. Її призначення - спростити і прискорити розробку скриптів, які працюють з елементами сторінки. Крім того, з нею спрощуються і інші важливі операції, наприклад ajax-запити.

Бібліотека jQuery не є самостійною мовою. Вона являє собою функцію, написану на JavaScript. Все, що дозволяє робити jQuery, можна зробити і на чистому JavaScript, але в деяких випадках це може зайняти кілька днів.

Окремі сторінки сайтів, в основному мають інформативний характер, не потребують jQuery. Але основні сторінки сайтів вже не можуть обійтися без його використання.

Щоб надати сторінці динамічності, постійно доводиться виконувати однотипні операції: пошук необхідних елементів і вчинення дій над ними. Бібліотека jQuery містить готові функції, використання яких дозволяє виконувати стандартні операції набагато простіше і швидше. Всі функції оптимізовані для розробки сайтів, тому бажаний результат досягається буквально в декілька дій. Від користувача потрібно лише вибрати потрібний елемент і застосувати до нього бажаний метод.

Назви функцій і методів дуже короткі, але вони максимально точно описують операції, що виконуються. Короткі назви зменшують кількість

помилок при написанні коду, а також дозволяють практично не звертатися до різних інформерам для пошуку потрібної функції.

Кросбраузерність сайту повинна бути не тільки в зовнішньому оформленні, але і при роботі скриптів. JavaScript вбудований в кожен браузер, але не всі можливості, описані в стандарті JavaScript, реалізовані в них однаково. Один і той же код може працювати по-різному в різних браузерах. Функції jQuery реалізовані таким чином, щоб забезпечувалася кросбраузерність.

Процес розробки програми можна розділити на наступні етапи:

1.Створення технічного завдання та предметної області для проекту. На цьому етапі було проведено дослідження предметної області розробки програми, було виявлено, яка інформація необхідна для користувачів системи. Також на основі потреб користувачів було виділено функціонал, який повинний бути реалізований в програмному продукті

2.створення структури проекту. На цьому етапі було розроблено базуданих та серверну модель.

3.реалізація серверної сторони проекту.

4.реалізація клієнтської частини. Було зроблено верстку клієнтську логіку

5.тестування

6.супроводження та доопрацювання проекту

Програма складається з таких основних частин:

1. Адміністративна частина
2. Клієнтська частина

Адмінастративна частина програми має 5 основних підрозділів:

1. Розділ замовлень. В даній частині даються замовлення клієнтам.
2. Розділ редагування виробу . В даному розділі можливо створювати виріб
3. Список користувачів системи. В даному розділі Адміністратор системи може добавити або видалити користувача.

4. Список виробів. Вданому розділі реалізована можливість додати виріб та перевірити етап його створення на виробництві.

5. Список статусів замовлення. В даному розділі можна побачити статус замовлення виробу.

Клієнт має доступ лише до одного розділу, де він може подивитися свій виріб.

Проект створенно на платформі asp .net з використаням фреймворка MVC (model view controller)

