МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Бакалаврська кваліфікаційна робота

на тему: «Розробка мобільного додатку з підбору комп'ютерного обладнання»

Виконав студент 4 курсу групи К-25
Спеціальність 122 комп'ютерні науки
Юліанчук Віктор Андрійович

Керівник асистент
Штефан Наталія Зінов'ївна

Консультант к.т.н., доцент
Великодний Станіслав Сергійович

Рецензент д. т. н., професор
Мещеряков Володимир Іванович

Одеса 2020

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ПК – персональний компютер

ОС – операційна система

СУБД – систем управління базами даних

ACW – Android Callable Wrappers

ART – Android Runtime

EF Core – Entity Framework Core

LINQ – Language Integrated Query

IL – Intermediate Language

WPF – Windows Presentation Foundation

UML – Unified Modeling Language

UWP – Universal Windows Platform

Діаграми Use-Case – діаграма варіантів використання

Action – дії на діаграмі діяльності

ASP.NET Core – технологія для створення веб-додатків

C # – мова програмування

Flow of events – потоком подій

Microsoft SQL Server – систем управління базами даних

ORM-інструментом – object-relational mapping – відображення даних на реальні об'єкти

Tag Helper – тег-хелпери

Web API – інтерфейс прикладного програмування для веб-серверу

Windows Forms – платформа Visual Studio

Xamarin.Forms – Android-додаток

ВСТУП

Збирання ПК не така складна, як здається – її часто називають «LEGO для дорослих». Проте проблеми сумісності, величезний асортимент брендів і самих комплектуючих для ПК можуть налякати початківців.

Пошук набору онлайн-інструментів, які допоможуть підібрати сумісні між собою комплектуючі та порівняти ціни в максимально простій формі є головною метою для багатьох покупців.

Лякаючий момент в створенні ПК – це нескінченні комбінації апаратного забезпечення. Той, хто не любить текстові керівництва, або не хоче розбиратися в тонкощах питання, може покластися на думку фахівця.

Як показує практика, можливості власного комп'ютера влаштовують користувачів лише до пори до часу – рано чи пізно доводиться вдаватися до апгрейду. Все банально просто – нові версії ОС і використовуваних додатків, як правило, вимагають все більше ресурсів.

Найпростіший спосіб оновити свій ПК – придбати новий системний блок із сучасною «начинкою», що при наявності грошей не є проблемою. Однак це не завжди розумно – найчастіше для підвищення продуктивності досить вдатися до заміни окремих комплектуючих.

Щоб вибрати стратегію апгрейда, потрібно з'ясувати, яке апаратне забезпечення встановлено, і зрозуміти, чого не вистачає комп'ютера для швидшої роботи – потужності процесора, можливостей відеосистеми, обсягів пам'яті, швидкості читання/запису даних жорстким диском і т.п. Але це лише одна сторона медалі. Після придбання нового системного блоку або апгрейда старого потрібно оперативно з'ясувати, чи відповідає «начинка» системного блоку заявленої при покупці (без відкриття самого блоку, оскільки на ньому може стояти пломба), оцінити, на скільки підвищилася продуктивність, і зрозуміти, чи дійсно комп'ютер стабільно працює.

Будь-який професійний збирач зможе легко вирішити перераховані завдання, оскільки має в своєму арсеналі чимало різноманітних і вузькоспеціалізованих інформаціоннодіагностіческіх інструментів. Звичайному користувачеві обзаводитися подібними рішеннями ні до чого, однак встановити просту комплексну утиліту для отримання інформації про залізо і тестування комп'ютера все одно необхідно [1][1)].

Загальні характеристики кваліфікаційної роботи:

- повний обсяг сторінок пояснювальної записки – 56;
- кількість рисунків – 27;
- кількість таблиць – 2;
- кількість посилань – 17.

[1)] [1] Полезные утилиты для диагностики и тестирования железа. URL: https://compress.ru/article.aspx?id=22304. (дата звернення 18.03.2020).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Дуже часто як користувачі, так і продавці-початківці комп'ютерної техніки не володіють повною інформацією щодо сумісності різноманітного комплектуючого.

Якщо мова йде про покупку онлайн, то найбільш популярні сайти мають функцію «Підбор комплектуючих», але тільки у залежності від асортименту свого товару. Якщо необхідно скористатися кишеньковим довідником, то такий програмний додаток доступний не кожному.

Не кожна програма в змозі включати до своєї бази нові моделі процесорів, відео карт та іншого заліза (все залежить від повноти бази і регулярності її поновлення), та й обсяг інформації щодо виявлених комплектуючих може бути різним – від мінімального до вичерпного [2][1)].

Тому актуальність створення програмного додатку у такій області є актуальним.

1.3 Огляд аналогів

Для чіткого формування вимог до об'єкту розробки на першому етапі проектування слід розглянути декілька аналогів. Це допоможе надалі чітко сформувати вимоги до дипломного проекту. Першим аналогом розглянемо пошуковий функціонал «NerdPart», який являє собою конфігуратор комп'ютера (рис. 1). Він розташован на платформі онлайн- вітрини «telemart.ua» [3][2)].

[1)] [2] Полезные утилиты для диагностики и тестирования железа. URL: https://compress.ru/article.aspx?id=22304. (дата звернення 15.03.2020).

[2)] [3] Офіційний сайт компанії telemart. URL: https://telemart.ua. (дата звернення 15.03.2020).

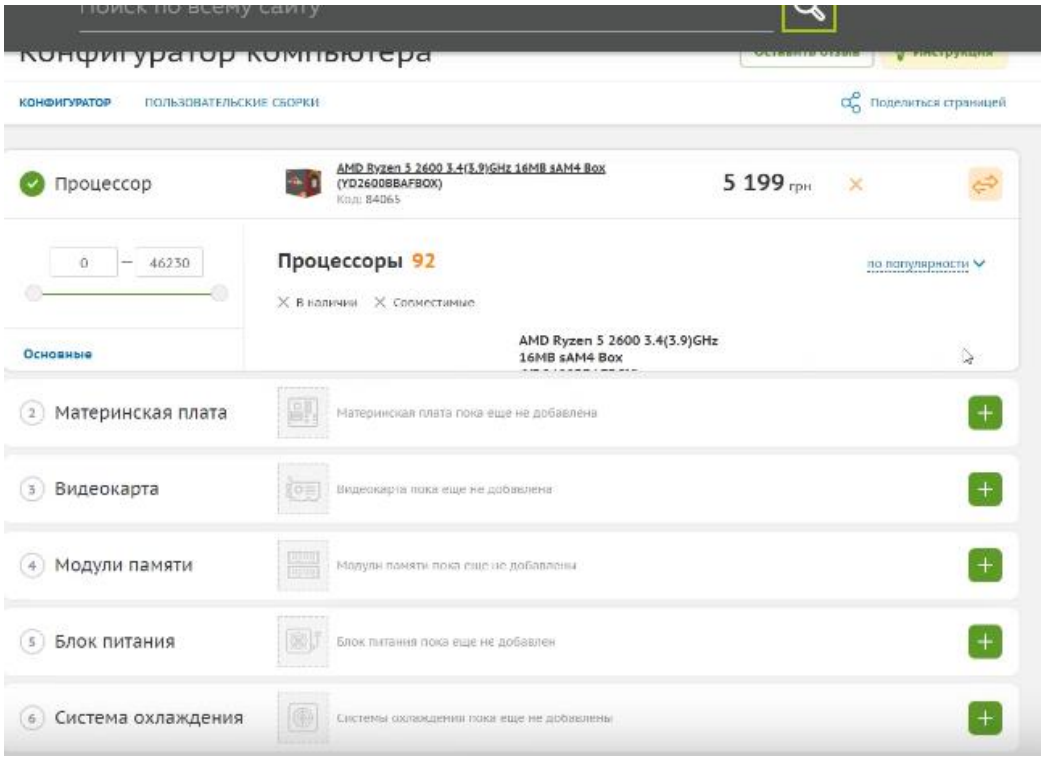

Рисунок 1 – Скріншот робочого вікнконфігуратора «NerdPart»

Даний конфігуратор предназначений для спрощення онлайн підбору комплектуючих для майбутного компютера, який покупець прибдає на «Телемарті». Модуль дозволяє підібрати повністю сумісні компоненти, а також інформує про помилки або проблемних місцях вашої майбутньої збірки, які ви можете усунути в режимі онлайн.

Конфігуратор збірки носить виключно рекомендаційний характер і не обмежує покупця в процесі складання вашого майбутнього ПК.

Модуль призначений для:

- збірки повноцінної і сумісної конфігурації ПК;
- підбору комплектуючих для апгрейда вже існуючої системи.

Модуль дозволяє почати вибір компонента з будь-якої категорії комплектуючих, наприклад, це може бути процесор, ОЗУ або корпус [4][1)].

[1)] [4] Конфигуратор компьютера NerdPart. URL: https://telemart.ua/assembly-start.html. (дата звернення 15.03.2020).

У процесі підбору модуль спочатку пропонує тільки сумісні комплектуючі, які доступні до покупки. У разі якщо необхідно знайти альтернативний товар, який недоступний, то варто прибрати обмеження в рамках основних фільтрів.

У процесі підбору можливо:

- додавати, замінювати, видаляти будь-які компоненти для вашого майбутнього ПК;
- змінювати критерії підбору комплектуючих за допомогою фільтрів в кожній категорії;
- порівнювати товари в рамках категорії;
- вивчати рекомендації системи і виправляти помилки в збірці.

Конфігуратор в реальному режимі інформує про помилки та проблеми в збірці. Інформація про помилки вказує на:

- несумісність компонентів між двома або більше комплектуючими;
- недолік потужності блоку живлення для системи;
- недолік слотів розширення для комплектуючих;
- інші типи помилок в процесі складання.

Наступним аналогом розглянемо сайт компанії «InTeam»[3][1)] .

На даному сайті для користувача представлено функцію візуального підбору комплектуючого (за назвою та графічною іконкою). Обравши конкретній компонент, відкривається закладка для вибору товару з урахуванням його технічних характеристик (рис. 3).

Такий пошук є дуже зручним для користувача, особливо фільтр «Ціновий діапозон». Таким чином, покупець має можливість придбати собі товар як за вимогами характеристик, так і з урахуванням бюджету. Що є дуже актуальним на сьогодні.

[1)] [3] Офіційний сайт компанії InTeam. URL: https://inteam.com.ua/sborka-pc/. (дата звернення 15.03.2020).

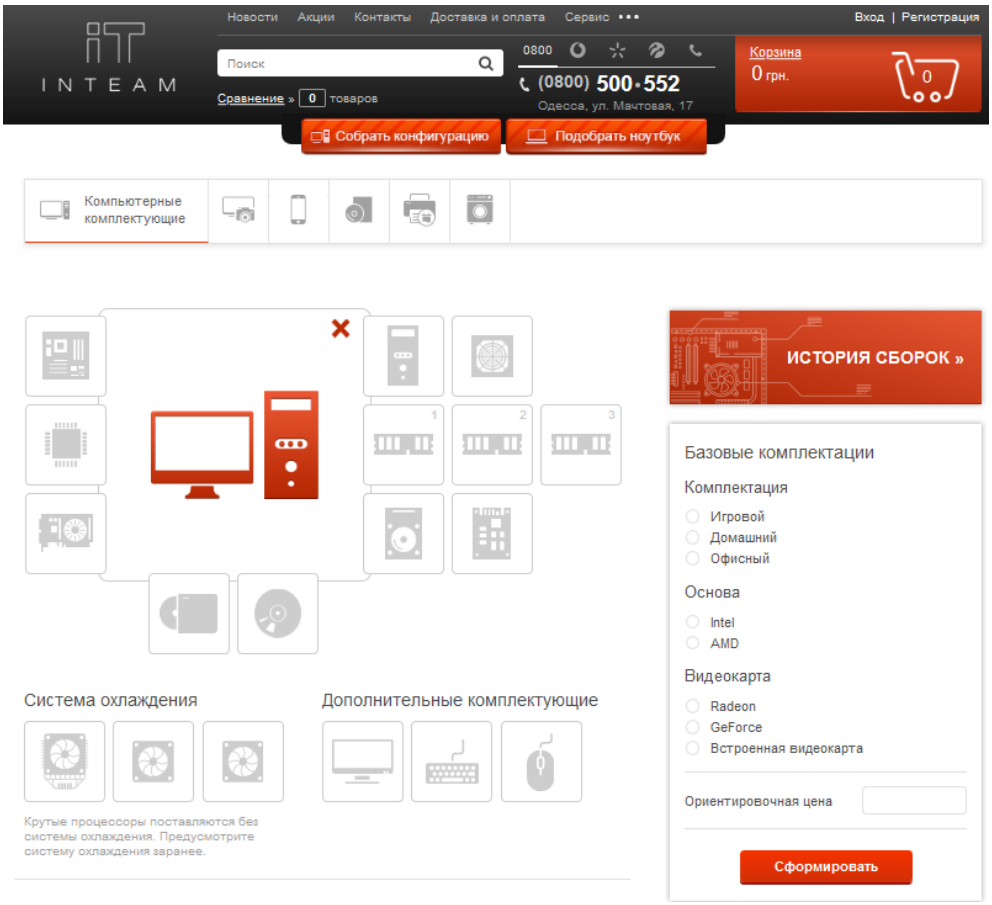

Рисунок 2 – Скріншот головної сторінки сайту компанії «InTeam»

