**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ТАРАСА ШЕВЧЕНКА**

Факультет комп’ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

**Кваліфікаційна робота**

**на здобуття ступеня бакалавра**

за спеціальністю 121 Інженерія програмного забезпечення

на тему:

**СТВОРЕННЯ ВЕБ-СЕРВІСУ ДЛЯ ВЕДЕННЯ ОБЛІКУ ТА СИСТЕМАТИЗАЦІЇ ІТ-ОБЛАДНАННЯ В ЗАКЛАДАХ ВИЩОЇ ОСВІТИ**

Виконав студент 4-го курсу

Ліпський Даниїл Олександрович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

Науковий керівник:

доцент, кандидат фіз.-мат. наук

Шевченко Володимир Петрович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

Засвідчую, що в цій роботі немає запозичень

з праць інших авторів без відповідних

посилань.

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

Роботу розглянуто й допущено до захисту

на засіданні кафедри інтелектуальних програмних систем

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 р.,

протокол № \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Завідувач кафедри

О. І. Провотар \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

Київ – 2021

ЗМІСТ

[РЕФЕРАТ 3](#_Toc72425988)

[ВСТУП 4](#_Toc72425989)

[Мета та завдання роботи. 4](#_Toc72425990)

[Об'єкт, методи й засоби розробки. 5](#_Toc72425991)

[Можливі сфери застосування. 5](#_Toc72425992)

[Взаємозв’язок з іншими роботами. 5](#_Toc72425993)

[Реалізований функціонал веб-сервісу. 6](#_Toc72425994)

[Актуальність роботи та підстави для її виконання. 6](#_Toc72425995)

[АРХІТЕКТУРА СИСТЕМИ 6](#_Toc72425996)

[Перелік використаних технологій 6](#_Toc72425997)

[Особливості фреймворка Django 8](#_Toc72425998)

[СЕРВЕРНА ЧАСТИНА СИСТЕМИ 9](#_Toc72425999)

[Моделі (Models) 9](#_Toc72426000)

[Функції відображення (Views) 11](#_Toc72426001)

[Форми (Forms) 15](#_Toc72426002)

[ІНТЕРФЕЙС СИСТЕМИ 18](#_Toc72426003)

[Основні відомості 18](#_Toc72426004)

[Інтерфейс для перегляду специфікацій 19](#_Toc72426005)

[Інтерфейс для створення об’єктів 30](#_Toc72426006)

[Інтерфейс для редагування об’єктів 36](#_Toc72426007)

[Інтерфейс для перегляду та створення актів виконаних робіт 42](#_Toc72426008)

[ІНТЕРФЕЙС КОРИСТУВАЧА 45](#_Toc72426009)

[Вікно авторизації 45](#_Toc72426010)

[Головне вікно програми 46](#_Toc72426011)

[Вікно додавання нового користувача 49](#_Toc72426012)

[Вікно вибору обладнання та комплектуючих для додавання 50](#_Toc72426013)

[Вікно додавання обладнання 51](#_Toc72426014)

[Вікно зі списком обладнання 52](#_Toc72426015)

[Вікно зі списком комплектуючих до обладнання 53](#_Toc72426016)

[Паспорт обладнання 54](#_Toc72426017)

[Вікно редагування обладнання 55](#_Toc72426018)

[Вікно перегляду актів виконаних робіт 56](#_Toc72426019)

[Вікно додавання акту виконаних робіт 57](#_Toc72426020)

[Вікно детального перегляду акту виконаних робіт 57](#_Toc72426021)

[ВИСНОВКИ 58](#_Toc72426022)

[ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 59](#_Toc72426023)

РЕФЕРАТ

Обсяг роботи 59 сторінок, 17 ілюстрацій, 5 джерел посилань.

Метою кваліфікаційної роботи є створення веб-сервісу для ведення обліку та систематизації ІТ-обладнання в закладах вищої освіти. Факультет “Комп’ютерних наук та кібернетики” Київського національного університету імені Тараса Шевченка було обрано як приклад закладу вищої освіти, який потребує точного обліку інформаційно-технічного обладнання. В межах факультету створено підрозділ – Інформаційно-обчислювальний сектор. Основною метою даного сектору є забезпечення технічної можливості проведення занять студентами в аудиторіях факультету. Однією з умов надання цієї можливості є забезпечення персональними комп’ютерами студентів даного факультету. В межах підрозділу провідним інженером була розроблена схема звітності та обліку персональних комп’ютерів та їх компонентів. Це дало змогу розробити швидку стратегію ведення ремонтних та профілактичних робіт обладнання. Суттєвим недоліком є нагромадження актів специфікацій персональних комп’ютерів та актів проведених робіт. Більш того, існує потреба у групуванні комплектуючих деталей за типом та призначенням.

Реалізований веб-сервіс є інструментом систематизації ІТ-обладнання в закладах вищої освіти. Очікується, що використання сервісу спростить ведення обліку ІТ-обладнання та надасть єдиний стандарт акту специфікації. Також, сервіс інтерпретує статистику по комплектуючим персональних комп’ютерів в динамічні діаграми з метою пришвидшення сприйняття інформації щодо їх кількості та технічних характеристик.

Сервіс було реалізовано в інтегрованому середовищі розробки для платформи Python – PyCharm з використанням високорівневого відкритого Python-фреймворка для розробки веб-систем – Django.

Очікується, що веб-сервіс знайде практичне застосування та він стане корисним інструментом в роботі.

ВСТУП

Мета та завдання роботи.

Сьогодні ІТ-галузь займає провідні позиції в технологічному розвитку. У кожній галузі зайнятості людей певною мірою використовується науково-технічна основа, розроблена програмістами. Розробники створили різні програмні та системні рішення для прискорення та спрощення робочого процесу компаній, які використовують ці системи.

Метою кваліфікаційної роботи є створення веб-сервісу для ведення обліку та систематизації ІТ-обладнання на підприємствах. Для досягнення цієї мети були розглянуті наступні завдання:

1. Дослідити поточну структуру ведення обліку ІТ-обладнання.
2. Створити список всіх наявних компонент ІТ-обладнання. В цей список ввійшли такі об’єкти: материнські плати, твердотільні накопичувачі, накопичувачі на жорстких магнітних дисках, блоки живлення, відеокарти, мережеві плати, звукові плати, оптичні приводи.
3. Розробити функціонально-логічну схему сервісу.
4. Спроектувати відповідні моделі, які будуть відповідати за опис окремих компонентів ІТ-обладнання.
5. Розробити форми для додавання та редагування моделей.
6. Реалізувати функції відображення які приймають Web-запити та повертають Web-відповіді.

Об'єкт, методи й засоби розробки.

Об’єктом розробки є веб-сервіс з найбільш зручним інтерфейсом для ведення обліку, моніторингу ІТ-обладнання підприємства та формування актів специфікацій персональних комп’ютерів .

Інструментом створення програмного засобу було обрано PyCharm - IDE з використанням високорівневого відкритого Python-фреймворка для розробки веб-систем – Django. Створення додатку у вигляді веб-сервісу не обмежує користувача платформою , все що потрібно - це наявність Інтернет з’єднання та веб-браузера на самому пристрої.

Можливі сфери застосування.

Програмний продукт може бути використаний для ведення обліку та систематизації в будь-якій установі, яка працює з ІТ-обладнанням (школа, університет, коледж, торгівельний центр, а також в офісних будівлях та бізнес-центрах).

Взаємозв’язок з іншими роботами.

За методами розробки та інструментальними засобами робота виконувалася сумісно з вивченням об’єктно-орієнтованого програмування, системного програмування та операційних систем.

Реалізований функціонал веб-сервісу.

* додавання/редагування/видалення комплектуючих до персональних комп’ютерів;
* перегляд статистики комплектуючих (зібраної на основі технічних характеристик, які описані в моделях);
* додавання/редагування/видалення персональних комп’ютерів;
* генерація паспорту вибраних персональних комп’ютерів у форматі pdf;
* додавання/видалення актів виконаних робіт;
* генерація актів виконаних робіт у форматі pdf;
* логіка авторизації та реєстрації в веб-сервісі;
* обмеження доступу до функціонала сервісу користувачам згідно їх прав доступу.

Актуальність роботи та підстави для її виконання.

Розробці програмного засобу передувала гостра необхідність в зручному способі обробки даних пов’язаних з ІТ-обладнанням на факультеті. Даний веб-сервіс спрощує та пришвидшує ведення обліку та систематизації ІТ-обладнання. Також сервіс замінює паперові акти специфікацій персональних комп’ютерів та акти проведених робіт.

На сьогоднішній день системні адміністратори ведуть всю документацію у паперовому вигляді. Але, на жаль, зберігання документів в такий спосіб не є оптимальним з практичної точки зору. Паперовий формат накладає часові обмеження на пошук потрібного акту проведених робіт або акту специфікації. До того ж існує проблема накопичення актів та дотримання єдиної структури їх заповнення.

АРХІТЕКТУРА СИСТЕМИ

Перелік використаних технологій

При розробці веб-сервісу було використано наступні технології:

Python – інтерпретована об’єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Має ряд переваг, які пришвидшують та спрощують розробку. Стандартний дистрибутив Python вже містить вагому кількість модулів різного функціоналу. Також розробники мають можливість створювати окремі модулі (в тестовому режимі) в діалоговому вікні. До того ж дана інтерпретована мова - простий та ефективний підхід до об’єктно-орієнтованого програмування.

Django – високорівневий відкритий Python-фреймворк для розробки веб-систем.

HTML – мова розмітки, котра використовується для структурування та відображення веб-сторінки разом з її контентом.

CSS – мова стилю HTML-сторінок, що використовується для опису їхнього зовнішнього вигляду. При розробці веб-сервісу було використано клієнтський фреймворк Bootstrap – набір інструментів з відкритим кодом для створення веб-сайтів та веб-додатків. Він містить у собі шаблони HTML та CSS. Використання даного фреймворка значно пришвидшило розробку веб-сервісу, а комбінування різних шаблонів призвело до створення зрозумілого та адаптивного дизайну.