На рисунку 3.1 представлено структуру проекту

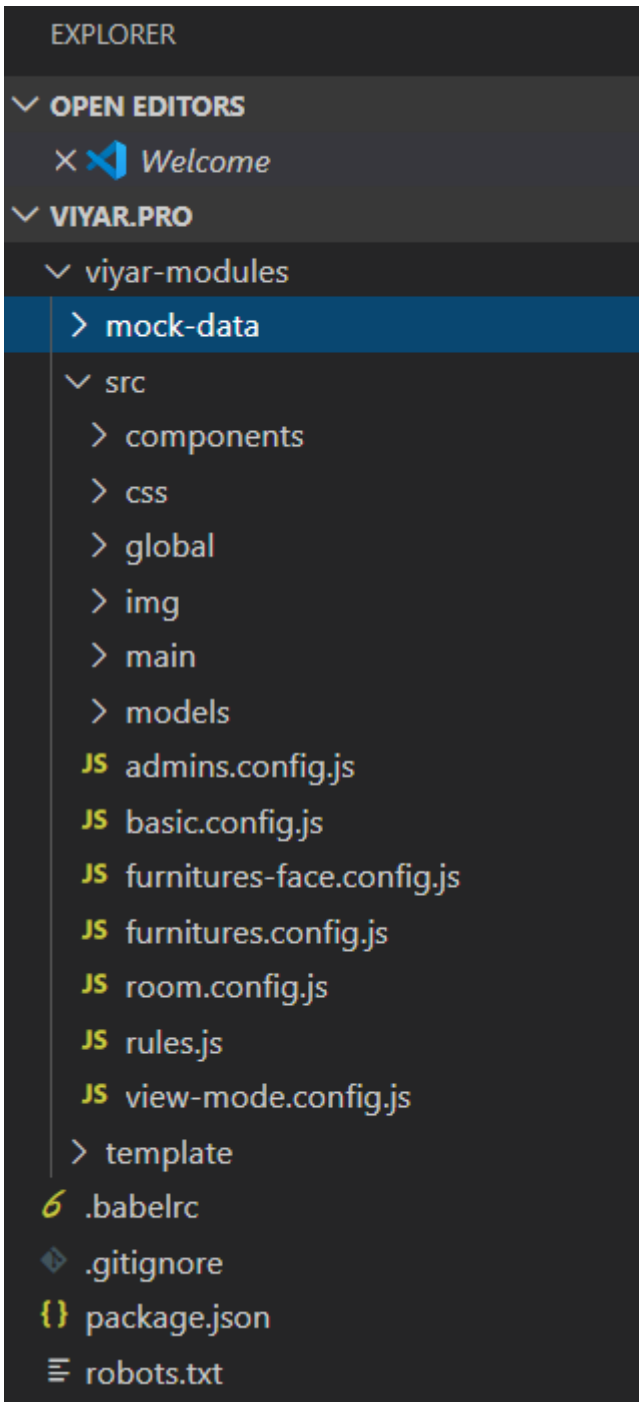


Рис. 3.1. Структура проекту

Проект складається з двох частин:

Серверної частини (Models та controlers)

Клієнтської частини (VIEW (верстка та клиенская логіка)

Нижче представлено структуру серверної частини:

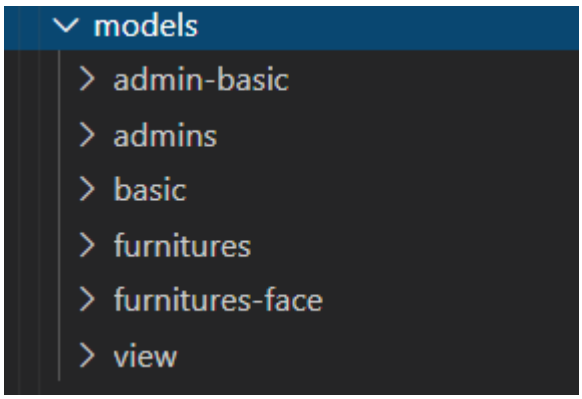


Рис. 3.2. Структура серверної частини

Розглянемо кожний структурний елемент серверної частини проекту.

1) Admin-basic. Модель відповідає за реалізацію основних модулів системи.

2) Admins. Модель відповідає за адміністраторську частину.

3) Basic. Модель відповідає за клієнтську частину створення приміщення.

4) Furnitures. Модель відповідає за частину створення фурнітури (внутрішньої).

5) Furnitures. Модель відповідає за частину створення фурнітури (внутрішньої).

6) Furnitures-Face. Модель відповідає за частину створення фурнітури (зовнішньої).

7) View. Модель відповідає за клієнтську частину створення виробів.

Розглянемо структуру Controllers

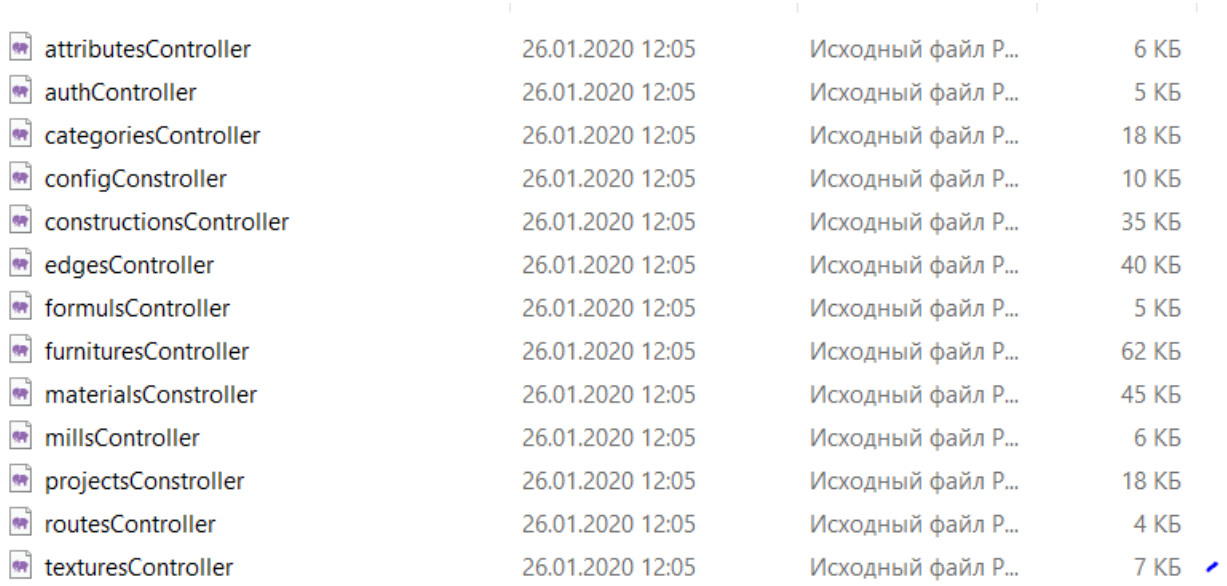


Рис. 3.3. Структура Controllers

1) AttributesController – контролер, що відповідає за реалізацію аттрибутів сутностей і виконання над ними різноманітних операцій (додавання, видалення, редагування, зміна прав та ролей і т. д)