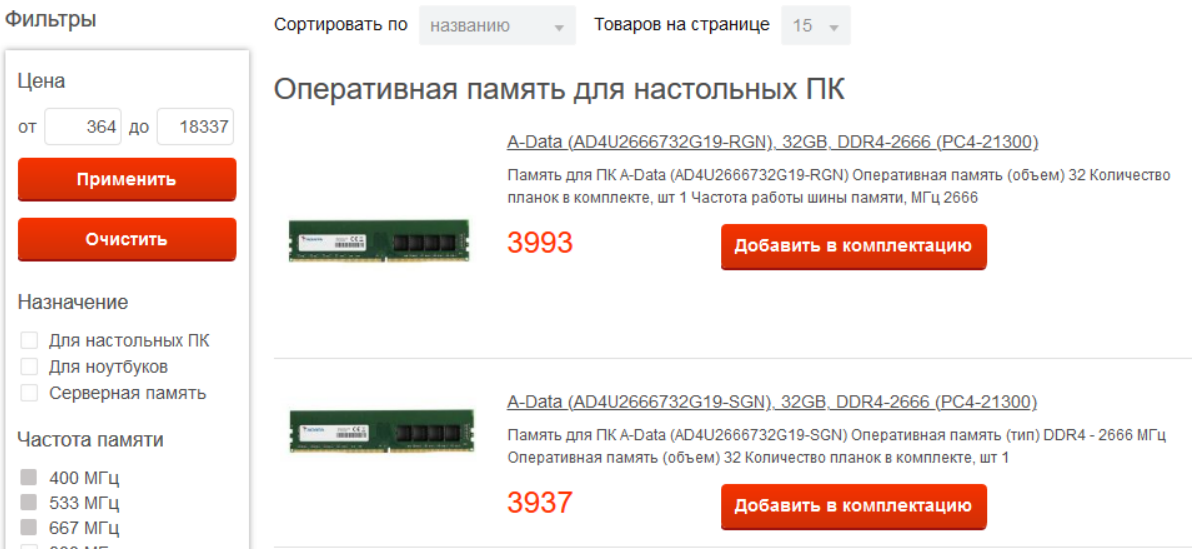

Рисунок 3 – Скріншот сторінки з вибору по критеріям товару на сайті компанії «InTeam»

Останнім аналогом розглянемо роботу функції пошуку комплектуючого на сайті компанії «dns-shop» [5][1)] (рис. 4):

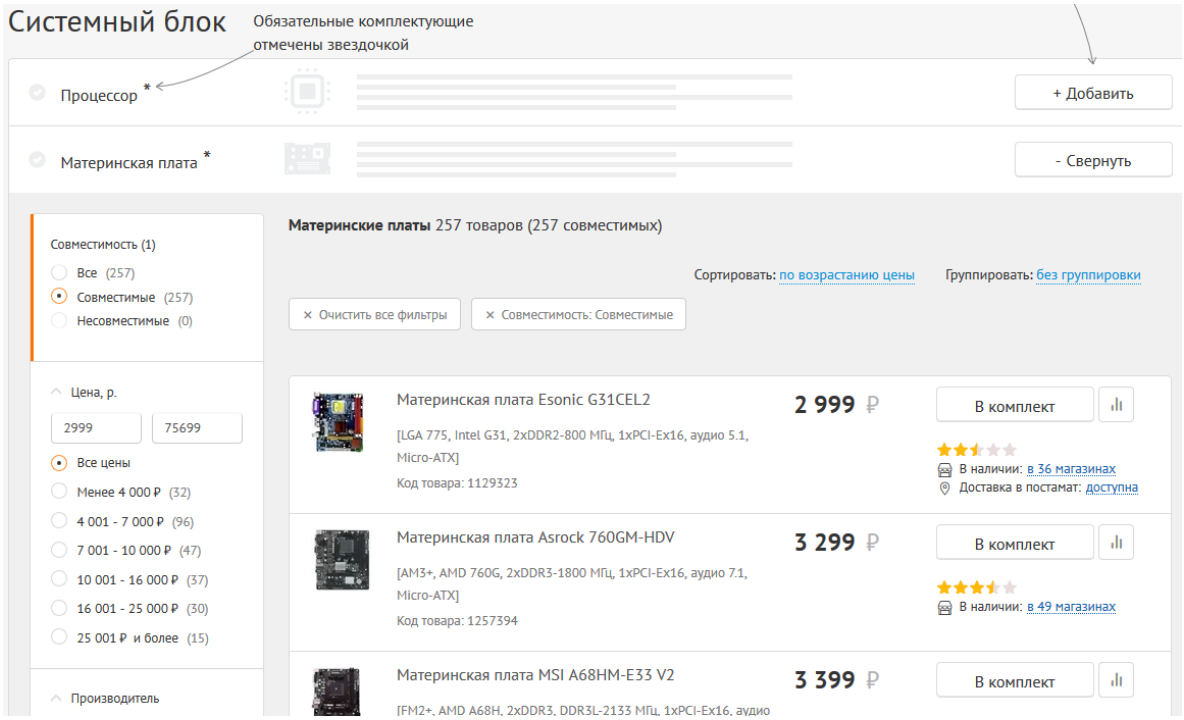

Рисунок 4 – Скріншот конфігуратора комп’ютера на сайті компанії «dns-shop»

Це хороший інтернет ресурс для підборки комплектуючих вашого майбутнього ПК з описом, характеристиками, актуальними цінами й іншою корисною інформацією.

Тут реалізовано зручний фільтр сумісності комплектуючих, який не дозволить вам купити процесор від компанії Intel і материнську карту сумісну тільки з процесорами AMD.

Інструмент відображає причини несумісності і рекомендації по заміні пристроїв, а також видаляє зі списку несумісні деталі.

[1)] [5] Конфигуратор компьютера «dns-shop». URL: https://www.dns-shop.ru/configurator/. (дата звернення 16.03.2020).

Для початку слід обрати материнську карту, процесор і будь-яку іншу деталь комп'ютера, і сайт автоматично запропонує вам залишилися комплектуючі сумісні з вашим початковим вибором. Вартість і термін отримання замовлення оновлюється після кожної доданої деталі.

Користувач зможе скопіювати посилання на список товарів, доданих в конфігуратор і поділитися нею з друзями. Є можливість збереження конфігурації в особистому кабінеті. Ви можете скористатися конфігуратором для отримання списку необхідних деталей, після чого придбати їх на іншому сайті.

Аналіз аналогів показує, що всі вони вбудовані у веб-сторінки інтертен-магазинів. По факту, кишенькового програмного продукту з підбору комп’ютерного комплектуючого немає.

1.4 Вибір засобів розробки

1.4.1 MS SQL Server 2017

В останні роки генерується і обробляється все більше даних, збільшується різноманітність їх форми і змісту. Частина даних як і раніше має реляційний формат і генерується традиційними транзакційними інструментами. Зазвичай такі дані структуровані, їх зміст і цінність добре зрозумілі і відомі.

Але величезна кількість даних має набагато більше сирої вид. Це дані з датчиків і сенсорів (той самий Інтернет речей), відеокамер, записуючих пристроїв. Ці дані, швидше за все, несуть цінність, але витягти її поки набагато складніше [6][1)]. Роль сучасної платформи даних – прийняти такі різнорідні дані, інтегрувати їх, обробити і витягти цінну для бізнесу інформацію. Крім того, така платформа повинна:

- обробляти дані як в уже існуючих локальних інфраструктурах, так і в хмарах. це гібридне стан триватиме ще невизначено довгий час;

[1)] [6] 10 причин перейти на Microsoft SQL Server 2019. URL: https://softline.ru/about/blog/10-prichin-pereyti-na-microsoft-sql-server-2019. (дата звернення 20.03.2020).

- дозволяти переносити існуючі інструменти обробки даних в хмарну платформу без серйозних змін;
- дозволяти розробляти сучасні хмарні додатки з нуля, задіюючи всі хмарні інструменти;
- аналізувати дані однаково успішно як в локальному середовищі, так і в хмарної платформі [6][1)].

SQL Server є однією з найбільш популярних систем управління базами даних (СКБД) в світі. Дана СУБД підходить для самих різних проектів: від невеликих додатків до великих високонавантажених проектів.

SQL Server характеризується такими особливостями як:

- продуктивність. SQL Server працює дуже швидко;
- надійність і безпека. SQL Server надає шифрування даних;
- простота. З даної СУБД відносно легко працювати і вести адміністрування.

У SQL Server 2017 з'явилося нове покоління функцій обробки запитів, що дозволяють поліпшити продуктивність запитів в SQL Server шляхом адаптації до середовища виконання робочих навантажень додатків.

Дана можливість включає наступні функції:

- зворотній зв'язок по тимчасово надається буферу пам'яті в пакетному режимі;
- адаптивні з'єднання в пакетному режимі;
- виконання з чергуванням для функцій з табличним значенням з декількома інструкціями [7][2)].

[1)] [6] 10 причин перейти на Microsoft SQL Server 2019. URL: https://soft-line.ru/about/blog/10-prichin-pereyti-na-microsoft-sql-server-2019. (дата звернення 20.03.2020).

[2)] [7] Обзор основных нововведений в Microsoft SQL Server 2017. URL: https://info-comp.ru/novosti/594-review-microsoft-sql-server-2017.html. (дата звернення 20.03.2020).

1.4.2 Платформа ASP.NET Core

Платформа ASP.NET Core представляє технологію від компанії Microsoft, призначену для створення різного роду веб-додатків: від невеликих веб-сайтів до великих веб-порталів і веб-сервісів. ASP.NET Core тепер повністю є opensource-фреймворком.

ASP.NET Core може працювати поверх крос-платформної середовища .NET Core, яка може бути розгорнута на основних популярних операційних системах: Windows, Mac OS, Linux. І таким чином, за допомогою ASP.NET Core ми можемо створювати крос-платформні додатки. І хоча Windows як середовище для розробки і розгортання програми досі превалює, але тепер вже ми не обмежені тільки цією операційною системою.

Тобто можна запускати веб-додатки не тільки на ОС Windows, але і на Linux і Mac OS. Завдяки модульності фреймворка всі необхідні компоненти веб-додатки можуть завантажуватися як окремі модулі через пакетний менеджер Nuget. Крім того, на відміну від попередніх версій платформи немає необхідності використовувати бібліотеку System.Web.dll.

ASP.NET Core включає в себе фреймворк MVC, який об'єднує функціональність MVC, Web API і Web Pages. У попередніх версії платформи дані технології реалізувалися окремо і тому містили багато дублюючої функціональності. Зараз же вони об'єднані в одну програмну модель ASP.NET Core MVC. А Web Forms повністю пішли в минуле [8][1)].

Крім об'єднання вищезазначених технологій в одну модель в MVC був доданий ряд додаткових функцій. Однією з таких функцій є тег-хелпери (tag helper), які дозволяють більш органічно поєднувати синтаксис html з кодом C #. ASP.NET Core характеризується розширюваністю. Фреймворк побудований з набору щодо незалежних компонентів. І ми можемо або використовувати

[1)] [8] Введение в ASP.NET Core. ASP.NET Core – новая эпоха в развитии ASP.NET. URL: https://metanit.com/sharp/aspnet5/1.1.php. (дата звернення 22.03.2020).

вбудовану реалізацію цих компонентів, або розширити їх за допомогою механізму спадкування, або зовсім створити і застосовувати свої компоненти зі своїм функціоналом.

Також було спрощено управління залежностями і конфігурація проекту. Фреймворк тепер має свій легкий контейнер для впровадження залежностей, і більше немає необхідності застосовувати сторонні контейнери. Хоча при бажанні їх також можна продовжувати використовувати [8][1)].

В якості інструментарію розробки ми можемо використовувати останні випуски Visual Studio, починаючи з версії Visual Studio 2015.

1.4.3 Entity Framework Core

Entity Framework Core (EF Core) являє собою об'єктно-орієнтовану, легковажну і расширяемую технологію від компанії Microsoft для доступу до даних. EF Core є ORM-інструментом (object-relational mapping – відображення даних на реальні об'єкти).

Тобто EF Core дозволяє працювати базами даних, але є більш високий рівень абстракції: EF Core дозволяє абстрагуватися від самої бази даних і її таблиць і працювати з даними незалежно від типу сховища. Якщо на фізичному рівні ми оперуємо таблицями, індексами, первинними і зовнішніми ключами, але на концептуальному рівні, який нам пропонує Entity Framework, ми вже працюємо з об'єктами.