JavaScript – скриптова мова програмування, яка використовується для створення сценаріїв веб-сторінки. Це надає можливість на стороні клієнта взаємодіяти з користувачем, обмінюватися даними з сервером , змінювати структуру веб-сторінки.

Chart.js - бібліотека JavaScript з відкритим кодом. Використовується для візуалізації даних шляхом побудови діаграм 8 типів: бар, лінія, область, пончик, бульбашка, радар, полярний та розсіяний.

Jinja - шаблонізатор, котрий дозволяє відокремити функціональну частину сайту від візуалізації. Даний шаблон дозволяє зробити HTML-сторінки більш гнучкими за допомогою перевірки умов, вивід масивів через цикл, вивід змінних та багато іншого. При використанні даного шаблонізатора очікується, що разом зі створенням розмітки HTML-документів розробник реалізує всередині серцевий код. Це дозволить реалізувати вагому функціональність за допомогою виразів та операторів шаблону Jinja.

SQLite3 - реляційна система керування базами даних, котра має простий та легкий у використані API, підтримує крос-платформовість, швидший за популярні рушії клієнт-серверних баз даних для найпоширеніших операцій.

Особливості фреймворка Django

Django – високорівневий відкритий Python-фреймворк для розробки веб-систем. Даний фреймворк має істотну архітектурну відмінність від інших. Сайт на Django створюється на основі окремих частин – додатків. Іншими слова, фреймворк Django має модульну структуру. Кожен модуль є незалежним та містить в собі окремий функціонал.

Під час розробки певного веб-сервісу, додатку або програмного забезпечення важливу роль відіграє архітектура. Дотримання архітектурного шаблону впродовж розробки сприяє більш швидкому знаходженні недоліків або помилок в коді та надає гнучкий дизайн, який полегшує майбутні зміни та розширення проекту. Також шаблон дозволяє повторне використання окремих модулів, компонент. Крім того дотримання даного архітектурного рішення дозволяє розробити систематизовану структуру з мінімальною складністю.

Основною архітектури Django проектів є архітектурний шаблон MVC. Цей шаблон дозволяє поділити розроблену систему на три частини: модель, представлення та контролер. Дана логіка застосовується для відокремлення даних від інтерфейсу користувача для того, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними. Також, це дозволяє вносити зміни в моделі без змін інтерфейсу користувача. Модель відповідальна за зберігання даних та їх структуру. Представлення (вигляд) відповідає за інтерфейс користувача, тобто представлення даних користувачеві. Контролер отримує вхідні дані й перетворює їх на команди для моделі чи вигляду.

У свою чергу розробники фреймворка Django змінили структуру та запропонували свій архітектурний шаблон MTV. Відмінність полягає в двох речах – контролер замінили на вид, а вид в класичній моделі MVC став шаблоном. Таким чином, архітектура “Модель-Вигляд-Контролер” перетворився на “Модель-Шаблон-Вигляд”.

Django підтримує парадигму ООП. Кожному типу об’єкта в базі даних ставиться у відповідність певна модель. Даний фреймворк містить у собі технологію ORM, яка надає можливість зв’язувати базу даних з концепціями ООП.

Під час обрання фреймворка на початковому етапі планування сервісу було розглянуто три варіанти фреймворка: Spring, Flask та Django. Однією з суттєвих переваг фреймворка Django є наявність власного веб-сервера. Сервер автоматично перезапускається при виявленні змін в сирцевому коді.

Також фреймворк Django дозволяє створювати легкі проекти з високим рівнем підтримки.

СЕРВЕРНА ЧАСТИНА СИСТЕМИ

Моделі (Models)

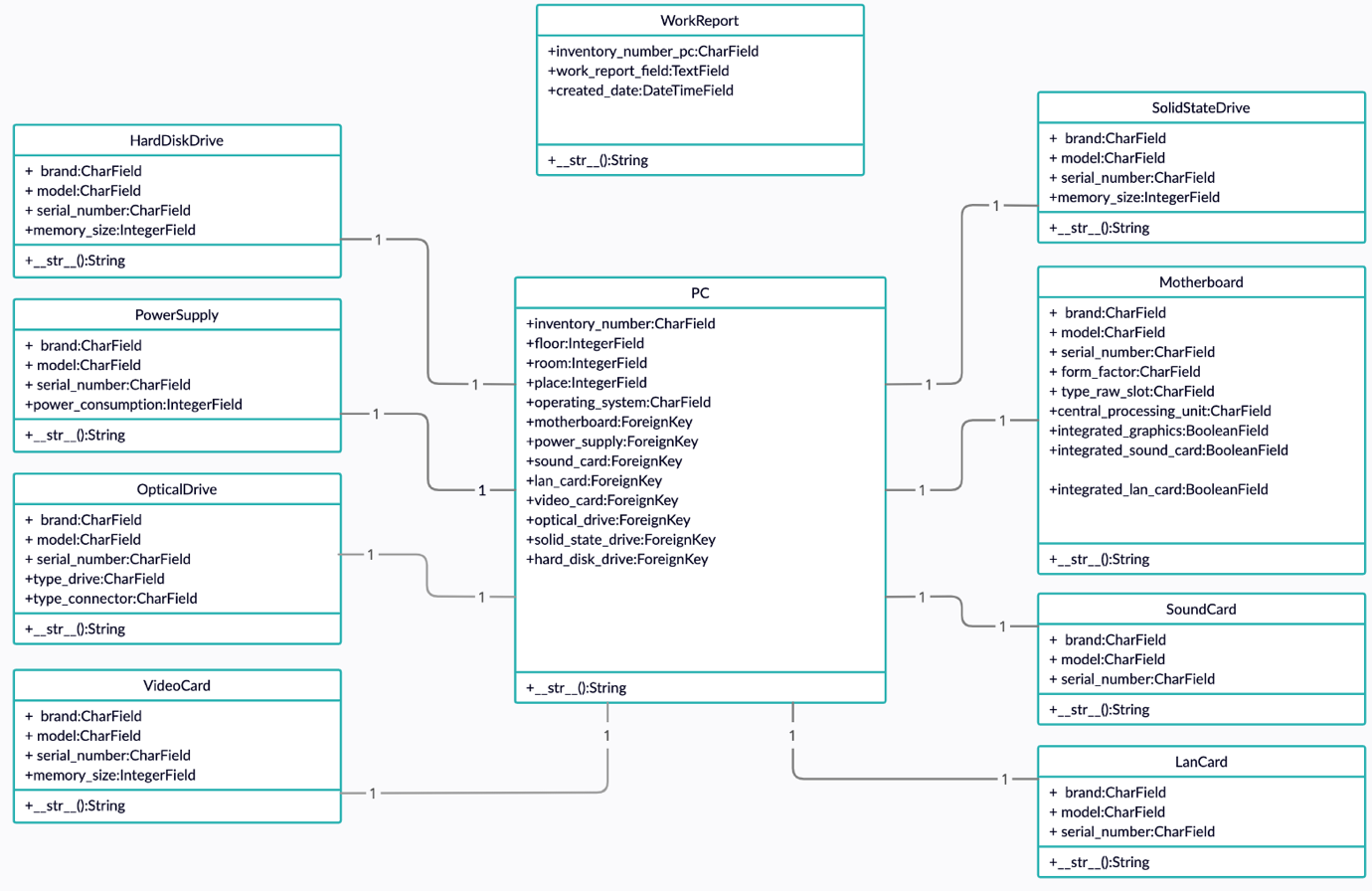
Веб-додатки Django отримують доступ і управляють даними через об'єкти Python, так званими моделями. Моделі визначають структуру даних, що зберігаються, включаючи типи полів і, можливо, їх максимальний розмір, значення за замовчуванням, параметри списку вибору, текст довідки для документації, текст міток для форм і т. д. Визначення моделі не залежить від основної бази даних - розробник можете вибрати один з декількох компонентів налаштування проекту. Після обрання бази даних не потрібно безпосередньо працювати з нею - розробник реалізує свою структуру моделі і код, а Django виконує роботу, пов'язану з базою даних самостійно.

Під час створення веб-сервісу було реалізовано 8 моделей, які описують комплектуючі персонального комп’ютера, одна модель відповідає за саме представлення персонального комп’ютера та одна модель обумовлює структуру акту проведених робіт.

В даній таблиці наведена основа інформація по реалізованим моделям:

|  |  |  |  |
| --- | --- | --- | --- |
| Назва моделі | Поле | Тип поля | Опис |
| Motherboard  (*Материнська плата*) | brand | CharField | Бренд |
| model | CharField | Модель |
| serial\_number | CharField | Серійний номер |
| form\_factor | CharField | Форм-фактор (наявний перелік для вибору) |
| type\_ram\_slot | CharField | Тип слоту для ОЗУ (наявний перелік для вибору) |
| central\_processing\_unit | CharField | Центральний процесор (наявний перелік для вибору, по назві) |
| integrated\_graphics | BooleanField | Інтегрована відеокарта (відмічається наявність) |
| integrated\_sound\_card | BooleanField | Інтегрована звукова-карта (відмічається наявність) |
| integrated\_lan\_card | BooleanField | Інтегрована мережева-карта (відмічається наявність) |
| SolidStateDrive  (*Твердотільний накопичувач*) | brand | CharField | Бренд |
| model | CharField | Модель |
| serial\_number | CharField | Серійний номер |
| memory\_size | IntegerField | Обсяг пам'яті (вказується в GB) |
| HardDiskDrive  (*Жорсткий магнітний диск*) | brand | CharField | Бренд |
| model | CharField | Модель |
| serial\_number | CharField | Серійний номер |
| memory\_size | IntegerField | Обсяг пам'яті (вказується в GB) |
| PowerSupply  (*Блок живлення*) | brand | CharField | Бренд |
| model | CharField | Модель |
| serial\_or\_inventory\_  number | CharField | Серійний номер |
| power\_consumption | IntegerField | Максимальна споживана потужність від мережі |
| VideoCard  (*Відео карта*) | brand | CharField | Бренд |
| model | CharField | Модель |
| serial\_number | CharField | Серійний номер |
| memory\_size | IntegerField | Обсяг пам'яті (вказується в MB) |
| LanCard  (*Мережева плата*) | brand | CharField | Бренд |
| model | CharField | Модель |
| serial\_number | CharField | Серійний номер |
| SoundCard  (*Звукова плата*) | brand | CharField | Бренд |
| model | CharField | Модель |
| serial\_number | CharField | Серійний номер |
| OpticalDrive  (*Оптичний накопичувач*) | brand | CharField | Бренд |
| model | CharField | Модель |
| serial\_number | CharField | Серійний номер |
| type\_drive | CharField | Тип оптичного привода (наявний перелік для вибору) |
| type\_connector | CharField | Тип роз'єму (наявний перелік для вибору) |
| PC  (*Персональний комп'ютор*) | inventory\_number | CharField | Інвентарний номер |
| floor | IntegerField | Номер поверху |
| room | IntegerField | Номер кабінету |
| place | IntegerField | Номер учбового місця |
| operating\_system | CharField | Операційна система |
| motherboard | ForeignKey | Материнська плата  (наявний динамічний перелік для вибору) |
| solid\_state\_drive | ForeignKey | Твердотільний накопичувач (наявний динамічний перелік для вибору) |
| hard\_disk\_drive | ForeignKey | Жорсткий магнітний диск (наявний динамічний перелік для вибору) |
| power\_supply | ForeignKey | Блок живлення (наявний динамічний перелік для вибору) |
| video\_card | ForeignKey | Відео карта (наявний динамічний перелік для вибору) |
| lan\_card | ForeignKey | Мережева плата (наявний динамічний перелік для вибору) |
| sound\_card | ForeignKey | Звукова плата (наявний динамічний перелік для вибору) |
| optical\_drive | ForeignKey | Оптичний накопичувач |
| WorkReport  (*Акт проведеної роботи*) | inventory\_number\_pc | CharField | Інвентарний номер персонального комп'ютера |
| work\_report\_field | TextField | Опис проведених робіт |
| created\_date | DateTimeField | Дата та час проведених робіт |

Загальну структуру моделей можна представити наступною схемою:



Діаграма 1

Функції відображення (Views)

Відображення - це основа веб-додатку, що отримує HTTP-запити від веб-клієнта та повертає HTTP-відповіді. Окрім цього вони взаємодіють з ресурсами фреймворка для доступу до баз даних, шаблонів та інших компонент розробки.

В рамках розробленого веб-сервісу було реалізовано два додатки - IT\_items та main. В модуль IT\_items винесена вся логіка, яка відповідає за створення, видалення, збереження персональних комп’ютерів та їх компонент. Модуль main відповідає за дочірній функціонал веб-сервісу (авторизація, реєстрація) та формує на головній сторінці веб-сервісу статистичні дані (таблиці та діаграми) по персональним комп’ютерам та їх компонент на основі запитів до бази даних.

У модулі IT\_items загалом реалізовано 34 метода. Більша частина методів реалізовані для роботи з описаними моделями та базою даних: створення, збереження, видалення, редагування об’єктів в базі даних. Наприклад, метод, який реалізує створення об’єкта - персональний комп’ютер:

*@check\_denied\_access\_add  
def add\_pc(request):  
 if request.method == 'POST':  
 form = AddPcForm(request.POST)  
 if form.is\_valid():  
 cd = form.cleaned\_data  
 item = PC(inventory\_number=cd['inventory\_number'], floor=cd['floor'], room=cd['room'], place=cd['place'],  
 operating\_system=cd['operating\_system'],  
 motherboard=cd['motherboard'],  
 solid\_state\_drive=cd['solid\_state\_drive'],  
 hard\_disk\_drive=cd['hard\_disk\_drive'],  
 power\_supply=cd['power\_supply'],  
 video\_card=cd['video\_card'],  
 lan\_card=cd['lan\_card'],  
 sound\_card=cd['sound\_card'],  
 optical\_drive=cd['optical\_drive'])  
 item.save()  
 return HttpResponseRedirect(reverse('IT\_items:IT\_items'))  
 else:  
 form = AddPcForm()  
 pass  
  
 return render(request, 'IT\_items/add\_pc.html', {'form': form})*

Логіка та реалізація форм в веб-сервісі разом з декораторами буде розглянуто далі. Основне в даному прикладі - процес отримання з шаблону на сервер значення відповідних полів за допомогою метода POST, створення нового об’єкта типу PC, його збереження до бази даних, переадресація користувача на шаблон з переліком всіх персональних комп’ютерів.

Також існує декілька функцій, які відображають певні шаблони без використання запитів до БД. Дані методи створені для переходу на певні сторінки веб-сервісу, вони несуть в собі транспортну функцію - клієнт пересувається по розділам (сторінкам) веб-сервісу. Наприклад, метод

*@check\_denied\_access\_add  
def add(request):  
 return render(request, 'IT\_items/add.html')*

перенаправляє користувача на шаблон зі списком тих об’єктів, які він може додати - обладнання та комплектуючі до них. Список реалізований у вигляді кнопок-посилань на відповідні шаблони для створення нових об’єктів.

Крім того, реалізовані методи, які роблять запити до бази даних з метою подальшої передачі об’єктів в шаблони. Наприклад, метод

*def IT\_items(request):  
 items\_pc = PC.objects.all()  
 return render(request, 'IT\_items/IT-items.html', {'items\_pc': items\_pc})*

зберігає в зміну items\_pc список всіх персональних комп’ютерів, наявних в базі даних. Потім поєднуємо заданий шаблон IT-items.html зі зміною items та повертаємо HTTP-відповідь з даними, які ми передали.

Також було реалізовано декілька декораторів. Декоратор - це функція обгортка, вона накладає додаткову логіку на виконання метода, до якого прикріплена. Наприклад, метод

*@check\_denied\_access\_add  
def add(request):  
 return render(request, 'IT\_items/add.html')*

Логіка цього метода була описана вище, але не до кінця. Перед виконанням метода *add* виконається метод *check\_denied\_access\_add*. Даний метод створений для перевірки прав авторизованого користувача на виконання дій, описаних в методі *add*. Іншими словами, метод *check\_denied\_access\_add* перевіряє - чи має даний користувач права для виклику метода *add*. Якщо так, то в методі *check\_denied\_access\_add* викликається метод *add*, інакшеспрацьовує перенаправлення на шаблон, який повідомляє користувачу про обмеження. Реалізація метода *check\_denied\_access\_add*

*def check\_denied\_access\_add(func):  
 def wrapper(request):  
 if not request.user.groups.filter(name\_\_in=['Редактор']).exists():  
 return render(request, 'IT\_items/denied\_access.html')  
 else:  
 return func(request)  
 return wrapper*

У модулі main реалізовані методи, які можна поділити на дві групи:

* робота з обліковими записами;
* реалізація запитів до бази даних с метою формування статистики.

Наведемо приклади з кожної групи. Метод *user\_sign\_up*

*def user\_sign\_up(request):  
 if request.method == 'POST':  
 form = SignUpForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 username = form.cleaned\_data.get('username')  
 raw\_password = form.cleaned\_data.get('password1')  
 user = authenticate(username=username, password=raw\_password)  
 group = form.cleaned\_data['group']  
 group.user\_set.add(user)  
 login(request, user)  
 return redirect('/')  
 else:  
 form = SignUpForm()  
 return render(request, 'main/signup.html', {'form': form})*

відповідає за реєстрацію нового користувача. За допомогою метода POST сервер отримує інформацію про нового користувача, яку ввів адміністратор, створює нового користувача та виконує вхід в його обліковий запис. Метод *user\_login* має схожу логіку, але використовує іншу форму в шаблоні та слугує для авторизації. Метод *user\_logout* викликає в собі стандартну функцію Django для виходу з облікового запису.

Також, в даному модулі реалізовано ряд методів для отримання даних з полів всіх об’єктів, які наявні в базі даних. Потім ці дані у вигляді окремих списків поєднуються з шаблоном головної сторінки веб-сервісу. На головній сторінці за допомогою технології Bootstrap та бібліотеки Chart.js динамічно створюються таблиці та графіки з даними про комплектуючі до персональних комп’ютерів.

Форми (Forms)

HTML-форма  - це група з одного або декількох полів на веб-сторінці, яка використовується для збору інформації від користувача для подальшої відправки на сервер. Форми є гнучким механізмом збору даних, оскільки мають цілий набір віджетів для введення різних типів даних: текстові поля, прапорці, перемикачі й багато іншого. В реалізованому веб-сервісі форми Django використовуються для створення нових об’єктів відповідних моделей. За допомогою форм ми описуємо всі необхідні поля, які потрібно отримати для формування того чи іншого об’єкта. Кожен тип об’єкта має свою визначену форму.

Форми є відносно безпечним способом взаємодії користувача клієнта і сервера, оскільки вони дозволяють відправляти дані в POST-запитах, застосовуючи захист від міжсайтової підробки запиту.

В загальному випадку процес реалізації форми на HTML є доволі складним. Розробник після опису самої форми повинен перевірити її валідність, перевірити введені користувачем дані на стороні сервера та клієнта. У разі виникнення помилок користувач повинен отримати повідомлення, що потрібно знову заповнити дану форму, але при цьому вказати де були помилки. Потім у разі успіху виконати необхідні операції та повідомити про це користувача.