2) AuthController – контролер, що відповідає за авторизацію

4) CategoriesController – контролер, що відповідає за реалізацію категорій сутностей і виконання над ними різноманітних операцій (додавання, видалення, редагування, зміна прав та ролей і т. д)

5) MaterialsController – контролер, відповідальний за матеріали

6) EdgesController – контролер, відповідальний за кромки

7) FurnituresController – контролер, відповідальний за фурнітуру

8) TexturesController – контролер, відповідальний за текстури

9) ConfigController – контролер, відповідальний за конфігурації

10) FormulsController – контролер, відповідальний за формули

11) MillsController – контролер, відповідальний за фрезеровки

12) ConstructionsController – контролер, відповідальний за конструкції

13) ProjectsController – контролер, відповідальний за проекти

Нижче представлено структуру VIEW

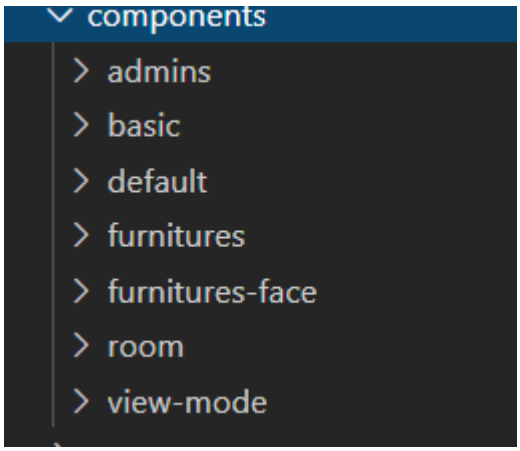


Рис. 3.4. Структура VIEW

1) Default. компонента відповідає за реалізацію основних модулів системи.

2) Admins. компонента відповідає за адміністраторську частину.

3) Basic. компонента відповідає за клієнтську частину створення приміщення.

4) Furnitures. компонента відповідає за частину створення фурнітури (внутрішньої).

5) Furnitures. компонента відповідає за частину створення фурнітури (внутрішньої).

6) Furnitures-Face. компонента відповідає за частину створення фурнітури (зовнішньої).

7) View. компонента відповідає за клієнтську частину створення виробів.

Загальна конфигурація проекта представлена нижче:

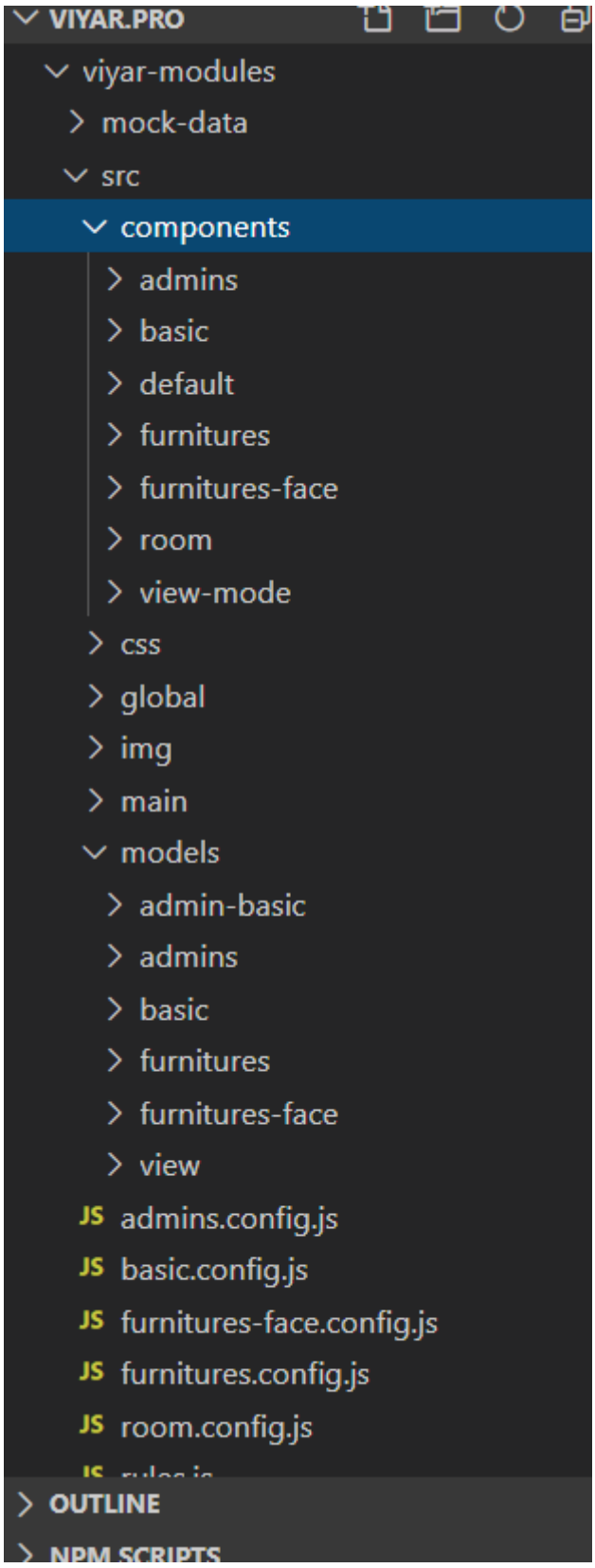


Рис. 3.5. Загальна конфигурація проекта

Файл Web Config - головний файл конфігурації проекту, який налаштовує всю систему.

Нижче представлено структуру файлу для запуску конфігурації

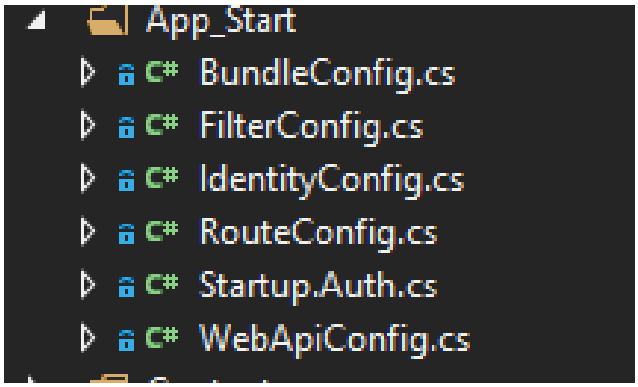


Рис. 3.6. Структура файлу для запуску конфігурації

BundleConfig – відповідає за зв’язки конфігурації

FilterConfig – відповідає за конфігурацію фільтрів

IdentityConfig – відповідає за конфігурацію ідентифікації

RouteConfig – відповідає за конфігурацію кореневої структури

Startup.Auth – відповідає за конфігурацію аутентифікації

WebApiConfig – відповідає за конфігурацію додатка на сервері.

3.2. Висновки до розділу

У даному розділі аналізуються етапи розробки та інструменти, які використовувались у процесі роботи. В ході розробки використовувалась мова програмування C#, платформа ASP.NET, стандартизована мова html та бібліотека jquery.

Процес розробки програми можна розділити на наступні етапи: 1. Дослідження. На цьому етапі було проведено дослідження предметної області розробки програми, було виявлено, яка інформація необхідна для користувачів системи. 2. Створення форми. Даний етап було присвячено розробці інтерфейсів програмного продукту. 3. Розробка програми.

Програма складається з таких основних частин: 1. Адміністративна частина; 2. Клієнтська частина. Адмінастративна частина програми має 5

основних підрозділів: 1. Розділ замовлень. В даній частині даються замовлення клієнтам. 2. Розділ редагування адреси доставки. В даному розділі можливо редагувати, змінювати адреси доставки 3. Список користувачів системи. В даному розділі Адміністратор системи може добавити або видалити користувача. 4. Список завдань. Вданому розділі реалізована можливість додати завдання та превірити звітність по його виконанню. 5. Список статусів замовлення. В даному розділі можна побачити статуси замовлення.

РОЗДІЛ 4
РОБОТА З ПРОГРАМОЮ

4.1. Вимоги до технічного та програмного забезпечення

Для повноцінного функціонування програми на персональному комп’ютері необхідні ті ж самі вимоги, що і до функціонування браузера. А саме:

Вимоги до Windows:

Операційна система: Windows XP з пакетом оновлення 2 +; Windows Vista; Windows 7; Windows 8; Windows 10.

Процесор: Intel Pentium 4 / Athlon 64 або більш пізньої версії з підтримкою SSE2

Вільне місце на диску: 350 Мб

Оперативна пам'ять: 512 Мб.

Вимоги до Mac:

Операційна система: Mac OS X 10.6 або більш пізньої версії

Процесор: Intel Pentium 4 / Athlon 64 або більш пізньої версії з підтримкою SSE2

Вільне місце на диску: 350 Мб

Оперативна пам'ять: 512 Мб.

Вимоги до Linux:

Операційна система: Ubuntu 10.04 +, Debian 6 +, OpenSuSE 11.3 +, Fedora Linux 14

Процесор: Intel Pentium 4 / Athlon 64 або більш пізньої версії з підтримкою SSE2

Вільне місце на диску: 350 Мб

Оперативна пам'ять: 512 Мб.