Entity Framework Core підтримує безліч різних систем баз даних. Таким чином, ми можемо через EF Core працювати з будь-якої СУБД, якщо для неї є потрібний провайдер [9][2)].

[1)] [8] Введение в ASP.NET Core. ASP.NET Core – новая эпоха в развитии ASP.NET. URL: https://metanit.com/sharp/aspnet5/1.1.php. (дата звернення 22.03.2020).

[2)] [9] Введение в Entity Framework Core. Что такое Entity Framework Core. URL: https://metanit.com/sharp/entityframeworkcore/1.1.php. (дата звернення 22.03.2020).

Як технологія доступу до даних Entity Framework Core може використовуватися на різних платформах стека .NET. Це і стандартні платформи типу Windows Forms, консольні додатки, WPF, UWP і ASP.NET Core. При цьому кроссплатформенная природа EF Core дозволяє задіяти її не тільки на ОС Windows, але і на Linux і Mac OS X.

Центральної концепцією Entity Framework є поняття сутності або entity. Сутність визначає набір даних, які пов'язані з певним об'єктом. Тому дана технологія передбачає роботу не з таблицями, а з об'єктами і їх колекціями.

Будь-яка сутність, як і будь-який об'єкт з реального світу, має низку властивостей. Наприклад, якщо сутність описує людини, то ми можемо виділити такі властивості, як ім'я, прізвище, зріст, вік.

Властивості необов'язково представляють прості дані типу int або string, але можуть також представляти і більш комплексні типи даних. І у кожної сутності може бути одна або кілька властивостей, які будуть відрізняти цю сутність від інших і будуть унікально визначати цю сутність. Подібні властивості називають ключами [9][1)].

При цьому суті можуть бути пов'язані асоціативної зв'язком один-ко-многим, один-ко-одному і багато-до-багатьох, подібно до того, як в реальній базі даних відбувається зв'язок через зовнішні ключі.

Відмінною рисою Entity Framework Core, як технології ORM, є використання запитів LINQ для вибірки даних з БД. За допомогою LINQ ми можемо створювати різні запити на вибірку об'єктів, в тому числі пов'язаних різними асоціативними зв'язками. А Entity Framework при виконання запиту транслює вираження LINQ в вирази, зрозумілі для конкретної СУБД (як правило, в вирази SQL) [10][2)].

[1)] [9] Введение в Entity Framework Core. Что такое Entity Framework Core. URL: https://metanit.com/sharp/entityframeworkcore/1.1.php. (дата звернення 22.03.2020).

[2)] [10] Начинаем работу с ASP.NET Core и Entity Framework 6. URL: Введение в Entity Framework Core. (дата звернення 25.03.2020).

1.4.4 Android-додаток Xamarin.Forms

Xamarin.Forms представляє платформу, яка націлена на створення кросплатформенних додатків під Android, iOS і Windows 10. У звязку з тим, що значна частина мобільних додатків створюється більш ніж для однієї платформи, наприклад, для Android і iOS, розробники стикаються з наступними труднощами:

- відмінність в підходах побудова графічного інтерфейсу так чи інакше впливає на розробку. Розробники змушені підлаштовувати додаток під вимоги до інтерфейсу на конкретній платформі;
- різні API – розбіжність у програмних інтерфейсів і реалізації тих чи інших функциональностей також вимагає від програміста облік цих специфічних особливостей;
- різні платформи для розробки.

Для створення додатків під Windows використовується Visual Studio. Дуже ефективно мати один інструмент, який дозволяв легко і просто створювати додатки відразу для всіх платформ. І саме таким інструментом і є платформа Xamarin, який дозволяє створювати одну единственную логіку додатка із застосуванням C# і .NET відразу для всіх трьох платформ – Android, iOS, UWP [11][1)].

Переваги використання Xamarin.Forms:

- в процесі розробки створюється єдиний код для всіх платформ;
- xamarin надає прямий доступ до нативним api кожної платформи;
- при створенні додатків ми можемо використовувати платформу .net і мова програмування c# (а також f#), який є досить продуктивним, і в той же час ясним і простим для освоєння і застосування;
- xamarin forms підтримує кілька платформ.

[1)] [11] Разработка приложений для Android с C#. URL: https://habr.com/ru/post/169467/. (дата звернення 27.03.2020).

Xamarin працює поверх фреймворка Mono, який надає opensource-реалізацію .NET Framework. За допомогою Xamarin.Android код C# з використанням Xamarin компілюється в Intermediate Language (IL), який потім при запуску програми компілюється в нативну збірку. Xamarin-додатки запускаються в середовищі виконання Mono. Безпосередньо код не може звертатися до API Android.

Спеціальна прошарок Managed Callable Wrappers (MCW) дозволяє транслювати виклику managed-коду в нативні виклики і звертатися до функціональності просторів імен Android. * І Java. *

І навпаки, коли Android Runtime (ART) звертається до додатка з кодом Xamarin, то всі виклики проходять через обгортку Android Callable Wrappers (ACW) [11][1)].

[1)] [11] Разработка приложений для Android с C#. URL: https://habr.com/ru/post/169467/. (дата звернення 27.03.2020).

2 ПРОЕКТУВАННЯ СИСТЕМИ UML-ЗАСОБАМИ

Інженерія програмного забезпечення для досягнення позитивних результатів вимагає застосування спеціалізованих методик управління, як самим процесом, так і всіма стадіями аналізу, проектування, реалізації та подальшого використання розробленого програмного продукту. В якості основи даного процесу розглянуто уніфікована мова моделювання UML (Unified Modeling Language) і способи його застосування для управління процесами розробки додатків в цілому [12][1)].

2.1 Діаграма варіантів використання

Діаграми варіантів використання описують взаємини і залежності між групами варіантів використання і дійових осіб, які беруть участь в процесі. Діаграми Use-Case використання не призначені для відображення проекту і не можуть описувати внутрішній устрій системи.

Діаграми варіантів використання призначені для спрощення взаємодії з майбутніми користувачами системи, з клієнтами, і особливо знадобляться для визначення необхідних характеристик системи. Іншими словами, діаграми варіантів використання говорять про те, що система повинна робити, не вказуючи самі застосовувані методи [13][2)].

Діаграма Use-Case є найбільш загальним поданням функціональних вимог до системи. Для подальшого проектування системи потрібні більш конкретні деталі, які описуються в документі, званому сценарієм варіанти використання або потоком подій (flow of events).

[1)] [12] Проектирование программного обеспечения с использованием стандартов uml 2. 0 и SysML 1. 0. URL: https://cyberleninka.ru/article/n/proektirovanie-programmnogo-obespecheniya-s-ispolzovaniem-standartov-uml-2-0-i-sysml-1-0. (дата звернення 27.03.2020).

[2)] [13] UML – диаграмма вариантов использования (use case diagram). URL: https://habr.com/ru/post/47940/. (дата звернення 27.03.2020).

Сценарій детально документує процес взаємодії дійової особи з системою, що реалізується в рамках варіанту використання. Основний потік подій описує нормальний хід подій (при відсутності помилок). Альтернативні потоки описують відхилення від нормального перебігу подій (помилкові ситуації) і їх обробку.

Переваги моделі варіантів використання полягають в тому, що вона:

- визначає користувачів і кордони системи;
- визначає системний інтерфейс;
- зручна для спілкування користувачів з розробниками;
- використовується для написання тестів;
- є основою для написання документації користувача;
- добре вписується в будь-які методи проектування (як об'єктно-орієнтовані, так і структурні) [14][1)].

На діаграмах варіантів використання зображуються актори і варіанти використання, між якими існують відносини. Тут можна показувати і інші елементи UML.

Актором називають зовнішню по відношенню до ПС сутність, яка може взаємодіяти з системою. Акторами можуть бути як люди, так і зовнішні системи або пристрою. Слід завжди пам'ятати, що актор - це не конкретна людина або пристрій, а роль (посадовий обов'язок), в якій він виступає по відношенню до програмної системи.

Наприклад, в якості актора «Бухгалтер» може виступати весь готівковий штат бухгалтерії. У той же час одна конкретна людина може грати кілька ролей по відношенню до системи.

При взаємодії актора з системою остання виконує ряд робіт, які утворюють варіант використання системи (use case). Кожен актор може використову-

---

[1)] [14] Построение диаграммы вариантов использования. URL: https://flexberry.github.io/ru/gpg_use-case-diagram.html. (дата звернення 27.03.2020).

вати систему по різному, тобто ініціювати виконання різних варыантыв використання. Він не представляє собою конструкцію, безпосередньо реалізовується в програмному коді. Всі його поведінка реалізується у вигляді класів і компонент.

На рисунку 5 представлено діаграму варіантів використання для об’єкту розробки:

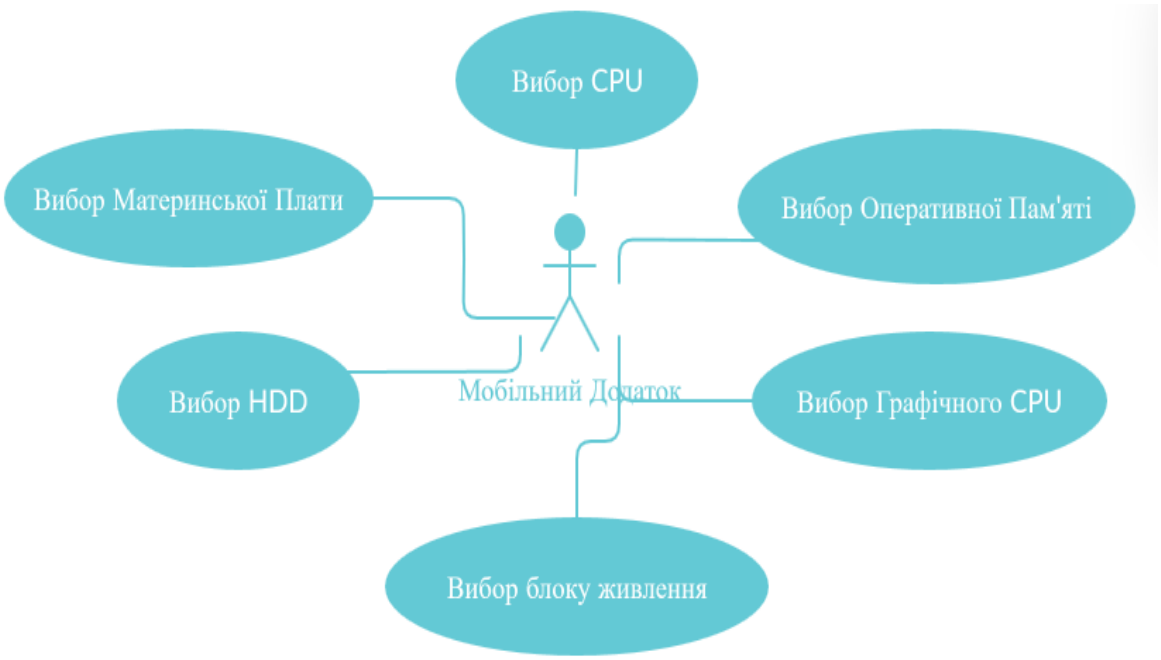

Рисунок 5 – Діаграма Use-Case для мобільного додатку

На наступному етапі проектування необхідно побудувати декомпозицію, тобто деталізацію, основної діаграми використання. Кожен етап формування зборки комплектуючого складається з вибору декількох характеристик. Наприклад, при обранні процесору необхідно обрати год випуску, кількість ядер, кількість потоків роботи, фірму, тип оперативної памяті і т.п.

Всі обрані критерії допомогають користувачу мобільного додатку максимально точно підібрати необхідне комплектуюче до свого комп’ютера, особливо, якщо до функціоналу додатку входить пошукова функція за критеріями.