Фреймворк Django містить у собі потужний стек технологій роботи з формами, а саме форми Django. Це дозволяє розробнику на стороні back-end визначити форму та її поля програмно. А потім використовувати поля реалізованої форми для генерації безпосереднього коду HTML-форми. Форми Django контролюють будь-які взаємодії користувача з формою - показує помилки, підказки та перевіряє валідацію. Розглянемо форму додавання материнської плати:

*class AddMotherboardForm(forms.ModelForm):  
 motherboard\_serial\_number = forms.CharField(label='Серійний номер', max\_length=30,required=True, localize=True,help\_text='Обов’язково',  
 error\_messages={ 'unique': 'Материнська плата з таким серійним номером вже існує.'},widget=widgets.TextInput(attrs={'size': 1, 'class': 'form-control'}))*

*motherboard\_brand = forms.CharField(label='Бренд', max\_length=30,  
 required=True,  
 help\_text='Обов’язково, у разі відсутності - вказати відсутньо',  
 widget=widgets.TextInput(attrs={'size': 1, 'class': 'form-control'}))  
 motherboard\_model = forms.CharField(label='Модель', max\_length=30,  
 required=True,  
 help\_text='Обов’язково, у разі відсутності - вказати відсутньо',  
 localize=True,  
 widget=widgets.TextInput(attrs={'size': 1, 'class': 'form-control'}))  
 motherboard\_form\_factor = forms.ChoiceField(label='Форм-фактор', choices=MOTHERBOARD\_FROM\_FACTORS,  
 required=False, localize=True, help\_text='Необов’язково',  
 widget=widgets.Select(attrs={'size': 1, 'class': 'form-control'}))  
 motherboard\_type\_ram\_slot = forms.ChoiceField(label='Тип слоту для ОЗУ', choices=TYPE\_RAM\_SLOTS,  
 required=False, localize=True, help\_text='Необов’язково',  
 widget=widgets.Select(attrs={'size': 1, 'class': 'form-control'}))  
 motherboard\_central\_processing\_unit = forms.ChoiceField(label='Центральний процесор', choices=TYPE\_CENTRAL\_PROCESSING\_UNIT,  
 required=False, localize=True, help\_text='Необов’язково',  
 widget=widgets.Select(  
 attrs={'size': 1, 'class': 'form-control'}))  
 motherboard\_integrated\_graphics = forms.BooleanField(label='Інтегрована відеокарта', localize=True,  
 required=False, help\_text='Вказати чи наявна',  
 widget=forms.CheckboxInput(attrs={'class': 'checkbox'}))  
 motherboard\_integrated\_sound\_card = forms.BooleanField(label='Інтегрована звукова-карта', localize=True,  
 required=False, help\_text='Вказати чи наявна', widget=forms.CheckboxInput(attrs={'class': 'checkbox'}))  
 motherboard\_integrated\_lan\_card = forms.BooleanField(label='Інтегрована мережева-карта', localize=True,  
 required=False, help\_text='Вказати чи наявна',  
 widget=forms.CheckboxInput(attrs={'class': 'checkbox'}))  
 class Meta:  
 model = Motherboard  
 fields = ['motherboard\_serial\_number', 'motherboard\_brand', 'motherboard\_model',  
 'motherboard\_integrated\_graphics', 'motherboard\_integrated\_sound\_card',  
 'motherboard\_integrated\_lan\_card', 'motherboard\_form\_factor']*

В даній формі поля описані за аналогією з моделлю материнської плати. Кожне поле, яке міститься в формі, створюється як об’єкт та містить у собі певні атрибути. Кожне поле містить перелік необхідних характеристик:

* label - підпис даного поля, ця інформація може бути використана на стороні HTML-шаблона. Підписавши поле, розробник робить форму зрозумілою для користувача. Користувач розуміє яке поле за що відповідає;
* required приймає значення True або False. Це дозволяє визначити чи є заповнення поля форми обов’язковим. Якщо значення True, а користувач не ввів в поле дані - форма Django самостійно покаже повідомлення про помилку (під кожен тип помилки користувач може задати окреме повідомлення). Якщо значення False й користувач не ввів в поле дані - Django підставить значення за замовчуванням (значення за замовчуванням розробник описує під час реалізації моделі);
* help\_text це повідомлення підказка. Це поле є необов’язковим, але є корисним для користувача, якщо за назвою поля не можливо однозначно точно з’ясувати, що потрібно ввести в дане поле;
* choices використовується, якщо поле типу ChoiceField. Даний атрибут приймає перелік значень. Користувач вибирає конкретне значення (своє ввести не може) й саме це значення прийме дане поле.

Кожна форма - це клас, який унаслідується від forms. ModelForm - форми Django. В кожному такому класі необхідно прописати метаклас для зв’язку відповідної форми з відповідною моделлю та також оголосити поля, які будуть задіяні в HTML-шаблонах.

ІНТЕРФЕЙС СИСТЕМИ

Основні відомості

HTML-документи обов’язкова частина будь-якого веб-сайту реалізованого за допомогою фреймворка Django. Кожний окремий додаток в рамках одного проекту містить свій набір сторінок HTML. Загалом кількість веб документів та розмір HTML-коду в них може бути доволі великою. Тому при реалізації даного веб-сервісу була використана технологія Jinja.

За допомогою даного шаблонізатора було визначено базову сторінку base\_main.html як шаблон. Він розташований в кореневому каталозі папки templates та не відноситься до жодного з реалізованих додатків.

В додатку main задіяні чотири HTML-сторінки:

* main.html;
* login.html;
* login\_invalid.html;
* signup.html.

В документі base\_main.html було визначено основну структуру сайту, стиль, відступи та заголовок з відповідною панеллю меню, який винесено в окремий фрейм. Використовуючи дану сторінку у якості шаблону, розробник збереже верхній фрейм та стилістику сайту. Це у свою чергу створить єдиний дизайн веб-сервісу.

При першому заході на веб-сервіс користувач потрапляє на сторінку login.html для авторизації, вона унаслідується від base\_main.html. В коді даної веб-сторінки за допомогою шаблонізатора Jinja використовуються поля форми LoginForm. До даних полів застосовані відповідні стилі та накладена відповідна HTML-розмітка. При натисканні кнопки “Увійти” формується запит на сервер. Якщо дані введені коректно, тобто такий користувач існує, відбувається авторизація користувача та переадресація на сторінку main.html, інакше користувач потрапляє на сторінку login\_invalid.html, де користувачу виводиться повідомлення про помилку авторизації.

Сторінка main.html - головна сторінка веб-сервісу, вона унаслідується від base\_main.html. Користувач після авторизації потрапляє на дану сторінку, де може переглянути статистику по кожній з компонент персонального комп’ютера у вигляді відповідних діаграм та таблиць, за допомогою меню перейти на інші сторінки веб-сервісу.

Діаграми на сторінці main.html формуються за допомогою JavaScript з використанням бібліотеки Chart.js. Реалізація діаграм винесена у відповідний тег <script> для формування відповідного сценарію візуалізації діаграм. З допомогою JavaScript реалізована можливість приховування діаграм - всіх за допомогою відповідної кнопки в головному меню, або окремих. Кожен блок діаграм та таблиць містить відповідні кнопки для згортання та розгортання відповідних фреймів.

В додатку IT\_items представлена більша кількість HTML-сторінок. Логічно їх можна поділити на наступні категорії:

* HTML-сторінки для роботи з формами (додавання, видалення та редагування об’єктів);
* HTML-сторінки для перегляду переліку персональних комп’ютерів;
* HTML-сторінки для перегляду переліку комплектуючих до персональних комп’ютерів;
* HTML-сторінки для перегляду детальної інформації по конкретному персональному комп’ютеру;
* HTML-сторінки для перегляду детальної інформації по конкретній комплектуючій;
* HTML-сторінки для перегляду переліку актів виконаних робіт по конкретному персональному комп’ютеру;
* HTML-сторінки для перегляду вибраного акту виконаних робіт;
* HTML-сторінки для генерації паспорту обладнання.

Інтерфейс для перегляду специфікацій

Розглянемо ключові моменти реалізації front-end на прикладах коду з кожної категорії. В минулих розділах були розглянуті функції відображення. В рамках цього розділу важливо пояснити як дані з back-end передаються в front-end - а саме в HTML-документ. Наведений нижче код робить запит до бази даних та в зміну *items\_pc* зберігає список всіх персональних комп’ютерів.

*def IT\_items(request):  
 items\_pc = PC.objects.all()  
 return render(request, 'IT\_items/IT-items.html', {'items\_pc': items\_pc})*

Далі за допомогою оператора *return* ми поєднуємо шаблон *'IT\_items/IT-items.html'* із заданим контекстним словником *{'items\_pc': items\_pc}* та повертаємо об'єкт типу - HTTP-відповідь. При проектуванні даної логіки розробник повинен приділити увагу контекстному словнику. В мові програмування Python словник представлений у вигляді наступної структури: {'ключ': значення, ... , 'ключ': значення}. При розробці веб-сервісу за допомогою фреймворка Django словник застосовується наступним чином: значення - це та змінна, яку користувач хоче передати до HTML-документу, а ключ - це назва цієї змінної, яку користувач може використовувати при реалізації frond-end. Іншими словами, розробник, вписуючи в поле «значення» зміну *items\_pc*,зможе її використати за тою назвою, яка вписана у поле «ключ».

Нижче наведено фрагмент коду з шаблону *'IT\_items/IT-items.html'*

*{% if items\_pc|length > 0 %}  
{% for el in items\_pc %}  
<div class="alert alert-info">  
 <h2>Тип обладнання: {{ el.name\_for\_user }}</h2>  
 <h2>Інвентарний номер: {{ el.inventory\_number }}</h2>  
 <a href="/IT\_items/{{el.name}}/{{el.id}} " class="btn btn-info">Детальніше</a>  
</div>  
{% endfor %}  
{% else %}  
<div class="alert alert-info">  
 <h3>У нас ще немає персональних комп'ютерів</h3>  
</div>  
{% endif %}*

В даному фрагменті коду описана логіка виведення переліку персональних комп’ютерів. Кожен комп’ютер винесено в окрему стилістичну оболонку - alert - інтерфейс компонент в Bootstrap. Кожен alert містить повідомлення з інформацією про тип обладнання та інвентарний номер. За допомогою шаблонізатора Jinja вноситься додатковий функціонал до HTML-сторінки. Виконується перевірка кількості персональних комп’ютерів за допомогою оператора if. Потім за допомогою циклу for виконується перебір елементів списку *items\_pc*. Тобто, в межах кожної ітерації циклу створюється окремий alert та вписується в нього інформацію про тип обладнання та інвентарний номер конкретного обладнання із загального списку. Також присутня кнопка “Детальніше”, яке перенаправляє користувача на окрему сторінку з детальною інформацією про вибране обладнання.