4.2. Робота з програмою користувачем

Для запуску програми потрібно зайти на стартову сторінку та ввести логін та пароль користувача. С цієї сторінки користувач попаде або в адміністративну частину, або в клієнтську частину.

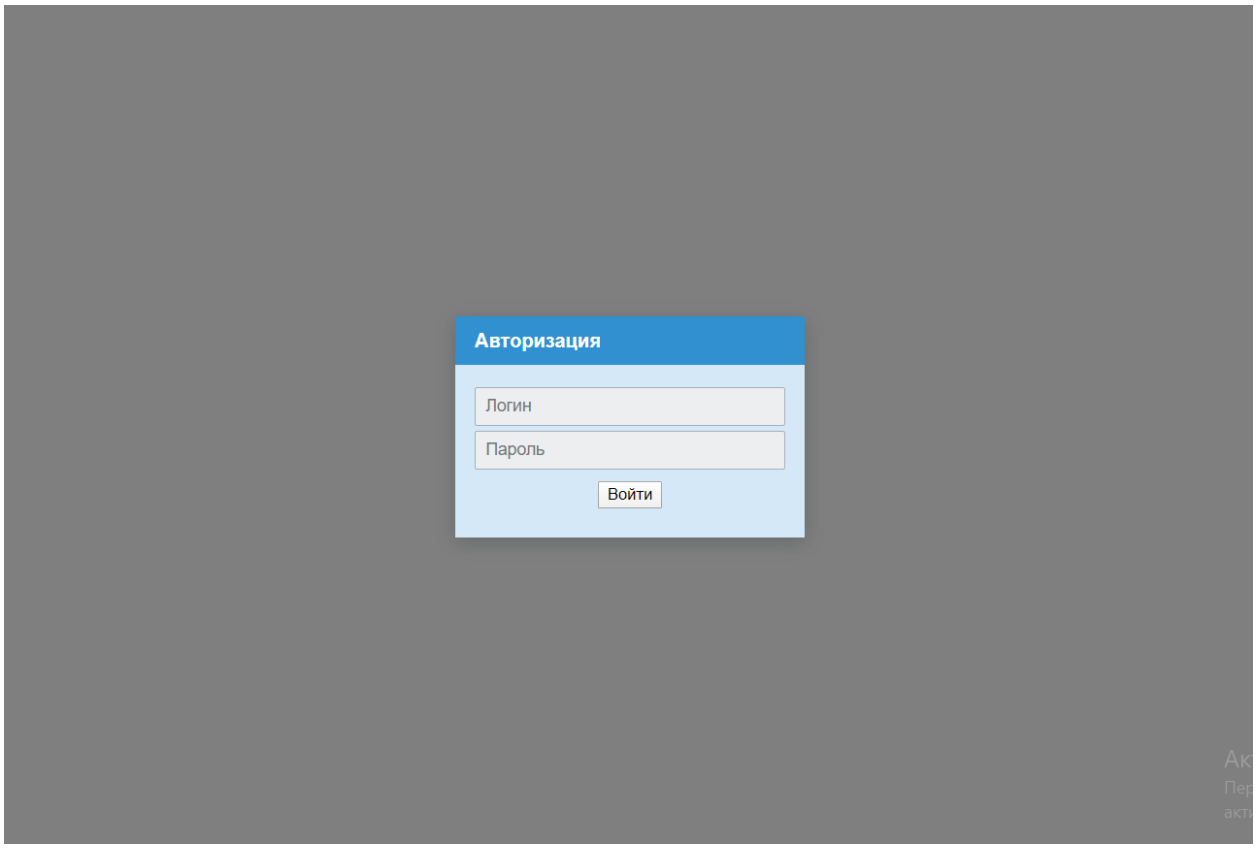


Рис. 4.1. Стартова сторінка

З адміністративної частини створюються вироби.