Так, для актора Користувач вищезазначені варіанти використання можна деталізувати наступним чином (рис. 6):

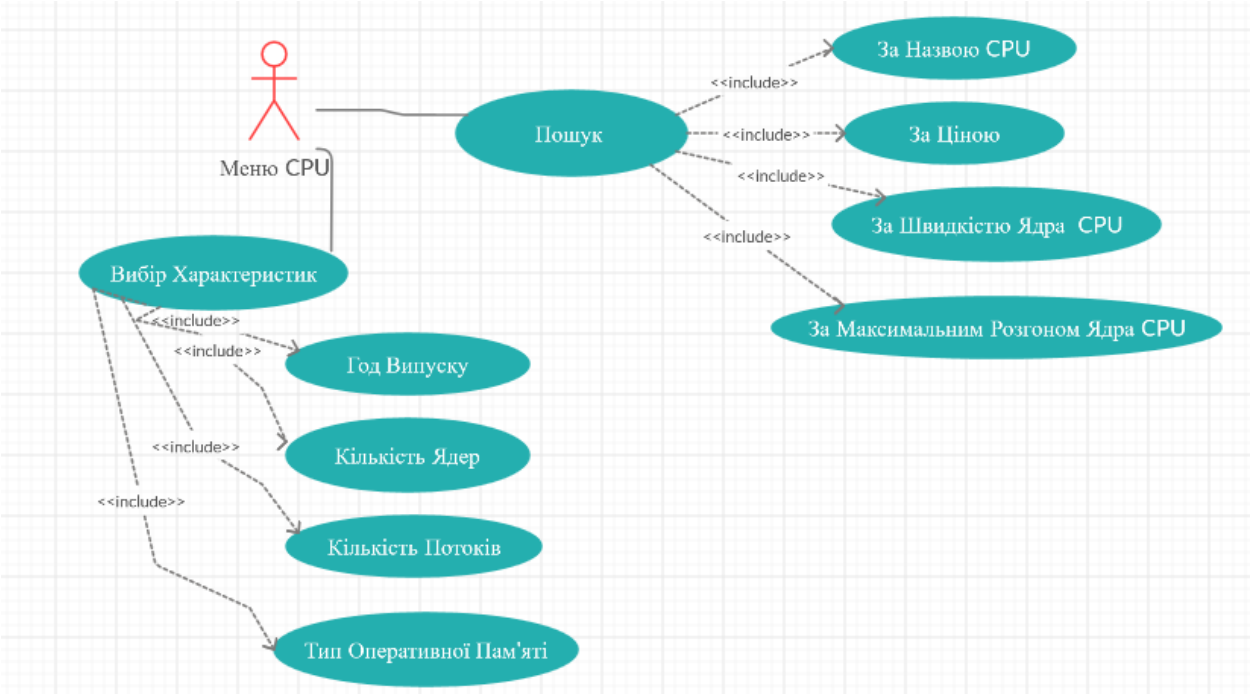

Рисунок 6 – Діаграма Use-Case для вибору CPU за характеристиками

2.2 Діаграма діяльності UML

Створення Інформаційної Системи – складний процес, який можна уявити як поетапний спуск від загальної концепції майбутньої ІС, через розуміння її логічної структури до найбільш детальним моделям, що описує фізичну реалізацію. Діаграма діяльності належить до логічної моделі.

В якості графічного уявлення для виділення основних функцій Системи ми застосовуємо діаграму варіантів використання (use case). Діаграма варіантів використання дає нам уявлення що повинна робити Система. На питання ЯК ми можемо відповісти, використовуючи діаграму активності.

Тобто якщо варіанти використання ставлять перед Системою мета, то діаграма діяльності показує послідовність дій, необхідних для її досягнення.

Дії (action) це елементарні кроки, які не передбачають подальшу декомпозицію. Діяльність може містити входять і/або вихідні дуги діяльності, що показують потоки управління і потоки даних. Якщо потік з'єднує дві діяльності, він є потоком управління. Якщо потік закінчується об'єктом, він є потоком даних [15][1)].

Діяльність виконується, тільки тоді, коли готові всі його «входи», після виконання, діяльність передає управління і (або) дані на свої «виходи». Саму діаграму діяльності прийнято розташовувати таким чином, щоб дії слідували зліва направо або зверху вниз.

Приклад діаграми діяльності для роботи мобільного додатку представлено на рисунку 7:

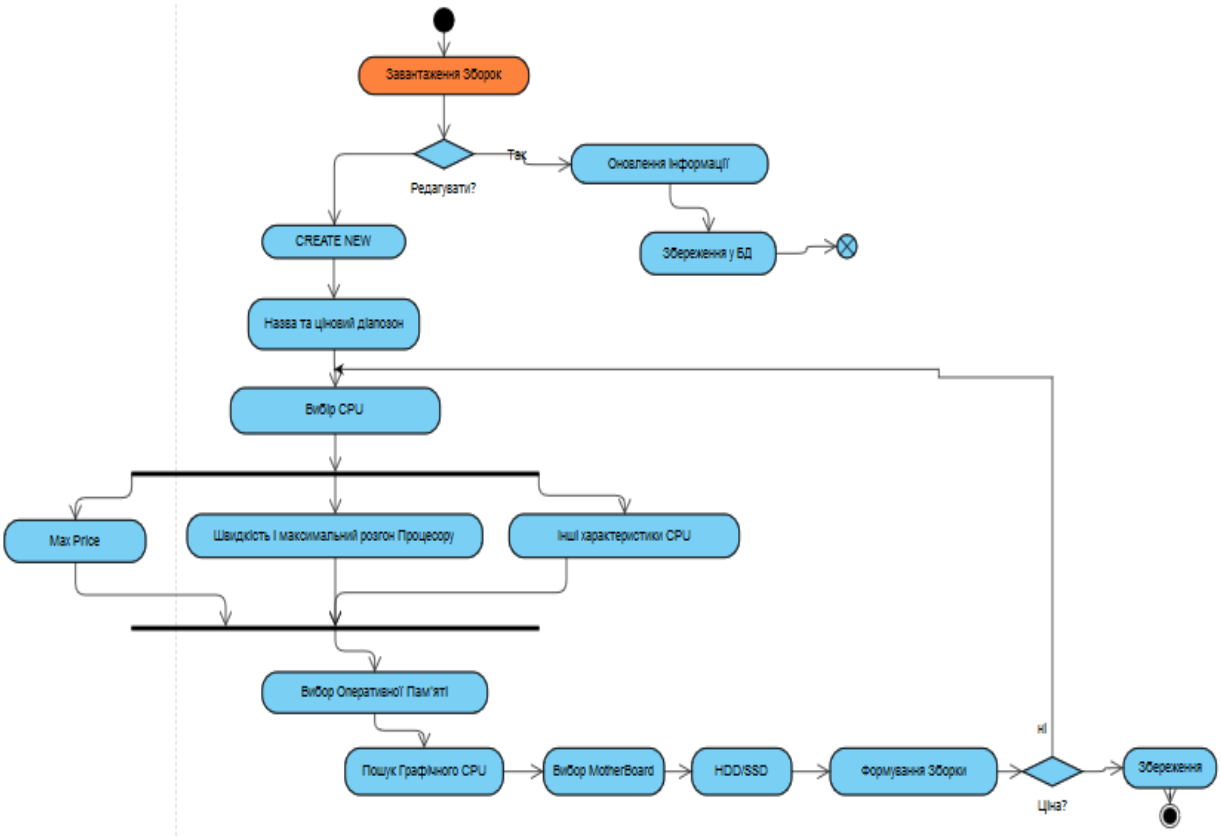

Рисунок 7 – Діаграма діяльності «Робота додатку»

[1)] [15] Теория и практика UML. Диаграмма деятельности. URL: https://it-gost.ru/articles/view_articles/96. (дата звернення 28.03.2020).

3 ОПИС РЕАЛІЗАЦІЇ ТА ІНСТРУКЦІЯ КОРИСТУВАННЯМ ДОДАТКОМ

3.1 Інтерфейсна частина програмного продукту

Завдяки правильно поставленим задачам на початку проектування додатку, було створено програмний продукт, що повністю відповідає технічному завданню.

На головному екрані додатку ми можемо побачити раніше створені списки компонентів зборки комп'ютера (рис. 8).

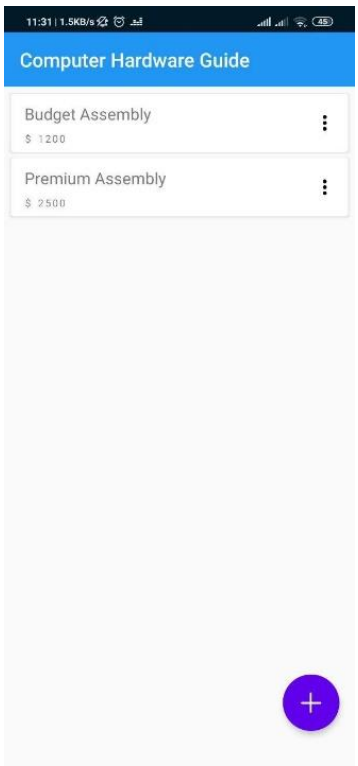

Рисунок 8 – Головний екран мобільного додатку

Кожен елемент списку має власну назву, максимальну ціну зборки, зазначену в доларах та випадаюче меню. Воно, в свою чергу, складається з двох елементів «View» та «Delete», що дозволяють відкрити та видалити зборку (рис. 9).

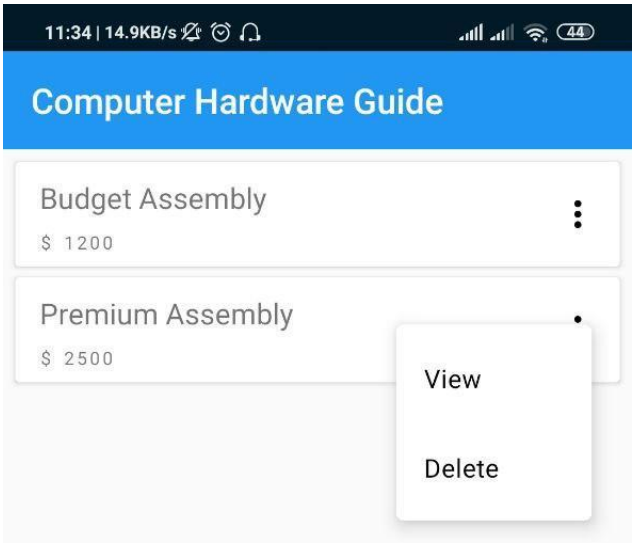

Рисунок 9 – Функції, які можна обрати для зборки

Окрім того, в нижній частині головного екрану розташована кнопка «+», за допомогою якої з’являється можливість додавання нового елементу списку – нову зборку (рис. 10). Після натискання кнопки, відкривається модальне вікно, що називається «New Assembly».

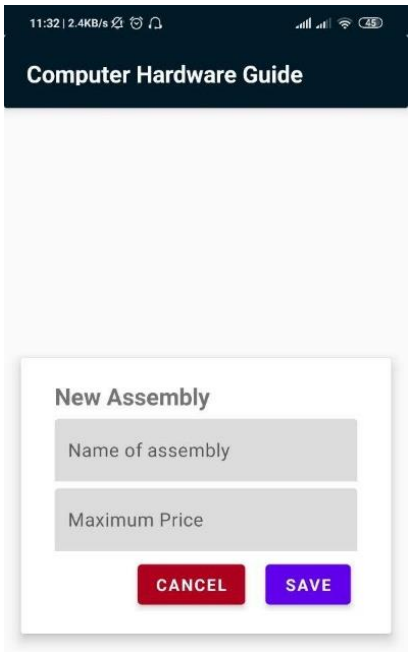

Рисунок 10 – Додавання нової зборки

В кожен з рядків необхідно вписати назву та максимальну ціну зборки. Після чого натиснути «Cancel», або «Save». Якщо поля не заповнені і користувач натиснув «Save», з’явиться помилка (рис. 11).

В даному випадку використовується валідація кожного поля – це особливі вимоги, призначені для даних полів, що повинні бути дотримані (заповнення всіх полів). Якщо всі поля були заповнені, нова зборка буде додана успішно.

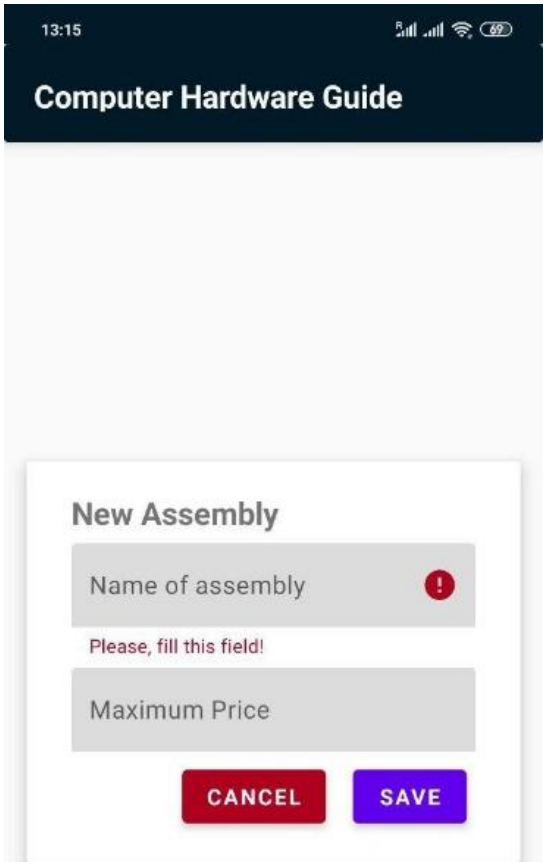

Рисунок 11 – Валідація полів

Також є можливість оновити зборку – «Update assembly». Це відбувається при натисканні на елемент списку в два етапи:

– зміна назви зборки «Change name» (рис. 12);
– зміна максимальної ціни «Change price» (рис. 12).