При натисканні на кнопку “Детальніше” виконується перехід за наступним посиланням */IT\_items/{{el.name}}/{{el.id}}*, де el.name та el.id назва обладнання та його ідентифікаційний номер в базі даних (не плутати з інвентарним номером). Постає ряд логічних питань - навіщо в URL вказувати назву обладнання разом з його ідентифікаційний номер та яким чином спрацьовує перехід на сторінку, де висвітлена інформація саме по цьому персональному комп’ютеру. Для того, щоб відповісти на дані запитання потрібно розглянути повну схему переходу від веб-сторінки зі списком всіх персональних комп’ютерів до сторінки, де описана детальна інформація конкретного обладнання - паспорт персонального комп’ютера.

Кожен окремий додаток містить перелік URL шаблонів, а ці шаблони зв’язані з відповідними функціями відображення. Повертаючись до конкретного прикладу, в файлі urls.py в додатку IT\_items прописано відповідний шлях: path('<str:item\_name>/<int:item\_id>', views.item\_detail, name='item\_detail').

Даний шлях відповідає посиланню */IT\_items/{{el.name}}/{{el.id}}*, тобто при переході на дану сторінку викликається функція представлення item\_detail() з переданими параметрами - назвою обладнання та ідентифікаційним номером

*def item\_detail(request, item\_name, item\_id):  
 item = None  
 try:  
 if item\_name == 'PC':  
 item = PC.objects.get(id=item\_id)  
 except:  
 raise Http404('ERROR item\_detail')  
  
 return render(request, 'IT\_items/IT\_item\_detail.html', {'item': item})*

За допомогою item\_name та item\_id формується запит до бази даних з метою отримання конкретного персонального комп’ютера та збереження цього об’єкту в змінну item. Далі поєднується шаблон *'IT\_items/IT-item\_detail.html'* із заданим контекстним словником *{'item': item}* та повертає об'єкт типу - HTTP-відповідь.

Нижче наведено фрагмент коду з шаблону *IT\_items/IT\_item\_detail.html*

*<div class="container mt-5">  
 <div class="d-flex justify-content-center">  
 <aside class="col-sm-10">  
 <div class="card">  
 <article class="card-body">  
 <div class="alert alert-info">  
 <h2 align="center" class="alert-heading">Паспорт обладнання</h2>  
 <hr>  
 <h2 class="alert-heading">Основне</h2>  
 <hr>  
 <h5 align="justify">Тип обладнання: {{ item.name\_for\_user }}</h5>  
 <h5 align="justify">Інвентарний номер: {{ item.inventory\_number }}</h5>  
 <hr>  
 <h2 class="alert-heading">Місцезнаходження</h2>  
 <hr>  
 <h5>Поверх: {{ item.floor }}</h5>  
 <h5>Кабінет: {{ item.room }}</h5>  
 <h5>Учбове місце: {{ item.place }}</h5>  
 <hr>  
 <h5>Операційна система: {{ item.operating\_system }}</h5>  
 <hr>  
 <h2 class="alert-heading">Компоненти</h2>  
 <hr>*

*<table class="col-sm-12" border="0">  
 <tr>  
 <td>  
 <h3 class="alert-heading">Материнська плата</h3>  
 </td>  
 <td align="right">  
 {% if item.motherboard %}  
 <h5><a href="/IT\_items/pc\_accessories/{{ item.motherboard.name }}/{{ item.motherboard.id }}/update-motherboard" class="badge badge-info">  
 Редагування  
 </a></h5>  
 {% endif %}  
 </td>  
 </tr>  
 </table>  
 <hr>  
 {% if item.motherboard %}  
 <h5>Бренд: {{item.motherboard.brand}}</h5>  
 <h5>Модель: {{item.motherboard.model}}</h5>  
 <h5>Серійний номер: {{item.motherboard.serial\_number}}</h5>  
 <h5>Форм-фактор: {{item.motherboard.form\_factor}}</h5>  
 <h5>Тип слоту для ОЗУ: {{item.motherboard.type\_ram\_slot}}</h5>  
 <br>  
 <h5>Центральный процессор: {{item.motherboard.central\_processing\_unit}}</h5>  
 {% if item.motherboard.integrated\_graphics%}  
 <h5>Інтегрована відеокарта: наявна</h5>  
 {% else %}  
 <h5>Інтегрована відеокарта: відсутня</h5>  
 {% endif %}  
 <br>  
 {% if item.motherboard.integrated\_sound\_card%}  
 <h5>Інтегрована звукова-карта: наявна</h5>  
 {% else %}  
 <h5>Інтегрована звукова-карта: відсутня</h5>  
 {% endif %}  
 {% if item.motherboard.integrated\_lan\_card%}  
 <h5>Інтегрована мережева-карта: наявна</h5>  
 {% else %}  
 <h5>Інтегрована мережева-карта: відсутня</h5>  
 {% endif %}  
 {% else %}  
 <h5>Відсутня</h5>  
 {% endif %}  
 <hr>  
 <table class="col-sm-12" border="0">  
 <tr>  
 <td>  
 <h3 class="alert-heading">Твердотільний накопичувач</h3>  
 </td>  
 <td align="right">  
 {% if item.solid\_state\_drive %}  
 <h5><a href="/IT\_items/pc\_accessories/{{ item.solid\_state\_drive.name }}/{{ item.solid\_state\_drive.id }}/update-solid-state-drive" class="badge badge-info">  
 Редагування  
 </a></h5>  
 {% endif %}  
 </td>  
 </tr>  
 </table>  
 <hr>  
 {% if item.solid\_state\_drive %}  
 <h5>Бренд: {{item.solid\_state\_drive.brand}}</h5>  
 <h5>Модель: {{item.solid\_state\_drive.model}}</h5>  
 <h5>Серійний номер: {{item.solid\_state\_drive.serial\_number}}</h5>  
 <h5>Обсяг пам'яті: {{item.solid\_state\_drive.memory\_size}} GB</h5>  
 {% else %}  
 <h5>Відсутня</h5>  
 {% endif %}  
 <hr>  
 <table class="col-sm-12" border="0">  
 <tr>  
 <td>  
 <h3 class="alert-heading">Накопичувач на жорстких магнітних дисках</h3>  
 </td>  
 <td align="right">  
 {% if item.hard\_disk\_drive %}  
 <h5><a href="/IT\_items/pc\_accessories/{{ item.hard\_disk\_drive.name }}/{{ item.hard\_disk\_drive.id }}/update-hard-disk-drive" class="badge badge-info">  
 Редагування  
 </a></h5>  
 {% endif %}  
 </td>  
 </tr>  
 </table>  
 <hr>  
 {% if item.hard\_disk\_drive %}  
 <h5>Бренд: {{item.hard\_disk\_drive.brand}}</h5>  
 <h5>Модель: {{item.hard\_disk\_drive.model}}</h5>  
 <h5>Серійний номер: {{item.hard\_disk\_drive.serial\_number}}</h5>  
 <h5>Обсяг пам'яті: {{item.hard\_disk\_drive.memory\_size}} GB</h5>  
 {% else %}  
 <h5>Відсутня</h5>  
 {% endif %}  
 <hr>  
 <table class="col-sm-12" border="0">  
 <tr>  
 <td>  
 <h3 class="alert-heading">Блок живлення</h3>  
 </td>  
 <td align="right">  
 {% if item.power\_supply %}  
 <h5><a href="/IT\_items/pc\_accessories/{{ item.power\_supply.name }}/{{ item.power\_supply.id }}/update-power-supply" class="badge badge-info">  
 Редагування  
 </a></h5>  
 {% endif %}  
 </td>  
 </tr>  
 </table>  
 <hr>  
 {% if item.power\_supply %}  
 <h5>Бренд: {{item.power\_supply.brand}}</h5>  
 <h5>Модель: {{item.power\_supply.model}}</h5>  
 <h5>Серійний номер: {{item.power\_supply.serial\_or\_inventory\_number}}</h5>  
 <h5>Максимальна споживана потужність від мережі: {{item.power\_supply.power\_consumption}}W</h5>  
 {% else %}  
 <h5>Відсутній</h5>  
 {% endif %}  
 <hr>  
 <table class="col-sm-12" border="0">  
 <tr>  
 <td>  
 <h3 class="alert-heading">Відеокарта</h3>  
 </td>  
 <td align="right">  
 {% if item.video\_card %}  
 <h5><a href="/IT\_items/pc\_accessories/{{ item.video\_card.name }}/{{ item.video\_card.id }}/update-video-card" class="badge badge-info">  
 Редагування  
 </a></h5>  
 {% endif %}  
 </td>  
 </tr>  
 </table>  
 <hr>  
 {% if item.video\_card %}  
 <h5>Бренд: {{item.video\_card.brand}}</h5>  
 <h5>Модель: {{item.video\_card.model}}</h5>  
 <h5>Серійний номер: {{item.video\_card.serial\_number}}</h5>  
 <h5>Обсяг пам'яті: {{item.video\_card.memory\_size}} MB</h5>  
 {% else %}  
 <h5>Відсутній</h5>  
 {% endif %}  
 <hr>  
 <table class="col-sm-12" border="0">  
 <tr>  
 <td>  
 <h3 class="alert-heading">Мережева плата</h3>  
 </td>  
 <td align="right">  
 {% if item.lan\_card %}  
 <h5><a href="/IT\_items/pc\_accessories/{{ item.lan\_card.name }}/{{ item.lan\_card.id }}/update-lan-card" class="badge badge-info badge">  
 Редагування  
 </a></h5>  
 {% endif %}  
 </td>  
 </tr>  
 </table>  
 <hr>  
 {% if item.lan\_card %}  
 <h5>Бренд: {{item.lan\_card.brand}}</h5>  
 <h5>Модель: {{item.lan\_card.model}}</h5>  
 <h5>Серійний номер: {{item.lan\_card.serial\_number}}</h5>  
 {% else %}  
 <h5>Відсутній</h5>  
 {% endif %}  
 <hr>  
 <table class="col-sm-12" border="0">  
 <tr>  
 <td>  
 <h3 class="alert-heading">Звукова плата</h3>  
 </td>  
 <td align="right">  
 {% if item.sound\_card %}  
 <h5><a href="/IT\_items/pc\_accessories/{{ item.sound\_card.name }}/{{ item.sound\_card.id }}/update-sound-card" class="badge badge-info badge">  
 Редагування  
 </a></h5>  
 {% endif %}  
 </td>  
 </tr>  
 </table>  
 <hr>  
 {% if item.sound\_card %}  
 <h5>Бренд: {{item.sound\_card.brand}}</h5>  
 <h5>Модель: {{item.sound\_card.model}}</h5>  
 <h5>Серійний номер: {{item.sound\_card.serial\_number}}</h5>  
 {% else %}  
 <h5>Відсутній</h5>  
 {% endif %}  
 <hr>  
 <table class="col-sm-12" border="0">  
 <tr>  
 <td>  
 <h3 class="alert-heading">Оптичний накопичувач</h3>  
 </td>  
 <td align="right">  
 {% if item.optical\_drive %}  
 <h5><a href="/IT\_items/pc\_accessories/{{ item.optical\_drive.name }}/{{ item.optical\_drive.id }}/update-optical-drive" class="badge badge-info badge">  
 Редагування  
 </a></h5>  
 {% endif %}  
 </td>  
 </tr>  
 </table>  
 <hr>  
 {% if item.optical\_drive %}  
 <h5>Бренд: {{item.optical\_drive.brand}}</h5>  
 <h5>Модель: {{item.optical\_drive.model}}</h5>  
 <h5>Серійний номер: {{item.optical\_drive.serial\_number}}</h5>  
 <h5>Тип приводу: {{item.optical\_drive.type\_drive}}</h5>  
 <h5>Тип роз'єму: {{item.optical\_drive.type\_connector}}</h5>  
 {% else %}  
 <h5>Відсутній</h5>  
 {% endif %}  
 <hr>  
 <table class="col-sm-12" border="0">  
 <tr>  
 <td width="40%">  
 <h3 class="alert-heading">Акти проведених робіт</h3>  
 </td>  
 <td width="28%" align="center">  
 <h4><a href="/IT\_items/{{ item.name }}/{{ item.id }}/add\_work\_report" class="badge badge-info badge">  
 Створити новий акт  
 </a></h4>  
 </td>  
 <td width="4%"></td>  
 <td width="28%" align="center">  
 <h4><a href="/IT\_items/{{ item.name }}/{{ item.id }}/work\_reports" class="badge badge-info badge">  
 Переглянути всі акти  
 </a></h4>  
 </td>  
 </tr>  
 </table>   
 <hr>  
 <table width="100%" align="center" border="0">  
 <tr>  
 <td width="5%"></td>  
 <td width="15%">  
 <a href="/IT\_items/{{ item.name }}/{{ item.id }}/del" class="btn btn-info btn-outline-light">Видалити</a>  
 </td>  
 <td width="30%">  
 {% if item.name == 'PC' %}  
 <a href="/IT\_items/{{ item.name }}/{{ item.id }}/update-pc" class="btn btn-info btn-outline-light">Редагування</a>  
 {% endif %}  
 </td>  
 <td width="45%" align="right">  
 <a href="/IT\_items/{{ item.name }}/{{ item.id }}/pdf" class="btn btn-info btn-outline-light">Створити PDF</a>  
 </td>  
 <td width=5%></td>  
 </tr>  
 </table>  
 </div>  
 </article>  
 </div>  
 </aside>  
 </div>  
</div>*