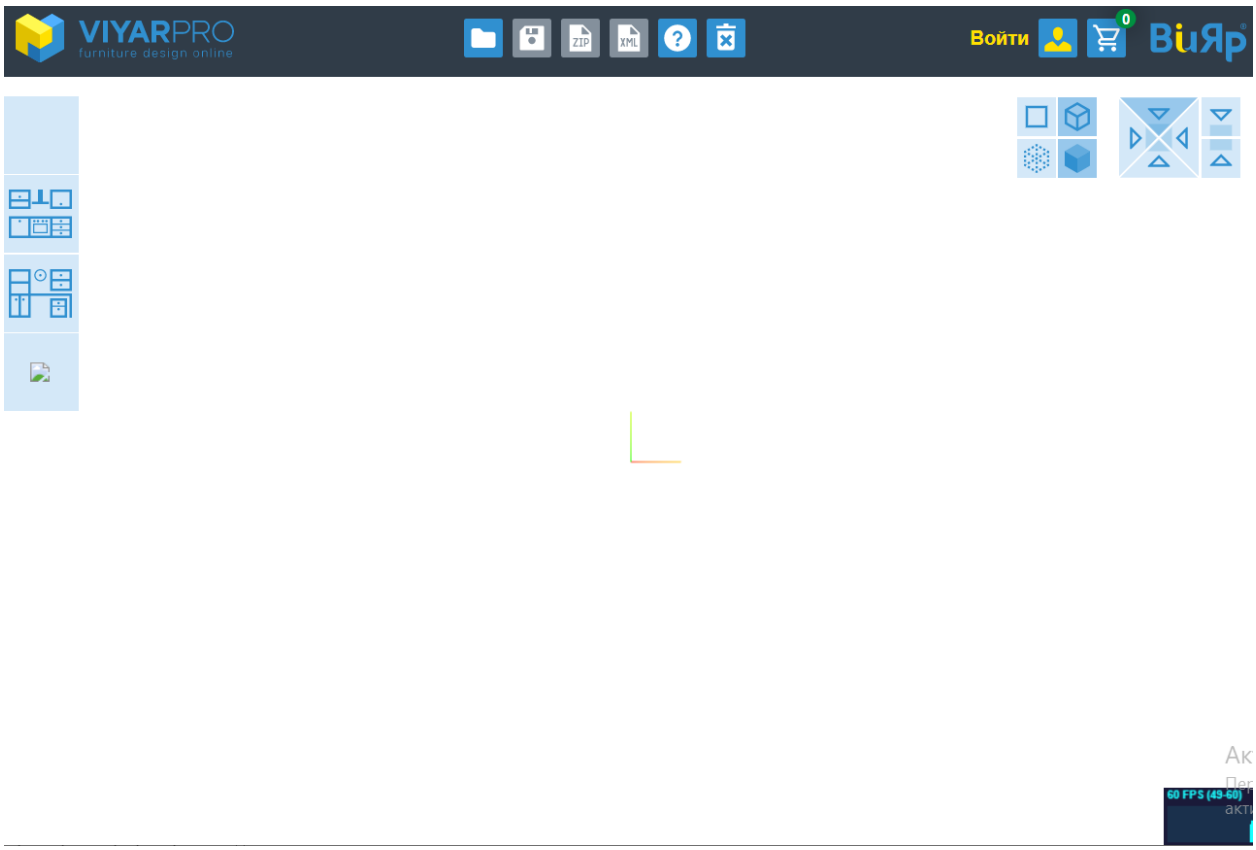


Рис. 4.2. Замовлення

Розділ, де редагується список виробів має вигляд, представлений на рисунку 4.3.