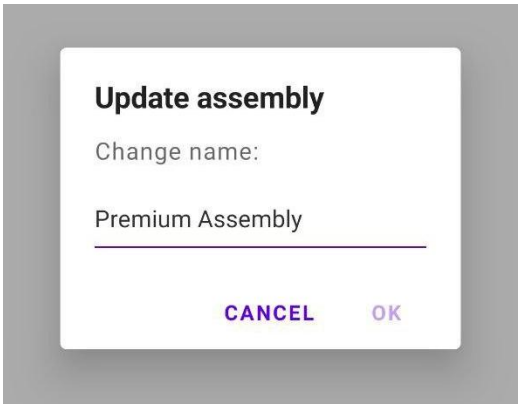

Рисунок 12 – Оновлення назви

При зміні безпосередньо ціни, можливо вводити тільки цифри.

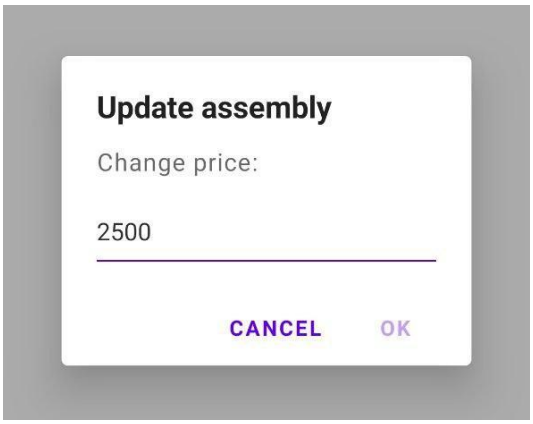

Рисунок 13 – Оновлення максимальної ціни

Якщо інформація була успішно змінена, після закінчення завантаження (рис. 14) оновлення будуть видимі на головному екрані.

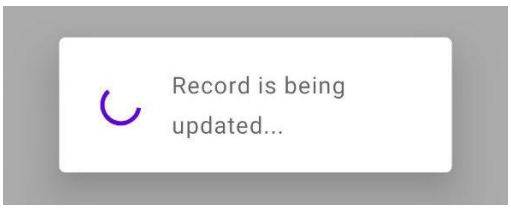

Рисунок 14 – Вікно завантаження

Отже, якщо ми хочемо продовжити створення нової зборки. Тут необхідно натиснути на елемент «View» у випадаючому списку по правій стороні на елементі списку.

Ось приклад як продовжити обирати компоненти для майбутнього комп’ютера на прикладі зборки «Budget Assembly».

Першим етапом є вибір процесора. На сторінці ми можемо побачити декілька фільтрів, за допомогою яких дуже просто обрати потрібний компонент комп’ютера по характеристикам пошуку.

Також є можливість пошуку по назві компонента, використовуючи поле «Search», що являє собою TextBox (рис. 15).

Рисунок 15 – Пошук по назві компонента

Фільтри, що знаходяться на сторінці є трьох типів, а саме Slider, RadioButton та CheckBox.

Фільтри типу Slider необхідні для вибору діапазонів певних характеристик, в даному випадку – це ціна, швидкість ядра та максимальний розгін ядра (рис. 16).

Наступним фільтром є RadioButton (рис. 17). За допомогою RadioButton обираємо поле «Yes», або «No», для вибору чи має процесор інтегрований GPU чи ні.

Далі розташована велика кількість фільтрів типу CheckBox, в першу чергу для зручного вибору характеристик процесора (рис. 18).

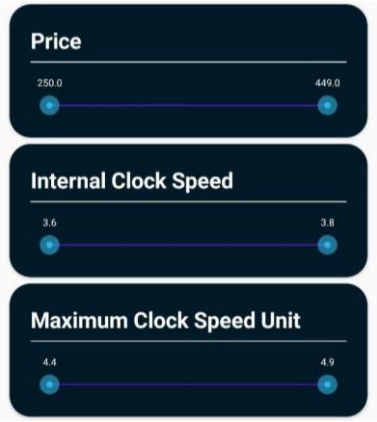

Рисунок 16 – Фільтри типу Slider

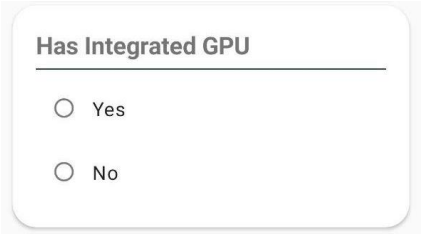

Рисунок 17 – Фільтр типу RadioButton

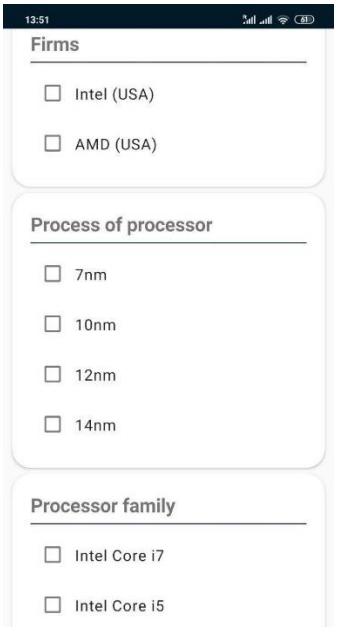

Рисунок 18 – Фільтри типу CheckBox

Найголовнішими фільтрами є:

– рок випуску;
– кількість ядер;
– кількість потоків процесора;
– фірма;
– тип оперативної пам’яті;
– сокет та інші.

Після вибору певних характеристик компонента натискаємо кнопку «Search CPU». Після завантаження, з'являються процесори, що відповідають обраним характеристикам.

Для того, щоб додати процесор, котрий відповідає нашим бажанням треба натиснути кнопку Add, що знаходиться в випадаючому списку з правої сторони (рис. 19).

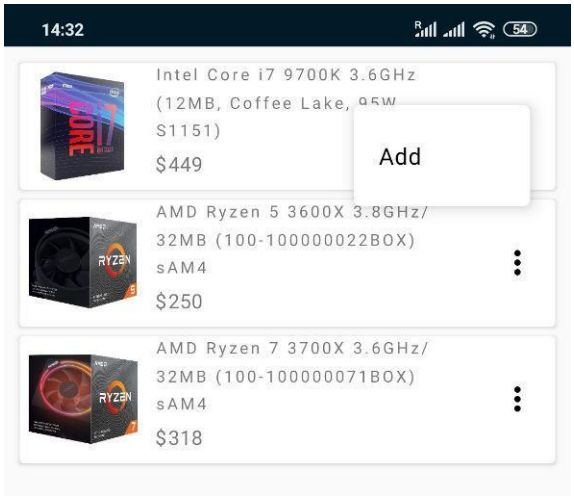

Рисунок 19 – Вибір необхідного компоненту

Наступним етапом є вибір оперативної пам’яті комп’ютера, тобто RAM (рис. 20). В даному випадку, існують такі запропоновані характеристики оперативної пам’яті:

– ціна;

– наявність радіаторів;
– об'єм оперативної пам'яті;
– частота;
– доступні фірми;
– тип оперативної пам’яті;
– таймінги.

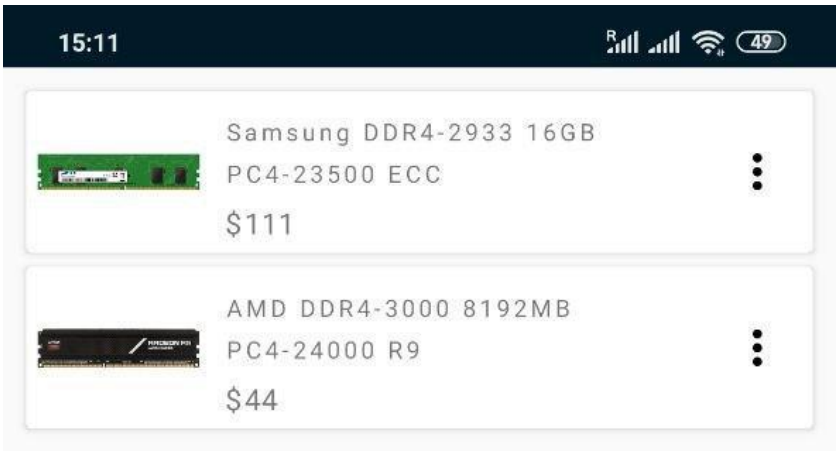

Рисунок 20 – Вибір оперативної пам’яті комп’ютера

Далі є можливість вибору графічного процесора – GPU (рис. 21). В даному випадку ми маємо до вибору наступні характеристики:

– ціна;
– доступні фірми;
– мінімальні потреби в електроенергії;
– ширина смуги пам'яті;
– об'єм GPU;
– інтерфейс GPU;
– тип GPU;
– джерело живлення, тощо.

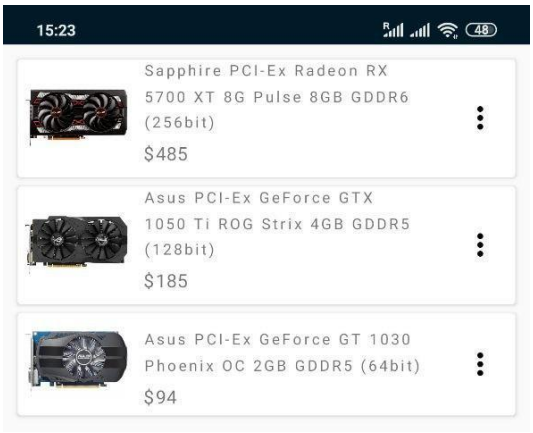

Рисунок 21 – Вибір графічного процесора

Наступним етапом є вибір блок живлення. Блоки живлення зазвичай мають такі характеристики:

– ціна;
– діаметр вентилятора;
– кількість SATA;
– джерело живлення;
– фірми.

Після вибору бажаних характеристик та натиску кнопки «Search Power Unit», з’являються можливі варіанти, котрі відповідають обраним характеристикам (рис. 22).

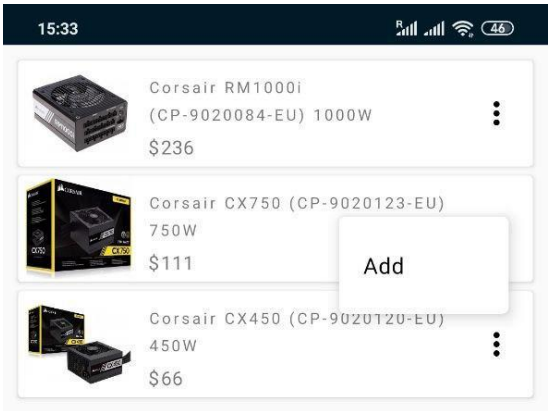

Рисунок 22 – Вибір блока живлення

Далі обираємо материнську плату за допомогою нижче представлених характеристик:

- ціна і кількість слотів оперативної пам’яті;
- інтерфейс GPU та інтерфейс ПЗУ;
- електропостачання;
- фірми;
- чіпсет та тип оперативної пам’яті;
- розмір форм-фактор і сокет.

Після кожного з етапів оновлюється стрічка процентного відношення, що розташована внизу сторінки. Вона показує, яка кількість етапів у відсотках була пройдена. Поряд з нею, з лівої сторони, відображається сума коштів всіх раніше обраних компонентів комп’ютера (рис. 23).

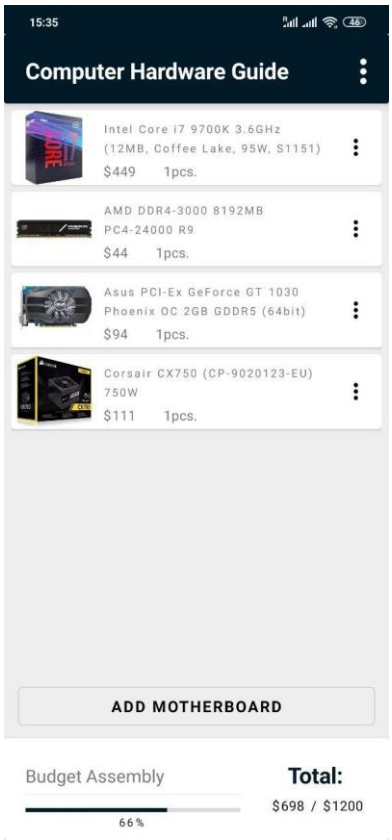

Рисунок 23 – Додані компоненти в списку, оновлений відсоток зборки та її вартість

Важливим є то, що попередні компоненти впливають на вибір подальших компонентів комп’ютера.

На наступному рисунку ми можемо побачити, що наступним етапом зборки є додавання HDD, або SSD (рис. 24).