В рамках цієї частини коду на веб-сторінці генерується паспорт персонального комп’ютера. За допомогою шаблонізатора Jinja є можливість виводу інформації по даному комп’ютеру, яка зберігається в змінній *item*. Першочергово виводиться основна інформація - тип обладнання, інвентарний номер та місце знаходження, а саме: поверх, номер кабінету, номер робочого місця. Також вказується тип операційної системи за допомогою меню, що випадає. Далі йде вивід інформації по кожній з компонент персонального комп’ютера. Логіка відображення компонент є однаковим незалежно від їх виду. Розглянемо вивід інформації по материнській платі. Спочатку створюється заголовок з назвою компоненти разом з кнопкою редагування. Варто зазначити, що дана кнопка доступна у разі наявності даної компоненти у складі обраного персонального комп’ютера та дана кнопка є посиланням на форму для редагування конкретної компоненти­­­. Далі за допомогою шаблонізатора Jinja та оператора if перевіряється чи має даний персональний комп’ютер материнську плату. Якщо дана компонента присутня виводимо інформацію по ній, відповідні поля описані в моделях. Інакше виводимо повідомлення, що дана компонента відсутня. Кожна з компонент персонального комп’ютера стилістично відокремлена горизонтальними лініями.

Далі після виводу інформації по всім компонентам йде розділ з актами виконаних робіт. В межах даного розділу представлено дві функціональні кнопки - створити новий акт та перегляд вже створених актів. Логіка перегляду актів аналогічна з переглядом обладнання. А створення актів реалізовано, як створення нового об’єкта та додавання його до бази даних - механізм даного процесу буде розглянуто далі. В кінці веб-сторінки з паспортом обладнання наявні три функціональні кнопки. Перша- це видалення даного об’єкту - реалізована відповідна функція відображення, яка, використовуючи назву обладнання та його ID, робить запит до бази даних на видалення. Друга кнопка - це перехід на форму редагування (буде розглянуто далі). Остання формує паспорт обладнання у форматі pdf, використовуючи HTML-розмітку.

Інтерфейс для створення об’єктів

Для додавання нового обладнання та нових комплектуючих до них було реалізовано ряд HTML-шаблонів. Оскільки для кожного типу об’єкту була реалізована своя форма для додавання на стороні back-end, то потрібно реалізувати таку саму кількість HTML-шаблонів. Розглянемо процес додавання материнської плати.

У файлі urls.py реалізовано наступну відповідність path('add-item/add-motherboard', views.add\_motherboard, name='add\_motherboard'). Тобто при переході на веб-сторінку за шляхом add-item/add-motherboard застосовується HTML-шаблон add\_motherboard.html. Розглянемо частину коду з даного шаблона, а саме логіку описану в тезі form.

*<form method="post">  
 {% csrf\_token %}  
 <div class="form-group">  
 <table class="col-sm-12" align="center" border="0">  
 <tr>  
 <td colspan="3">  
 <hr>  
 </td>  
 </tr>  
 <tr>  
 <td colspan="3" align="center">  
 <!--Name of field and input-->  
 {{ form.motherboard\_serial\_number.label\_tag }}  
 <div class="mx-auto" style="width: 50%">  
 {{ form.motherboard\_serial\_number }}  
 </div>  
 <!--Show help text for field-->  
 {% if form.motherboard\_serial\_number.help\_text %}  
 <small style="color: grey">{{ form.motherboard\_serial\_number.help\_text }}</small>  
 {% endif %}  
 <!--Show error text for field-->  
 {% for error in form.motherboard\_serial\_number.errors %}  
 <p style="color: red">{{ error }}</p>  
 {% endfor %}  
 </td>  
 </tr>  
 <tr>  
 <td colspan="3">  
 <hr>  
 </td>  
 </tr>  
 <tr>  
 <td colspan="3" align="center">  
 <!--Name of field and input-->  
 {{ form.motherboard\_brand.label\_tag }}*