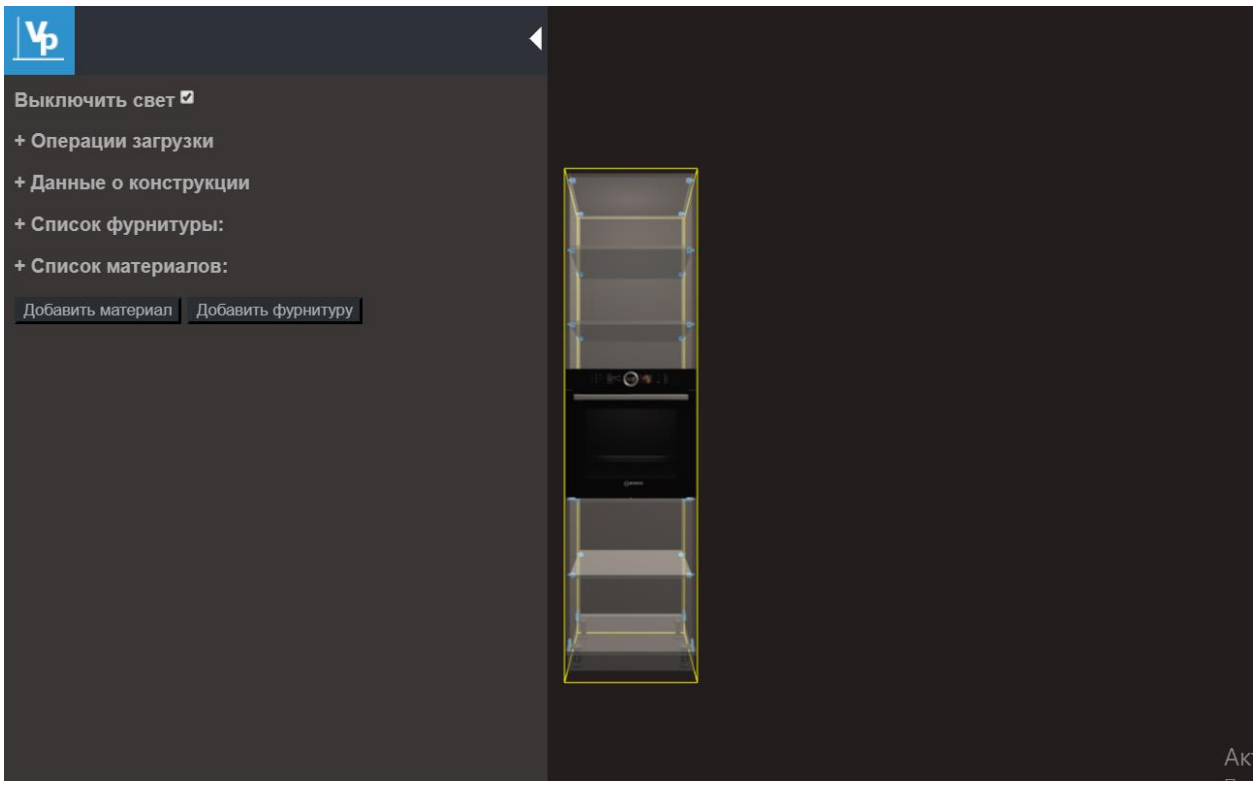


Рис. 4.3. Розділ, де редагується виріб

На рисунку нижче представлено адмінську частину , де створюється виріб

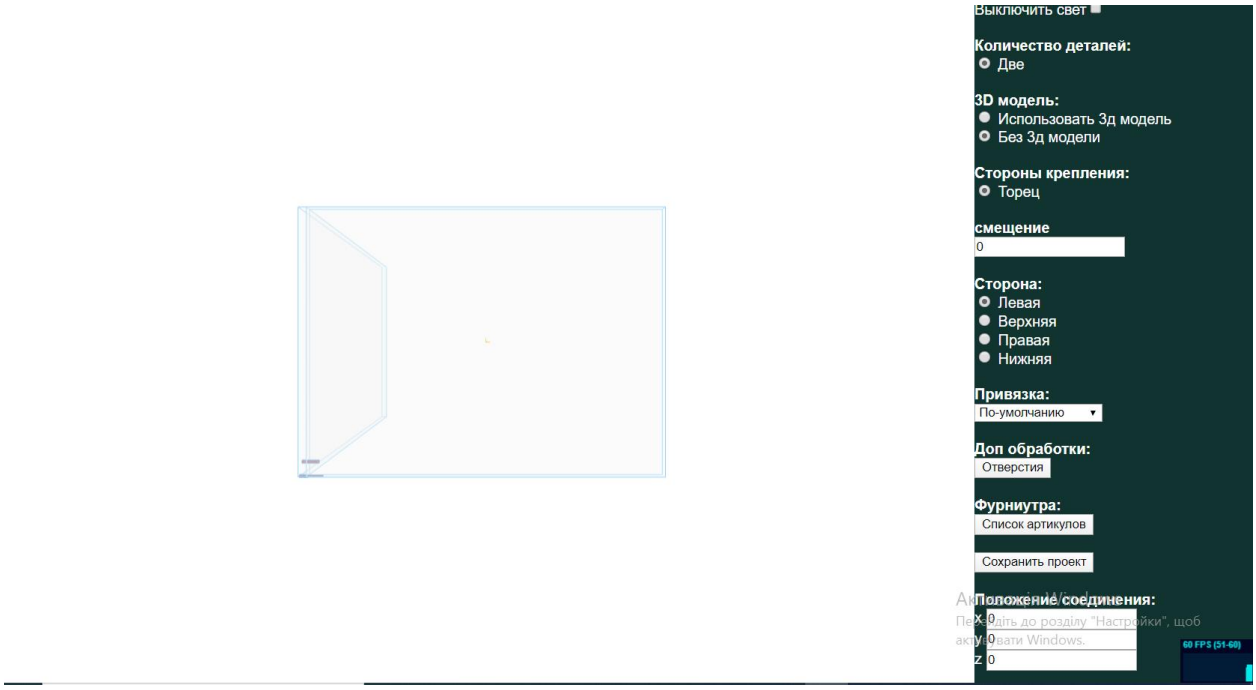


Рис. 4.4. На рисунку нижче представлено адмінську частину , де створюється фурнітура

Розділ фурнітури, де можна додати створити нову фурітуру та додати її до будь – якого виробу.

Рис. 4.5. Режим приміщення, де можна створити кухню

4.3. Висновки розділу

У розділі описано як працювати з програмою. Також показано скриншоти програми з вінками при різних діях користувача. Указано, які системні вимоги для комп’ютера, на якому буде запущена програма.

ВИСНОВКИ

Служба доставки у наші дні є найбільш поширеним і оптимальним варіантом перевезення вантажів і кореспонденції від відправника до одержувача. Послугами цих компаній користуються не тільки організації, зацікавлені у швидкій доставці ділової кореспонденції своїм партнерам, але і приватні особи, які користуються послугами кур'єра для надійної доставки особистих відправлень.

Завдання такої служби полягає у своєчасній і якісній доставці замовлень одержувачу. Для цього їм необхідна єдина інформаційна система, в якій буде відслідковуватися надходження і подальша доставка всіх замовлень.