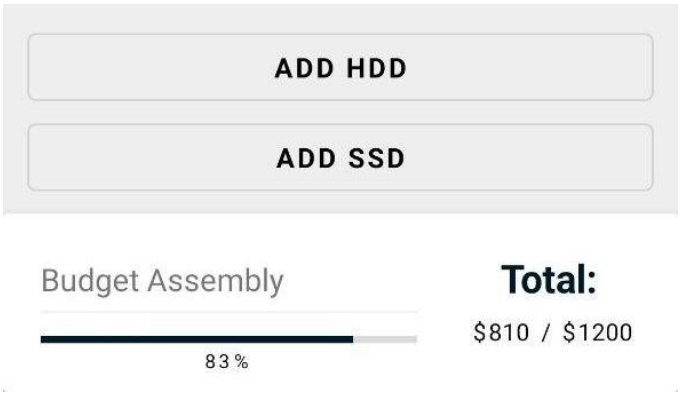

Рисунок 24 – Додавання HDD, або SSD

При виборі HDD, доступні наступні фільтри:

– обсяг;
– фірми;
– форм-фактор та інтерфейс ПЗУ;
– тип комірки пам'яті.

Компонент SSD обираємо за наступними характеристиками:

– ціна;
– обсяг;
– фірми.

Після натискання кнопки «Search», обираємо один з з’явившихся компонентів (рис. 25).

В кінцевому результаті, після проходження всіх етапів, утворюється повний список компонентів для даної зборки (рис. 26). Варто зауважити, що ми вклалися в зазначену раніше максимальну вартість зборки. Це можна побачити в нижній частині екрана (Total: $971 / $1200).

Рисунок 25 – Вибір компонента SSD

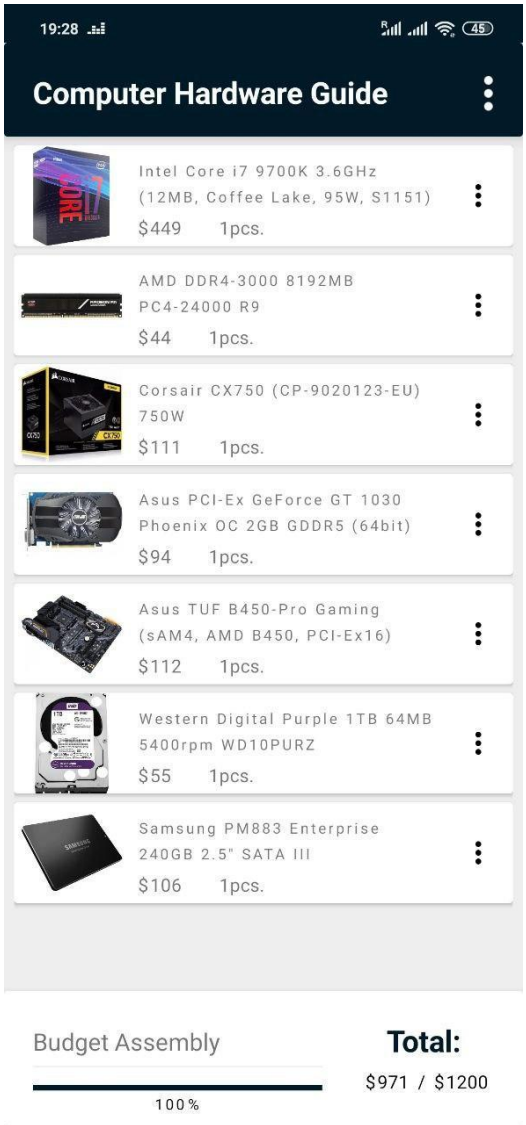

Рисунок 26 – Список повної зборки компонентів

3.2 Розробка бази даних

Для створення бази даних було використано підхід Code First, фреймворк – Entity Framework, який дозволяє створювати базу даних з коду C# не прибігаючи к SQL. Було автоматично згенеровано наступну базу даних (рис. 27).

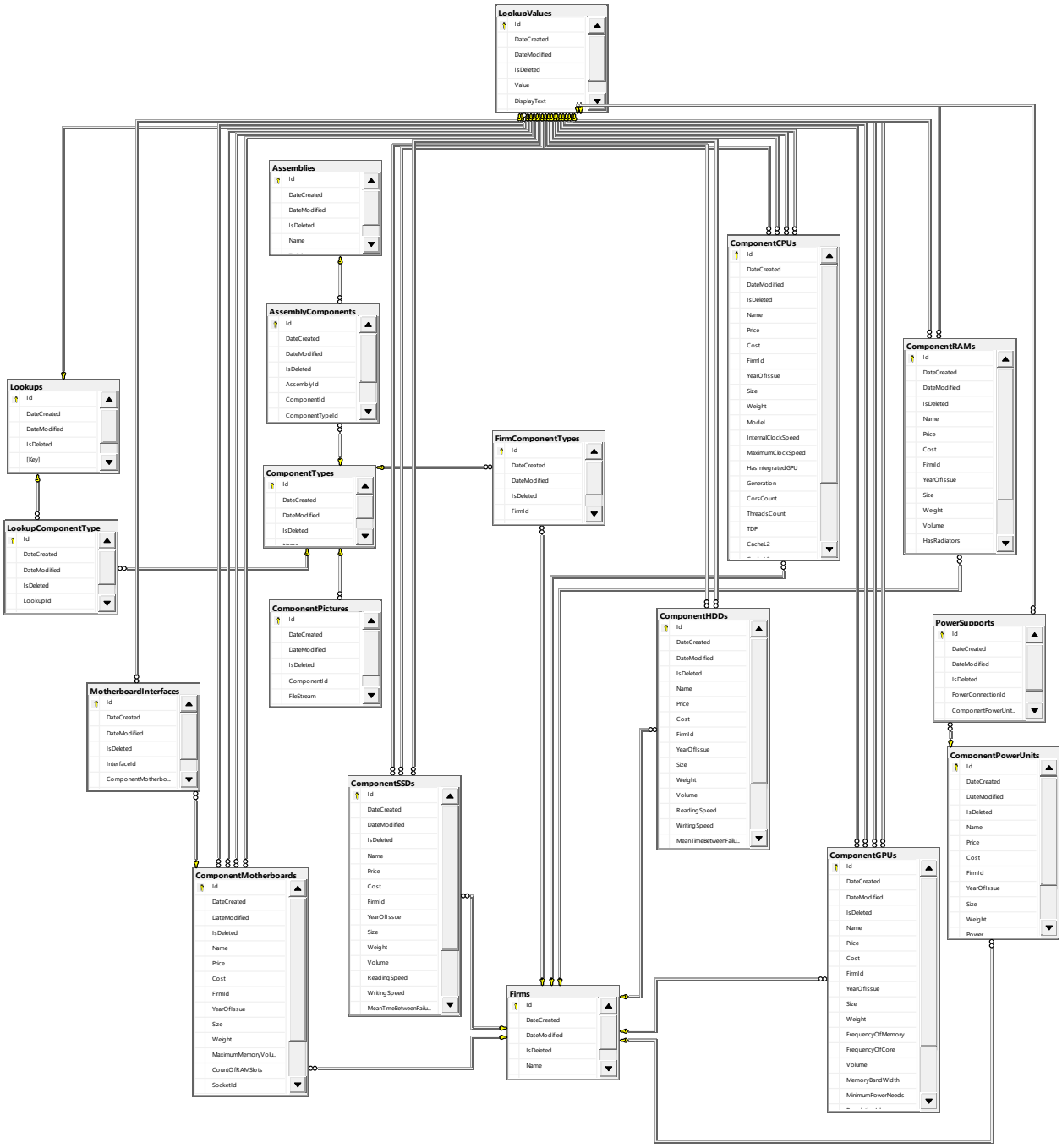

Рисунок 27 – ER-діаграма

ER-діаграма для мобільного додатку включає до себе 19 таблиць, це необхідно для коректного збегігання основної інформації щодо компютерного комплектуючого, та для зберігання і обробки зборок користувача.

Перелік таблиць та їх коротка характеристика представлено у таблиці 1, повне описання кожної таблиці наведено у додатку А.

Таблиця 1 – Назви та призначення таблиць БД

Назва таблиці	Призначення
ComponentSSDs	Компонент твердотілих накопичувачів системи (опціональна одиниця для збірки)
PowerSupports	Інтерфейси систем живлення
MotherboardInterfaces	Інтерфейси підключень до материнських плат
Sysdiagrams	Системні діаграми
Assemblies	Збереження існуючих комп’ютерних збірок
ComponentTypes	Перелік типів компонентів
Firms	Поточна інформація про фірми
Lookups	Словник, в якому зберігаються усі можливі групи інтерфейсів
AssemblyComponents	Компоненти збірки
ComponentPictures	Зображення, відноситься до компонента в системі
ComponentPowerUnits	Компонент живлення (необхідна одиниця для збірки)
FirmComponentTypes	Поєднання фірм та типів продукції, які вони виробляють
LookupComponentType	Зв’язок між типом компонента та інтерфейсами які він може приймати
LookupValues	Словник, в якому зберігаються усі можливі значення груп інтерфейсів
ComponentCPUs	Компонент центрального процесору системи (необхідна одиниця для збірки)
ComponentGPUs	Компонент графічного пристрою системи (необхідна одиниця для збірки)
ComponentHDDs	Компонент жорсткого диску системи (опціональна одиниця для збірки)

Аби впровадити більш розгорнуту інформацію, було згенеровано запит, який використовується для створення звіту про систему:

```
select t.name as [Назва таблиці],
       c.name as [Назва колонки],
       ty.name as [Тип даних],
       iif(c.max_length >= 0, cast(c.max_length as nvarchar(5)),
N'∞') as [Максимальний розмір],
       iif(c.is_nullable = 1, 'так', 'ні') as [Пусте значення],
       iif(c.is_identity = 1, 'так', 'ні') as [Ключ таблиці]
  from sys.tables t
  join sys.columns c on c.object_id = t.object_id
  join sys.types ty on c.system_type_id = ty.system_type_id
```

Розглянемо основну структуру проекта, яка дозволяє користувачам створеного API отримувати інформацію щодо даних збірки, компонентів, а також дозволяє видаляти збірки або компоненти конкретної зі збірок, оновлювати інформацію, кількість компонентів та їх склад та додавати нові компоненти.

Така структура має загальний вигляд стандартного контролера проекта, який має назву «AssemblyController»:

```
     [ApiController]
     [Route("[controller]")]
     public class AssemblyController : BaseComputerHard-
wareGuideController
     {

     public AssemblyController(IAppContext appContext) : base(ap-
pContext) { }

     }
```

Як бачимо з наведеного вище приклада, кожен контролер проекта буде мати постійний атрибут «ApiController», який дозволяє фреймворку надавати об’єктам різну поведінку. ASP.NET Core підтримує створення RESTful сервісів, також відомих як Web API, використовуючи C#. Для обробки запитів веб-API використовує контролери. Контролери у Web API – це класи, що походять від ControllerBase.

Контролери ASP.NET Core використовують проміжне програмне забезпечення для маршрутизації, щоб зіставити URL-адреси вхідних запитів і відобразити їх на дії.

Шаблони маршрутів:

- визначаються в коді або атрибутах запуску;
- опишіть, як шляхи URL-адреси підбираються до дій;
- використовуються для створення URL-адрес для посилань. Створені посилання, як правило, повертаються у відповідях.

Дії або умовно спрямовані, або атрибутивні. Розміщення маршруту на контролері або дії робить його маршрутизованим атрибутом.

Наступні ключові слова є зарезервованими назвами параметрів маршруту при використанні контролерів або сторінок Razor:

- action;
- area;
- controller;
- handler;
- page.

Наступним кроком будуть розглянуті основні типи методів та їх використання в поточній структурі. По-перше, необхідно обговорити CRUD-систему.

CRUD (англ. «Create-Read-Update-Delete») – 4 базових функцій управління даними: створення, зміна, читання та видалення. CRUD також використовується для опису конвенцій користувальницького інтерфейсу, які полегшують перегляд, пошук та зміну інформації; часто використовуючи комп’ютерні форми та звіти.

CRUD абревіатура відноситься до всіх основних функцій, які реалізовані в реляційних додатках бази даних (табл. 2). Кожна літера в абревіатурі може зіставляти стандартний виклад структуризованої мови запитів (SQL), метод протоколу передачі гіпертексту (HTTP) (зазвичай використовується для створення REST API) або операції служби розподілу даних (DDS).