Зараз багато компаній, які надають транспортно-експедиторські та кур’єрські послуги, і служби експрес-доставки в тому числі, знаходяться на тій стадії розвитку, коли повністю усвідомлена необхідність впровадження інформаційних технологій. При цьому розвиток IT-інфраструктури компанії починають розуміти стратегічно, як одну з основних складових майбутнього успіху компанії, як джерело конкурентних переваг.

Сучасні інформаційні технології забезпечують можливість для ефективного аналізу процесів, підготовки і представлення результатів для подальшого прийняття рішень. Застосування сучасних інформаційних технологій дозволяє підвищити ефективність доставки за рахунок можливості швидкого доступу до інформації про суб'єкти (покупець, перевізник, термінал) і об'єкти (товари, послуги) доставки.

Косплатформність - властивість програмного забезпечення працювати більш ніж на одній програмній (в тому числі - операційній системі) або апаратній платформи, та технології, що дозволяють досягти такої властивості. Кросплатформність дозволяє суттєво скоротити витрати на розробку нового або адаптацію існуючого програмного забезпечення.

Залежно від засобів реалізації поділяється на кросплатформність на рівні мов програмування (а також інструментів таких мов: компіляторів та редакторів зв'язків), середовища виконання, операційної системи та апаратної платформи.

Для підвищення можливостей переносимості може бути використано багатоплатформове забезпечення, що дозволяє працювати більш ніж на одній платформі. Використання кросплатформених засобів розробки (компіляторів, бібліотек і фреймворків) є одним із шляхів підвищення переносимості програмного коду.

Процес складання програми під платформу, відмінну від платформи, на якій запущено компілятор і компонувальник, називається кросскомпіляцією.

Процес розробки програми можна розділити на наступні етапи: 1. Дослідження. На цьому етапі було проведено дослідження предметної області розробки програми, було виявлено, яка інформація необхідна для користувачів системи. 2. Створення форми. Даний етап було присвячено розробці інтерфейсів програмного продукту. 3. Розробка програми.

Програма складається з таких основних частин: 1. Адміністративна частина; 2. Клієнтська частина. Адмінастративна частина програми має 5 основних підрозділів: 1. Розділ замовлень. В даній частині даються замовлення клієнтам. 2. Розділ редагування адреси доставки. В даному розділі можливо редагувати, змінювати адреси доставки 3. Список користувачів системи. В даному розділі Адміністратор системи може добавити або видалити користувача. 4. Список завдань. Вданому розділі реалізована можливість додати завдання та превірити звітність по його виконанню. 5. Список статусів замовлення. В даному розділі можна побачити статуси замовлення.

У четвертому розділі роботи описано як працювати з програмою. Також показано скриншоти програми з вінками при різних діях користувача. Указано, які системні вимоги для комп’ютера, на якому буде запущена програма.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Аналіз ринку кур'єрської доставки в Україні [Електроний ресурс]. – Режим доступу: https://pro-consulting.ua/ua/issledovanie-rynka/Analiz-rynka-kurerskoi-ekspress-dostavki-Ukrainy

2. Программы для служб доставки [Електроний ресурс]. – Режим доступу: http://www.livebusiness.ru/tools/delivery

3. Шикуть А.В. К вопросу о переносимости кода и некоторых возможностях использования кроссплатформенного программного обеспечения. *Инженерный журнал: наука и инновации*, 2013, вып. 6. URL: http://engjournal.ru/catalog/it/hidden/817.html

4. Bertrand Meyer. Approaches to portability. JOOP (Journal of Object-Oriented Programming), vol. 11, N 6, July-August 1998, pp. 93–95.

5. Tanenbaum A.S. Structured computer organization 5th Print. Amazon Prentice Hall, 2005, 800 p.

6. Hook B. Write portable code: an introduction to developing software for multi- ple platforms. No Starch Press, 2005, 248 p.

7. Переносимость. alice.pnzgu.ru/~dvn/uproc/books/site_tarasov/c15_port.html

8. Шилдт Герберт. Полный справочник по C. Москва, Издательский дом «Вильямс», 2005, 704 с.

9. Э. Фримен, Э. Фримен. Изучаем HTML, XHTML и CSS = Head First HTML with CSS & XHTML. - П.: «Питер», 2010. - 656 с.

10. Эд Титтел, Джефф Ноубл. HTML, XHTML и CSS для чайников, 7-е издание = HTML, XHTML & CSS For Dummies, 7th Edition. - М.: «Диалектика», 2011. - 400 с.

11. Адам Фримен. jQuery для профессионалов = Pro jQuery. - М.: «Вильямс», 2012. - 960 с.

12. Джейсон Ленгсторф. PHP и jQuery для профессионалов = Pro PHP and jQuery. - М.: «Вильямс», 2010. - С. 352.

13. Самков Г. jQuery. Сборник рецептов. - СПб.: БХВ-Петербург, 2010. - С. 416.

ДОДАТКИ

Додаток А
Код програми