Таблиця 2 – Відповідність команд у різних мовах

Operation	SQL	HTTP	REST API	DDS	MongoDB
Create	INSERT	PUT / POST	POST	write	Insert
Read	SELECT	GET	GET	read / take	Find
Update	UPDATE	PUT / POST / PATCH	PUT	write	Update
Delete	DELETE	DELETE	DELETE	dispose	Remove

Операція створення (Create), як бачимо із значення слова, дозволяє додавати нову інформацію, в випадку проекта, це стосується додавання нових збірок та компонентів.

```
[HttpGet]
public async Task<IEnumerable<Assembly>> Get()
{
    var query = AppContext.Assemblies.AsQueryable();
    return await query.ToListAsync();
}

[HttpGet("{id}")]
public async Task<GetAssemblyVM> Get(int id)
{
    var query = AppContext.Assemblies.Include(x => x.Assem-
blyComponents).ThenInclude(x => x.ComponentType).AsQueryable();
    var assembly = await query.FirstOrDefaultAsync(x => x.Id
== id);
    var total = 0.0;
    if (assembly.AssemblyComponents?.Count > 0)
    {
        foreach (var item in assembly.AssemblyComponents)
        {
            var component = GetQuery(item.ComponentType.Com-
ponentTypeEnumeration).FirstOrDefault(x=>x.Id  ==  item.Componen-
tId);
            LoadPictures(component, 1);
            item.BaseComponent = component;
            total += component.Price * item.Quantity;
        }
    }
    return new GetAssemblyVM
    {
        Assembly = assembly,
        Total = total,
    };
```

Кожен метод має асинхроний позначник, який дозволяє серверу опрацьовувати запити користувачів паралельно, що допомагає прискорити процес отримання інформації при масовому користуванню додатком.

Перший метод Get, який немає аргументів на вхід, повертає користувачу API список збірок, які вже були створені в системі.

Цей список має однорідні об’єкти типа «Assembly»:

- Id;
- Name;
- ToPrice;
- AssemblyComponents;
- DateCreated;
- DateModified;
- IsDeleted.

Крім того, слід зазначити, що для отримання списку збірок після використання методу був обраний інтерфейс «IEnumerable», який описує стандартні масиви даних, для будь якої мови або технології. Завдяки цьому, кожний парсер має змогу отримувати та відтворювати список у правильному форматі без деформації цих даних.

Наступний метод Get(int id), дозволяє отримувати усі компоненти, які були додані до збірки за ідентифікатором. Важливо звернути увагу на атрибут HttpGet("{id}"), який дозволяє записувати аргументи використовуючи більш просту форму передачі, у вигляді сегмента HTTP-запита.

Цей метод в першу чергу отримує загальну інформацію про збірку за ідентифікатором, якщо вона існує. Якщо ж ідентифікатор не існує в системі, тоді користувач отримує лист без компонентів або так званий пустий масив.

Для кожного з компонентів необхідно загрузити зображення, а також підрахувати суму за кількість товара, крім того необхідно знайти суму усіх таких локальних цін, щоб відобразити загальну вартість збірки.

У відповіді сервера маємо дві характеристики об’єкта:

– Total;

– Assembly.

В свою чергу, другий об’єкт був розглянутий вище, але слід зазначити, що в цьому об’єкті є список AssemblyComponents, який містить набір компонентів, які варто розглянути:

– Assembly;

– ComponentId;

– ComponentType;

– Quantity;

– BaseComponent;

– інші службові поля.

Отже бачимо, що завдяки такій структурі, можна мати можливість вкладати об’єкти один в одного та отримувати ієрархію без зайвих запитів до бази даних.

Наступними методами будуть методи Post:

```
[HttpPost]
public async Task<IActionResult> Post(Assembly assembly)
{
    try
    {
        AppContext.Assemblies.Add(assembly);
        await AppContext.SaveChangesAsync();
    }
    catch (Exception ex)
    {
        return BadRequest(ex);
    }

    return CreatedAtAction(nameof(Post), assembly);
}
```

Діні методи які дозволяють викликати операцію Create, для створення нових об’єктів в системі:

```
[HttpPost("component")]
public async Task<IActionResult> Post(AddAssemblyComponentVM
model)
{
    var assemblyComponent = (AssemblyComponent)null;
    try
    {
        var assembly = await AppContext.Assemblies.FirstOrDe-
faultAsync(x => x.Id == model.AssemblyId);
        var component = await GetQuery(model.Type).FirstOrDe-
faultAsync(x => x.Id == model.ComponentId);
        var componentType = await AppContext.Compo-
nentTypes.FirstOrDefaultAsync(x => x.Id == (int)model.Type);
```

Обираємо дані за критеріями користувача:

```
    assemblyComponent = new AssemblyComponent
        {
            Assembly = assembly,
            ComponentId = component.Id,
            ComponentType = componentType,
            Quantity = model.Quantity,
        };
        AppContext.AssemblyComponents.Add(assemblyComponent);
        await AppContext.SaveChangesAsync();
    }
    catch (Exception ex)
    {
        return BadRequest(ex);
    }

    return CreatedAtAction(nameof(Post), assemblyComponent);
}
```

У першому методі можна побачити прийняття аргумента типа Assembly, який потрапляє в метод додавання Add у фреймворку Entity Framework Core. Таким чином після виклику методу SaveChangesAsync додаток зберігає нові об’єкти в системі баз даних.

На прикладі другого методу можна побачити використання атрибуту HttpPost("component"), який дозволяє змінити стандартний шлях маршрутизації на новий, в даному випадку API буде використовувати цей метод лише тоді, коли тіло запиту буде мати сегмент component після назви контролера.

Нижче представлені інші методи, такі як:

– Put(UpdateAssemblyVM model);

– Put(UpdateAssemblyComponentVM model);

– Delete(int assemblyId);

– Delete(int assemblyId, int componentId).

Кожен з яких дозволяє оновлювати дані щодо конкретних збірок, оновлювати інформацію відносно компонентів збірки, видаляти збірки з системи або видаляти конкретний компонент із існуючої збірки, відповідно до листу зазначених вище операцій. Усі ці операції як і було зазначено вище створюють єдиний інтерфейс CRUD.

```
[HttpPut]
public async Task<IActionResult> Put(UpdateAssemblyVM model)
{
    var assembly = (Assembly)null;
    try
    {
        assembly = await AppContext.Assemblies.FirstOrDe-
faultAsync(x => x.Id == model.AssemblyId);
        if (!string.IsNullOrWhiteSpace(model.Name))
        {
            assembly.Name = model.Name;
        }
        if (model.ToPrice.HasValue)
        {
            assembly.ToPrice = model.ToPrice.Value;
        }

        AppContext.Assemblies.Update(assembly);
        await AppContext.SaveChangesAsync();
    }
    catch (Exception ex)
    {       return BadRequest(ex);    }
    return Ok(assembly);}
```

Метод для оновлення даних по компонентам обраної збірки:

```
[HttpPut("component")]
public async Task<IActionResult> Put(UpdateAssemblyCompo-
nentVM model)
{
    var assemblyComponent = (AssemblyComponent)null;
    try
    {
```

```
            if (model.Quantity < 1)
            {
             throw new ArgumentException("Quantity of assembly
component should be at least 1.", "quantity");
            }

           assemblyComponent = await AppContext.AssemblyCompo-
nents.FirstOrDefaultAsync(x => x.Id == model.AssemblyComponen-
tId);
       assemblyComponent.Quantity = model.Quantity;
      AppContext.AssemblyComponents.Update(assemblyComponent);
            await AppContext.SaveChangesAsync();
        }
        catch (Exception ex)
        {
            return BadRequest(ex);
        }
        return Ok(assemblyComponent);
    }
```

Методи для видалення даних по компонентам обраної збірки:

```
    [HttpDelete("component")]
    public async Task<IActionResult> Delete(int assemblyId, int
componentId)
    {
        //...
    }
    [HttpDelete]
    public async Task<IActionResult> Delete(int assemblyId)
    {
        //...
    }
```

ВИСНОВКИ

Пошук набору онлайн-інструментів, які допоможуть підібрати сумісні між собою комплектуючі та порівняти ціни в максимально простій формі є головною метою для багатьох покупців. Тому тема дипломного прокту є дуже актуальною.

На етапі аналвзу було виконано:

1. Розглянута предметная область.
2. Проаналізовані аналоги програмного додатку для виявлення їх найбільш сильних на слабких характеристик.
3. Сформовані вимоги до об’єкту розробки.

В ролі засобів розробки після детального розгляду існуючих було прийнято рішення зупинитися на:

1. MS SQL Server 2017.
2. Платформа ASP.NET Core.
3. Entity Framework Core.
4. Android-додаток Xamarin.Forms.

На етапі проектування за допомогою UML-засобів смодельовані такі діаграми:

1. Діаграма варіантів використання.
2. Діаграма діяльності для аглоритму пошуку компонентів.

В результаті розроблено мобільний додаток з пошитеною БД для підбору компютерного обладнення. Спроектован оптимальний інтерфейс для користувача. В подальшому додаток можна поширити функціоналом.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Полезные утилиты для диагностики и тестирования железа. URL: https://compress.ru/article.aspx?id=22304. (дата звернення 18.03.2020).
2. Полезные утилиты для диагностики и тестирования железа. URL: https://compress.ru/article.aspx?id=22304. (дата звернення 15.03.2020).
3. Офіційний сайт компанії telemart. URL: https://telemart.ua. (дата звернення 15.03.2020).
4. Конфигуратор компьютера NerdPart. URL: https://telemart.ua/assembly-start.html. (дата звернення 15.03.2020).
5. Конфигуратор компьютера «dns-shop». URL: https://www.dns-shop.ru/configurator/. (дата звернення 16.03.2020).
6. 10 причин перейти на Microsoft SQL Server 2019. URL: https://soft-line.ru/about/blog/10-prichin-pereyti-na-microsoft-sql-server-2019. (дата звернення 20.03.2020).
7. Обзор основных нововведений в Microsoft SQL Server 2017. URL: https://info-comp.ru/novosti/594-review-microsoft-sql-server-2017.html. (дата звернення 20.03.2020).
8. Введение в ASP.NET Core. ASP.NET Core – новая эпоха в развитии ASP.NET. URL: https://metanit.com/sharp/aspnet5/1.1.php. (дата звернення 22.03.2020).
9. Введение в Entity Framework Core. Что такое Entity Framework Core. URL: https://metanit.com/sharp/entityframeworkcore/1.1.php. (дата звернення 22.03.2020).
10. Начинаем работу с ASP.NET Core и Entity Framework 6. URL: Введение в Entity Framework Core. (дата звернення 25.03.2020).
11. Разработка приложений для Android с C#. URL: https://habr.com/ru/post/169467/. (дата звернення 27.03.2020).

12. Проектирование программного обеспечения с использованием стандартов uml 2. 0 и SysML 1. 0. URL: https://cyberleninka.ru/article/n/proektirovanie-programmnogo-obespecheniya-s-ispolzovaniem-standartov-uml-2-0-i-sysml-1-0. (дата звернення 27.03.2020).
13. UML – диаграмма вариантов использования (use case diagram). URL: https://habr.com/ru/post/47940/. (дата звернення 27.03.2020).
14. Построение диаграммы вариантов использования. URL: https://flexberry.github.io/ru/gpg_use-case-diagram.html. (дата звернення 27.03.2020).
15. Теория и практика UML. Диаграмма деятельности. URL: https://it-gost.ru/articles/view_articles/96. (дата звернення 28.03.2020).

Д О Д А Т К И

Додаток А – Опис таблиць БД

Таблиця – «ComponentSSDs»

Назва таблиці	Назва колонки	Тип даних	Макс. роз-мір	Пусте значення	Ключ табл.
ComponentSSDs	Id	int	4	ні	так
ComponentSSDs	DateCreated	datetime2	8	ні	ні
ComponentSSDs	DateModified	datetime2	8	ні	ні
ComponentSSDs	IsDeleted	bit	1	ні	ні
ComponentSSDs	Name	nvarchar	∞	так	ні
ComponentSSDs	Name	sysname	∞	так	ні
ComponentSSDs	Price	float	8	ні	ні
ComponentSSDs	Cost	float	8	так	ні
ComponentSSDs	FirmId	int	4	так	ні
ComponentSSDs	YearOfIssue	int	4	так	ні
ComponentSSDs	Size	nvarchar	∞	так	ні
ComponentSSDs	Size	sysname	∞	так	ні
ComponentSSDs	Weight	float	8	так	ні
ComponentSSDs	Volume	float	8	так	ні
ComponentSSDs	ReadingSpeed	int	4	так	ні
ComponentSSDs	WritingSpeed	int	4	так	ні
ComponentSSDs	MeanTimeBetweenFailures	int	4	так	ні
ComponentSSDs	ReadOnlyMemoryFormFactorId	int	4	так	ні
ComponentSSDs	InterfaceId	int	4	так	ні
ComponentSSDs	TypeOfCellMemoryId	int	4	так	ні

Таблиця – «PowerSupports»

Назва таблиці	Назва колонки	Тип даних	Макс. розмір	Пусте значення	Ключ табл.
PowerSupports	Id	int	4	ні	так
PowerSupports	DateCreated	datetime2	8	ні	ні
PowerSupports	DateModified	datetime2	8	ні	ні
PowerSupports	IsDeleted	bit	1	ні	ні
PowerSupports	PowerConnectionId	int	4	ні	ні
PowerSupports	ComponentPowerUnitId	int	4	ні	ні
PowerSupports	Count	int	4	ні	ні

Таблиця – «MotherboardInterfaces»

Назва таблиці	Назва колонки	Тип даних	Макс. розмір	Пусте значення	Ключ табл.
MotherboardInterfaces	Id	int	4	ні	так
MotherboardInterfaces	DateCreated	datetime2	8	ні	ні
MotherboardInterfaces	DateModified	datetime2	8	ні	ні
MotherboardInterfaces	IsDeleted	bit	1	ні	ні
MotherboardInterfaces	InterfaceId	int	4	ні	ні
MotherboardInterfaces	ComponentMotherboardId	int	4	ні	ні
MotherboardInterfaces	Count	int	4	ні	ні

Таблиця – «sysdiagrams»

Назва таблиці	Назва колонки	Тип даних	Макс. розмір	Пусте значення	Ключ табл.
sysdiagrams	name	nvarchar	256	ні	ні
sysdiagrams	name	sysname	256	ні	ні
sysdiagrams	principal_id	int	4	ні	ні
sysdiagrams	diagram_id	int	4	ні	так
sysdiagrams	version	int	4	так	ні
sysdiagrams	definition	varbinary	∞	так	ні

Таблиця – «Assemblies»

Назва таблиці	Назва колонки	Тип даних	Макс. розмір	Пусте значення	Ключ табл.
Assemblies	Id	int	4	ні	так
Assemblies	DateCreated	datetime2	8	ні	ні
Assemblies	DateModified	datetime2	8	ні	ні
Assemblies	IsDeleted	bit	1	ні	ні
Assemblies	Name	nvarchar	∞	так	ні
Assemblies	Name	sysname	∞	так	ні
Assemblies	ToPrice	int	4	ні	ні

Таблиця – «Firms»

Назва таблиці	Назва колонки	Тип даних	Макс. розмір	Пусте значення	Ключ табл.
ComponentTypes	Id	int	4	ні	так
ComponentTypes	DateCreated	datetime2	8	ні	ні
ComponentTypes	DateModified	datetime2	8	ні	ні
ComponentTypes	IsDeleted	bit	1	ні	ні
ComponentTypes	Name	nvarchar	∞	так	ні
ComponentTypes	Name	sysname	∞	так	ні

Таблиця – «Firms»

Назва таблиці	Назва колонки	Тип даних	Макс. розмір	Пусте значення	Ключ табл.
Firms	Id	int	4	ні	так
Firms	DateCreated	datetime2	8	ні	ні
Firms	DateModified	datetime2	8	ні	ні
Firms	IsDeleted	bit	1	ні	ні
Firms	Name	nvarchar	∞	так	ні
Firms	Name	sysname	∞	так	ні
Firms	Country	nvarchar	∞	так	ні
Firms	Country	sysname	∞	так	ні

Таблиця – «Lookups»

Назва таблиці	Назва колонки	Тип даних	Макс. розмір	Пусте значення	Ключ табл.
Lookups	Id	int	4	ні	так
Lookups	DateCreated	datetime2	8	ні	ні
Lookups	DateModified	datetime2	8	ні	ні
Lookups	IsDeleted	bit	1	ні	ні
Lookups	Key	nvarchar	∞	так	ні
Lookups	Key	sysname	∞	так	ні
Lookups	Name	nvarchar	∞	так	ні
Lookups	Name	sysname	∞	так	ні

Таблиця – «AssemblyComponents»

Назва таблиці	Назва колонки	Тип даних	Макс. розмір	Пусте значення	Ключ табл.
AssemblyComponents	Id	int	4	ні	так
AssemblyComponents	DateCreated	datetime2	8	ні	ні
AssemblyComponents	DateModified	datetime2	8	ні	ні
AssemblyComponents	IsDeleted	bit	1	ні	ні
AssemblyComponents	AssemblyId	int	4	так	ні
AssemblyComponents	ComponentId	int	4	ні	ні
AssemblyComponents	ComponentTypeId	int	4	так	ні
AssemblyComponents	Quantity	int	4	ні	ні

Таблиця – «ComponentPictures»

Назва таблиці	Назва колонки	Тип даних	Макс. розмір	Пусте значення	Ключ табл.
ComponentPictures	Id	int	4	ні	так
ComponentPictures	DateCreated	datetime2	8	ні	ні
ComponentPictures	DateModified	datetime2	8	ні	ні
ComponentPictures	IsDeleted	bit	1	ні	ні
ComponentPictures	ComponentId	int	4	ні	ні
ComponentPictures	FileStream	varbinary	∞	так	ні
ComponentPictures	ComponentTypeId	int	4	так	ні

Таблиця – «ComponentPowerUnits»

Назва таблиці	Назва колонки	Тип даних	Макс. розмір	Пусте значення	Ключ табл.
ComponentPowerUnits	Id	int	4	ні	так
ComponentPowerUnits	DateCreated	datetime2	8	ні	ні
ComponentPowerUnits	DateModified	datetime2	8	ні	ні
ComponentPowerUnits	IsDeleted	bit	1	ні	ні
ComponentPowerUnits	Name	nvarchar	∞	так	ні
ComponentPowerUnits	Name	sysname	∞	так	ні
ComponentPowerUnits	Price	float	8	ні	ні
ComponentPowerUnits	Cost	float	8	так	ні
ComponentPowerUnits	FirmId	int	4	так	ні
ComponentPowerUnits	YearOfIssue	int	4	так	ні
ComponentPowerUnits	Size	nvarchar	∞	так	ні
ComponentPowerUnits	Size	sysname	∞	так	ні
ComponentPowerUnits	Weight	float	8	так	ні
ComponentPowerUnits	Power	int	4	так	ні
ComponentPowerUnits	FanDiameter	int	4	так	ні
ComponentPowerUnits	CountOfSATA	int	4	так	ні

Таблиця – «LookupComponentType»

Назва таблиці	Назва колонки	Тип даних	Макс. розмір	Пусте значення	Ключ табл.
FirmComponentTypes	Id	int	4	ні	так
FirmComponentTypes	DateCreated	datetime2	8	ні	ні
FirmComponentTypes	DateModified	datetime2	8	ні	ні
FirmComponentTypes	IsDeleted	bit	1	ні	ні
FirmComponentTypes	FirmId	int	4	так	ні
FirmComponentTypes	ComponentTypeId	int	4	так	ні
LookupComponentType	Id	int	4	ні	так
LookupComponentType	DateCreated	datetime2	8	ні	ні
LookupComponentType	DateModified	datetime2	8	ні	ні
LookupComponentType	IsDeleted	bit	1	ні	ні
LookupComponentType	LookupId	int	4	так	ні
LookupComponentType	ComponentTypeId	int	4	так	ні

Таблиця – «LookupValues»

Назва таблиці	Назва колонки	Тип даних	Макс. розмір	Пусте значення	Ключ табл.
LookupValues	Id	int	4	ні	так
LookupValues	DateCreated	datetime2	8	ні	ні
LookupValues	DateModified	datetime2	8	ні	ні
LookupValues	IsDeleted	bit	1	ні	ні
LookupValues	Value	nvarchar	∞	так	ні
LookupValues	Value	sysname	∞	так	ні
LookupValues	DisplayText	nvarchar	∞	так	ні
LookupValues	DisplayText	sysname	∞	так	ні
LookupValues	LookupId	int	4	так	ні

Таблиця – «ComponentRAMs»

Назва таблиці	Назва колонки	Тип даних	Макс. розмір	Пусте значення	Ключ табл.
ComponentRAMs	Id	int	4	ні	так
ComponentRAMs	DateCreated	datetime2	8	ні	ні
ComponentRAMs	DateModified	datetime2	8	ні	ні
ComponentRAMs	IsDeleted	bit	1	ні	ні
ComponentRAMs	Name	nvarchar	∞	так	ні
ComponentRAMs	Name	sysname	∞	так	ні
ComponentRAMs	Price	float	8	ні	ні
ComponentRAMs	Cost	float	8	так	ні
ComponentRAMs	FirmId	int	4	так	ні
ComponentRAMs	YearOfIssue	int	4	так	ні
ComponentRAMs	Size	nvarchar	∞	так	ні
ComponentRAMs	Size	sysname	∞	так	ні
ComponentRAMs	Weight	float	8	так	ні
ComponentRAMs	Volume	float	8	так	ні
ComponentRAMs	HasRadiators	bit	1	так	ні
ComponentRAMs	Frequency	int	4	так	ні
ComponentRAMs	RAMTypeId	int	4	так	ні
ComponentRAMs	TimingsId	int	4	так	ні

Таблиця – «ComponentHDDs»

Назва таблиці	Назва колонки	Тип даних	Макс. розмір	Пусте значення	Ключ табл.
ComponentHDDs	Id	int	4	ні	так
ComponentHDDs	DateCreated	datetime2	8	ні	ні
ComponentHDDs	DateModified	datetime2	8	ні	ні
ComponentHDDs	IsDeleted	bit	1	ні	ні
ComponentHDDs	Name	nvarchar	∞	так	ні
ComponentHDDs	Name	sysname	∞	так	ні
ComponentHDDs	Price	float	8	ні	ні
ComponentHDDs	Cost	float	8	так	ні
ComponentHDDs	FirmId	int	4	так	ні
ComponentHDDs	YearOfIssue	int	4	так	ні
ComponentHDDs	Size	nvarchar	∞	так	ні
ComponentHDDs	Size	sysname	∞	так	ні
ComponentHDDs	Weight	float	8	так	ні
ComponentHDDs	Volume	float	8	так	ні
ComponentHDDs	ReadingSpeed	int	4	так	ні
ComponentHDDs	WritingSpeed	int	4	так	ні

Таблиця – «ComponentGPUs»

Назва таблиці	Назва колонки	Тип даних	Макс. розмір	Пусте значення	Ключ табл.
ComponentCPUs	Id	int	4	ні	так
ComponentCPUs	DateCreated	datetime2	8	ні	ні
ComponentCPUs	DateModified	datetime2	8	ні	ні
ComponentCPUs	IsDeleted	bit	1	ні	ні
ComponentCPUs	Name	nvarchar	∞	так	ні
ComponentCPUs	Name	sysname	∞	так	ні
ComponentCPUs	Price	float	8	ні	ні
ComponentCPUs	Cost	float	8	так	ні
ComponentCPUs	FirmId	int	4	так	ні
ComponentCPUs	YearOfIssue	int	4	так	ні
ComponentCPUs	Size	nvarchar	∞	так	ні
ComponentCPUs	Size	sysname	∞	так	ні
ComponentCPUs	Weight	float	8	так	ні
ComponentCPUs	Model	nvarchar	∞	так	ні
ComponentCPUs	Model	sysname	∞	так	ні
ComponentCPUs	InternalClockSpeed	float	8	так	ні
ComponentCPUs	MaximumClockSpeed	float	8	так	ні
ComponentCPUs	HasIntegratedGPU	bit	1	так	ні
ComponentCPUs	Generation	int	4	так	ні
ComponentCPUs	CorsCount	int	4	так	ні
ComponentCPUs	ThreadsCount	int	4	так	ні
ComponentCPUs	TDP	float	8	так	ні
ComponentCPUs	CacheL2	int	4	так	ні
ComponentCPUs	CacheL3	int	4	так	ні
ComponentCPUs	SocketId	int	4	так	ні
ComponentCPUs	ProcessorFamilyId	int	4	так	ні
ComponentCPUs	RAMTypeId	int	4	так	ні
ComponentCPUs	ProcessId	int	4	так	ні
ComponentGPUs	Id	int	4	ні	так
ComponentGPUs	DateCreated	datetime2	8	ні	ні
ComponentGPUs	DateModified	datetime2	8	ні	ні
ComponentGPUs	IsDeleted	bit	1	ні	ні
ComponentGPUs	Name	nvarchar	∞	так	ні
ComponentGPUs	Name	sysname	∞	так	ні

Таблиця – «ComponentMotherboards»

Назва таблиці	Назва колонки	Тип даних	Макс. розмір	Пусте значення	Ключ табл.
ComponentMotherboards	Id	int	4	ні	так
ComponentMotherboards	DateCreated	datetime2	8	ні	ні
ComponentMotherboards	DateModified	datetime2	8	ні	ні
ComponentMotherboards	IsDeleted	bit	1	ні	ні
ComponentMotherboards	Name	nvarchar	∞	так	ні
ComponentMotherboards	Name	sysname	∞	так	ні
ComponentMotherboards	Price	float	8	ні	ні
ComponentMotherboards	Cost	float	8	так	ні
ComponentMotherboards	FirmId	int	4	так	ні
ComponentMotherboards	YearOfIssue	int	4	так	ні
ComponentMotherboards	Size	nvarchar	∞	так	ні
ComponentMotherboards	Size	sysname	∞	так	ні
ComponentMotherboards	Weight	float	8	так	ні
ComponentMotherboards	MaximumMemoryVolume	float	8	так	ні
ComponentMotherboards	CountOfRAMSlots	int	4	так	ні
ComponentMotherboards	SocketId	int	4	так	ні
ComponentMotherboards	ChipsetId	int	4	так	ні
ComponentMotherboards	SizeFormFactorId	int	4	так	ні
ComponentMotherboards	RAMTypeId	int	4	так	ні