*<div class="mx-auto" style="width: 50%">  
 {{ form.motherboard\_brand }}  
 </div>  
 <!--Show help text for field-->  
 {% if form.motherboard\_brand.help\_text %}  
 <small style="color: grey">{{ form.motherboard\_brand.help\_text }}</small>  
 {% endif %}  
 <!--Show error text for field-->  
 {% for error in form.motherboard\_brand.errors %}  
 <p style="color: red">{{ error }}</p>  
 {% endfor %}  
 </td>  
 </tr>  
 <tr>  
 <td colspan="3">  
 <hr>  
 </td>  
 </tr>  
 <tr>  
 <td colspan="3" align="center">  
 <!--Name of field and input-->  
 {{ form.motherboard\_model.label\_tag }}  
 <div class="mx-auto" style="width: 50%">  
 {{ form.motherboard\_model }}  
 </div>  
 <!--Show help text for field-->  
 {% if form.motherboard\_model.help\_text %}  
 <small style="color: grey">{{ form.motherboard\_model.help\_text }}</small>  
 {% endif %}  
 <!--Show error text for field-->  
 {% for error in form.motherboard\_model.errors %}  
 <p style="color: red">{{ error }}</p>  
 {% endfor %}  
 </td>  
 </tr>  
 <tr>  
 <td colspan="3">  
 <hr>  
 </td>  
 </tr>  
 <tr>  
 <td colspan="3" align="center">  
 <!--Name of field and input-->  
 {{ form.motherboard\_form\_factor.label\_tag }}  
 <div class="mx-auto" style="width: 50%">  
 {{ form.motherboard\_form\_factor }}  
 </div>  
 <!--Show help text for field-->  
 {% if form.motherboard\_form\_factor.help\_text %}  
 <small style="color: grey">{{ form.motherboard\_form\_factor.help\_text }}</small>  
 {% endif %}  
 <!--Show error text for field-->  
 {% for error in form.motherboard\_form\_factor.errors %}  
 <p style="color: red">{{ error }}</p>  
 {% endfor %}  
 </td>  
 </tr>  
 <tr>  
 <td colspan="3">  
 <hr>  
 </td>  
 </tr>  
 <tr>  
 <td colspan="2" align="center">  
 <!--Name of field and input-->  
 {{ form.motherboard\_central\_processing\_unit.label\_tag }}  
 <div class="mx-auto" style="width: 85%">  
 {{ form.motherboard\_central\_processing\_unit }}  
 </div>  
 <!--Show help text for field-->  
 {% if form.motherboard\_central\_processing\_unit.help\_text %}  
 <small style="color: grey">{{ form.motherboard\_central\_processing\_unit.help\_text }}</small>  
 {% endif %}  
 <!--Show error text for field-->  
 {% for error in form.motherboard\_central\_processing\_unit.errors %}  
 <p style="color: red">{{ error }}</p>  
 {% endfor %}  
 </td>  
<!-- <td>2</td>-->  
 <td align="center">  
 <!--Name of field and input-->  
 {{ form.motherboard\_integrated\_graphics.label\_tag }}  
 {{ form.motherboard\_integrated\_graphics }}  
 <br>  
 <!--Show help text for field-->  
 {% if form.motherboard\_integrated\_graphics.help\_text %}  
 <small style="color: grey">{{ form.motherboard\_integrated\_graphics.help\_text }}</small>  
 {% endif %}  
 <!--Show error text for field-->  
 {% for error in form.motherboard\_integrated\_graphics.errors %}  
 <p style="color: red">{{ error }}</p>  
 {% endfor %}  
 </td>  
 </tr>  
 <tr>  
 <td colspan="3">  
 <hr>  
 </td>  
 </tr>  
 <tr>  
 <td align="center">  
 <!--Name of field and input-->  
 {{ form.motherboard\_integrated\_sound\_card.label\_tag }}  
 {{ form.motherboard\_integrated\_sound\_card }}  
 <br>  
 <!--Show help text for field-->  
 {% if form.motherboard\_integrated\_sound\_card.help\_text %}  
 <small style="color: grey">{{ form.motherboard\_integrated\_sound\_card.help\_text }}</small>  
 {% endif %}  
 <!--Show error text for field-->  
 {% for error in form.motherboard\_integrated\_sound\_card.errors %}  
 <p style="color: red">{{ error }}</p>  
 {% endfor %}  
 </td>  
 <td align="center" width="15%">  
 </td>  
 <td align="center">  
 <!--Name of field and input-->  
 {{ form.motherboard\_integrated\_lan\_card.label\_tag }}  
 {{ form.motherboard\_integrated\_lan\_card }}  
 <br>  
 <!--Show help text for field-->  
 {% if form.motherboard\_integrated\_lan\_card.help\_text %}  
 <small style="color: grey">{{ form.motherboard\_integrated\_lan\_card.help\_text }}</small>  
 {% endif %}  
 <!--Show error text for field-->  
 {% for error in form.motherboard\_integrated\_lan\_card.errors %}  
 <p style="color: red">{{ error }}</p>  
 {% endfor %}  
 </td>  
 </tr>  
 <tr>  
 <td colspan="3">  
 <hr>  
 </td>  
 </tr>  
 <tr>  
 <td colspan="3" align="center">  
 <!--Name of field and input-->  
 {{ form.motherboard\_type\_ram\_slot.label\_tag }}  
 <div class="mx-auto" style="width: 50%">  
 {{ form.motherboard\_type\_ram\_slot }}  
 </div>  
 <!--Show help text for field-->  
 {% if form.motherboard\_type\_ram\_slot.help\_text %}  
 <small style="color: grey">{{ form.motherboard\_type\_ram\_slot.help\_text }}</small>  
 {% endif %}  
 <!--Show error text for field-->  
 {% for error in form.motherboard\_type\_ram\_slot.errors %}  
 <p style="color: red">{{ error }}</p>  
 {% endfor %}  
 </td>  
 </tr>  
 <tr>  
 <td colspan="3">  
 <hr>  
 </td>  
 </tr>  
 </table>  
 </div>  
 <input type="submit" class="btn btn-outline-info" value="Створити">  
 </form>*

В даному випадку форма використовується для обміну даними між користувачем та сервером. В HTML-документі за допомогою шаблонізатора описані комірки, в котрих може з’явитися текст помилки при її знаходженні фреймворком. Кожне поле відповідної форми має підполя errors, help\_text, label\_tag. Відповідні підполя описуються на стороні back-end при реалізації форм. Отже, при натисканні кнопки “Створити”, після перевірки валідності введених даних, відповідні дані передаються до сервера та виконується наступна логіка відповідної функції представлення:

*if request.method == 'POST':  
 form = AddMotherboardForm(request.POST)  
 if form.is\_valid():  
 cd = form.cleaned\_data  
 motherboard = Motherboard(serial\_number=cd['motherboard\_serial\_number'],  
 brand=cd['motherboard\_brand'],  
 model=cd['motherboard\_model'], central\_processing\_unit=cd['motherboard\_central\_processing\_unit'], integrated\_graphics=cd['motherboard\_integrated\_graphics'],  
integrated\_sound\_card=cd['motherboard\_integrated\_sound\_card'], integrated\_lan\_card=cd['motherboard\_integrated\_lan\_card'],  
 form\_factor=cd['motherboard\_form\_factor'], type\_ram\_slot=cd['motherboard\_type\_ram\_slot'])  
  
 try:  
 motherboard.save()  
 except:  
 return render(request, 'IT\_items/error.html', {'serial\_number': motherboard.serial\_number,  
 'name\_of\_item': motherboard.name\_for\_user})  
 return HttpResponseRedirect(reverse('IT\_items:pc\_accessories'))  
else:  
 form = AddMotherboardForm()  
 pass  
  
return render(request, 'IT\_items/add\_motherboard.html', {'form': form})*

В змінну *form* зберігаємо значення полів форми *AddMotherboardForm*, які користувач заповнив у веб-браузері. Далі створюємо об’єкт типу Motherboard, в конструктор якого передаються параметри, які були збережені в змінну *form*. Потім робимо запит до бази даних для збереження нового об’єкта. Якщо на даному етапі виникає помилка, то виконується переадресація на сторінку з повідомленням про помилку. Інакше виконується переспрямування на HTML-шаблон pc\_accessories.html, де представлено перелік всіх компонент.

Інтерфейс для редагування об’єктів

Розглянемо процес редагування материнської плати. У файлі urls.py реалізовано наступну відповідність path('pc\_accessories/<str:item\_name> /<int:item\_id>/update-motherboard',views.motherboard\_update, name='mother board\_update'). Тобто, при переході на веб-сторінку за наведеним шляхом відбувається виклик наступної функції відображення:

*def motherboard\_update(request, item\_name, item\_id):  
 item = None  
 form\_factors = None  
 type\_ram\_slots = None  
 central\_processing\_units = None  
 if item\_name == 'Motherboard':  
 item = Motherboard.objects.get(id=item\_id)  
 form\_factors = [el[0] for el in MOTHERBOARD\_FROM\_FACTORS]  
 type\_ram\_slots = [el[0] for el in TYPE\_RAM\_SLOTS]  
 central\_processing\_units = [el[0] for el in TYPE\_CENTRAL\_PROCESSING\_UNIT]  
 if request.method == 'POST':  
 if item\_name == 'Motherboard':  
 item.brand = request.POST['motherboard\_brand']  
 item.model = request.POST['motherboard\_model']  
 item.serial\_number = request.POST['motherboard\_serial\_number'  
 item.central\_processing\_unit = request.POST.get('motherboard\_central\_processing\_unit', None)  
 item.form\_factor = request.POST.get('motherboard\_form\_factor', None)  
 item.type\_ram\_slot = request.POST.get('motherboard\_type\_ram\_slot', None)  
 item.integrated\_graphics = True if request.POST.get('motherboard\_integrated\_graphics') else False  
 item.integrated\_sound\_card = True if request.POST.get('motherboard\_integrated\_sound\_card') else False  
 item.integrated\_lan\_card = True if request.POST.get('motherboard\_integrated\_lan\_card') else False  
 try:  
 item.save()  
 return HttpResponseRedirect(reverse('IT\_items:pc\_accessories'))  
 except ObjectDoesNotExist:  
 return 'При оновлені обладнання трапилася помилка'  
 else:  
 return render(request, 'IT\_items/motherboard\_update.html', {'item': item,  
 'form\_factors': form\_factors,  
 'type\_ram\_slots': type\_ram\_slots,  
 'central\_processing\_units': central\_processing\_units})*

Головна відмінність між створенням об’єктів та їх редагуванням полягає в тому, що для створення - використовуються Django форми, а для редагування застосовано механізм взаємодії з HTML-формами на пряму. Кожне поле форми має унікальне ім’я, за яким відбувається до них звернення у функціях представлення за допомогою метода POST.

Перехід на сторінку редагування відбувається для конкретного об’єкта, тобто в URL явно вказується назва обладнання та його ідентифікаційний номер. Ці два значення передаються в функцію представлення, як параметри. За допомогою них робиться запит до бази даних з метою пошуку конкретного об’єкту. Шуканий об’єкт зберігається у змінну *item*. Окрім цієї змінної потрібно створити ще три списка - *form\_factors*, *type\_ram\_slots* та *central\_processing\_units*. Причина створення додаткових змінних полягає у наступному. Через відсутність Django форм, в котрих ми могли описати перелік значень для даних полів, ми повинні їх сформувати в функції представлення та передати разом з об’єктом *item* до HTML-шаблону через контекстний словник.

Розглянемо частину коду з шаблона motherboard\_update.html, а саме логіку описану в тезі form:

*<form method="post">  
 {% csrf\_token %}  
 <div class="">  
 <div class="form-group">  
 <table class="col-sm-12" align="center" border="0">  
 <tr>  
 <td colspan="2">  
 <hr>  
 </td>  
 </tr>  
 <tr>  
 <td colspan="2">  
 <p>Тип обладнання:</p>  
 <div class="mx-auto" style="width: 95%">  
 <input type="text" disabled name="name" id="name" class="form-control" value="{{ item.name\_for\_user }}" placeholder="Введіть тип обладнанням">  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td colspan="2">  
 <hr>  
 </td>  
 </tr>  
 <tr>  
 <td width="50%">  
 <p>Модель:</p>  
 <div class="mx-auto" style="width: 90%">  
 <input type="text" name="motherboard\_model" id="motherboard\_model" class="form-control" value="{{ item.model }}" placeholder="Введіть модель материнської плати">  
 </div>  
 </td>  
 <td>  
 <p>Бренд:</p>  
 <div class="mx-auto" style="width: 90%">  
 <input type="text" name="motherboard\_brand" id="motherboard\_brand" class="form-control" value="{{ item.brand }}" placeholder="Введіть тип обладнанням">  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td colspan="2">  
 <hr>  
 </td>  
 </tr>  
 <tr>  
 <td colspan="2">  
 <p>Серійний номер:</p>  
 <div class="mx-auto" style="width: 95%">  
 <input type="text" name="motherboard\_serial\_number" id="motherboard\_serial\_number" class="form-control" value="{{ item.serial\_number }}" placeholder="Введіть серійний номер материнської плати">  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td colspan="2">  
 <hr>  
 </td>  
 </tr>  
 <tr>  
 <td>  
 <p>Форм-фактор:</p>  
 <div class="mx-auto" style="width: 90%">  
 <select class="form-control" name="motherboard\_form\_factor" id="motherboard\_form\_factor">  
 {% for form\_factor in form\_factors %}  
 <option {% if form\_factor == item.form\_factor %} selected {% endif %}  
 value="{{ form\_factor }}">  
 {{ form\_factor }}  
 </option>  
 {% endfor %}  
 </select>  
 </div>  
 </td>  
 <td>  
 <p>Тип слоту для ОЗУ:</p>  
 <div class="mx-auto" style="width: 90%">  
 <select class="form-control" name="motherboard\_type\_ram\_slot" id="motherboard\_type\_ram\_slot">  
 {% for type\_ram\_slot in type\_ram\_slots %}  
 <option {% if type\_ram\_slot == item.type\_ram\_slot %} selected {% endif %}  
 value="{{ type\_ram\_slot }}">  
 {{ type\_ram\_slot }}  
 </option>  
 {% endfor %}  
 </select>  
 </div>  
 </td>  
 </tr>  
 <tr>  
 <td colspan="2">  
 <hr>  
 </td>  
 </tr>  
 <tr>  
 <td>  
 <p>Центральный процессор: </p>  
 <div class="mx-auto" style="width: 90%">  
 <select class="form-control" name="motherboard\_central\_processing\_unit" id="motherboard\_central\_processing\_unit">  
 {% for central\_processing\_unit in central\_processing\_units %}  
 <option {% if central\_processing\_unit == item.central\_processing\_unit %} selected {% endif %}  
 value="{{ central\_processing\_unit }}">  
 {{ central\_processing\_unit }}  
 </option>  
 {% endfor %}  
 </select>  
 </div>  
 </td>  
 <td align="center">  
 <label>  
 Інтегрована відеокарта:  
 <input type="checkbox" name=motherboard\_integrated\_graphics id="motherboard\_integrated\_graphics" {% if item.integrated\_graphics %} checked="checked" {% endif %}>  
 </label>  
 </td>  
 </tr>  
 <tr>  
 <td colspan="2">  
 <hr>  
 </td>  
 </tr>  
 <tr>  
 <td align="center">  
 <label>  
 Інтегрована звукова-карта:  
 &nbsp;&nbsp;&nbsp;  
 <input type="checkbox" name="motherboard\_integrated\_sound\_card" id="motherboard\_integrated\_sound\_card" {% if item.integrated\_sound\_card %} checked="checked" {% endif %}>  
 </label>  
 </td>  
 <td align="center">  
 <label>  
 Інтегрована мережева-карта:  
 <input type="checkbox" name="motherboard\_integrated\_lan\_card" id="motherboard\_integrated\_lan\_card" {% if item.integrated\_lan\_card %} checked="checked" {% endif %}>  
 </label>  
 </td>  
 </tr>  
 <tr>  
 <td colspan="2">  
 <hr>  
 </td>  
 </tr>  
 </table>  
 </div>  
 </div>  
 <input type="submit" class="btn btn-outline-info" value="Оновити">  
</form>*

Після заповнення форми та натискання кнопки “Оновити” спрацьовує метод POST та виконуються відповідні дії в функції представлення. А саме, відбувається зміна значень полів об’єкта *item* та подальший запит до бази даних на збереження оновленої інформації. При виявленні помилки з’явиться відповідне повідомлення.

Інтерфейс для перегляду та створення актів виконаних робіт

В рамках створеного веб-сервісу було реалізовано логіку, за якою можна переглянути перелік створених актів лише обравши конкретний персональний комп’ютер. В файлі urls.py реалізовано наступний шлях з функцією представлення:

*path('<str:item\_name>/<int:item\_id>/work\_reports', views.work\_reports, name='work\_reports')*

*def work\_reports(request, item\_name, item\_id):  
 item = PC.objects.get(id=item\_id)  
 reports = WorkReport.objects.filter(inventory\_number\_pc*

*=item.inventory\_number)  
 return render(request, 'IT\_items/work\_reports.html', {'reports': reports,  
 'item': item})*

При перегляді паспорту обладнання персонального комп’ютера користувач має кнопки для перегляду актів та створення нового. При натисканні на «перегляд актів» відбувається перехід за посиланням *<str:item\_name>/<int:item\_id>/work\_reports* та викликається метод *work\_reports,* в який передається інформація про тип обладнання та ідентифікаційний номер. В рамках даної функції представлення відбувається пошук потрібного персонального комп’ютера, а саме його інвентарного номера. Далі виконується запит до бази даних для пошуку всіх актів виконаних робіт, в котрих вказано інвентарний номер обраного персонального комп’ютера. Потім поєднується шаблон *'IT\_items/ work\_reports.html'* із заданим контекстним словником *, {'reports': reports, 'item': item}}*. На даній HTML-сторінці представлено перелік актів, кожен акт підписаний датою та часом створення. При необхідності користувач може відкрити будь-який акт та переглянути його. Разом зі списком актів передається й об’єкт - персональний комп’ютер. Це робиться з метою створення заголовку з інвентарним номером та назвою обладнання для зручності користувача та створення посилання на створення нового акту для даного ІТ-обладнання. Сирцевий код даного шаблона наведено нижче:

*<table class="col-sm-12" border="0">  
 <tr>  
 <td>  
 <h1>Акти проведених робіт</h1>  
 </td>  
 <td align="right">  
 <h1><a href="/IT\_items/{{ item.name }}/{{ item.id }}/add\_work\_report" class="badge badge-info">+</a></h1>  
 </td>  
 </tr>  
</table>  
<br>  
  
{% if reports|length > 0 %}  
{% for el in reports %}  
<div class="alert alert-info">  
 <h3>Час створення: {{ el.created\_date }}</h3>  
 <a href="/IT\_items/{{item.name}}/{{item.id}}/work\_reports/{{el.inventory\_number\_pc}}/{{el.id}}" class="btn btn-info">Детальніше</a>  
</div>  
{% endfor %}  
{% else %}  
<div class="alert alert-info">  
 <h3>У нас ще немає актів проведених робіт</h3>  
</div>  
{% endif %}*

В даній частині коду за допомогою шаблонізатора Jinja та циклу for реалізована генерація актів виконаних робіт у вигляді повідомлень (алертів). В кожному алерті присутня дата створення відповідного акту та посилання - перехід на веб-сторінку з детальною інформацією. При натисканні на кнопку "Детальніше" відбувається перехід за наступним посиланням:

path('<str:item\_name>/<int:item\_id>/work\_reports/<str:inventory\_number\_pc>/<int:report\_id>', views.work\_report\_detail, name='work\_report\_detail')

При відображені акту виконаних робіт застосовуємо логіку, яка була реалізована при відображенні паспорта обладнання, але з невеликим доповненням. В даному випадку ми передаємо дві пари параметрів - назва обладнання з його ідентифікаційним номером та інвентарний номер обладнання з ідентифікаційним номером акту виконаних робіт. Причина використання чотирьох змінних зрозуміла при розгляді відповідної функції відображення:

*item = PC.objects.get(id=item\_id)  
report = WorkReport.objects.get(inventory\_number\_pc=inventory\_number\_pc,*

*id=report\_id)  
return render(request, 'IT\_items/work\_report\_detail.html', {'report': report, 'item': item})*

Для пошуку відповідного акту виконаних робіт потрібно виконати два запити до бази даних - пошук відповідного обладнання та пошук відповідного акту за його ідентифікаційним номер та інвентарним номером самого обладнання.

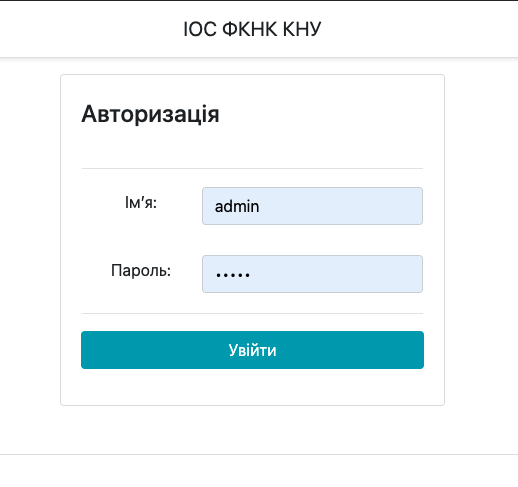
Результат роботи метода - поєднати шаблон *IT\_items/work\_report\_detail. html* з контекстним словником, в якому передаємо 2 об’єкта - акт виконаних робіт та саме обладнання. Передаємо обладнання з метою створення на HTML-сторінці кнопок-посилань на видалення акту та генерації pdf-документу. Код шаблона наведено нижче:

*<h2 align="center" class="alert-heading">Акт проведеної роботи</h2>  
<hr>  
<h5>Час створення: {{ report.created\_date }}</h5>  
<hr>  
<table width="95%" align="center" border="0">  
 <tr>  
 <td>  
 <p align="justify">{{ report.work\_report\_field }}</p>  
 </td>  
 </tr>  
</table>  
<hr>  
<table width="100%" align="center" border="0">  
 <tr>  
 <td width="5%"></td>  
 <td width="45%" align="center">  
 <a href="/IT\_items/{{item.name}}/{{item.id}}/work\_reports/{{report.inventory\_number\_pc}}/{{report.id}}/del" class="btn btn-info btn-outline-light">Видалити</a>  
 </td>  
 <td width="45%" align="center">  
 <a href="/IT\_items/{{item.name}}/{{item.id}}/work\_reports/ {{report.inventory\_number\_pc}}/{{report.id}}/pdf" class="btn btn-info btn-outline-light">Створити PDF</a>  
 </td>  
 <td width="5%"></td>  
 </tr>  
</table>*

ІНТЕРФЕЙС КОРИСТУВАЧА

Вікно авторизації

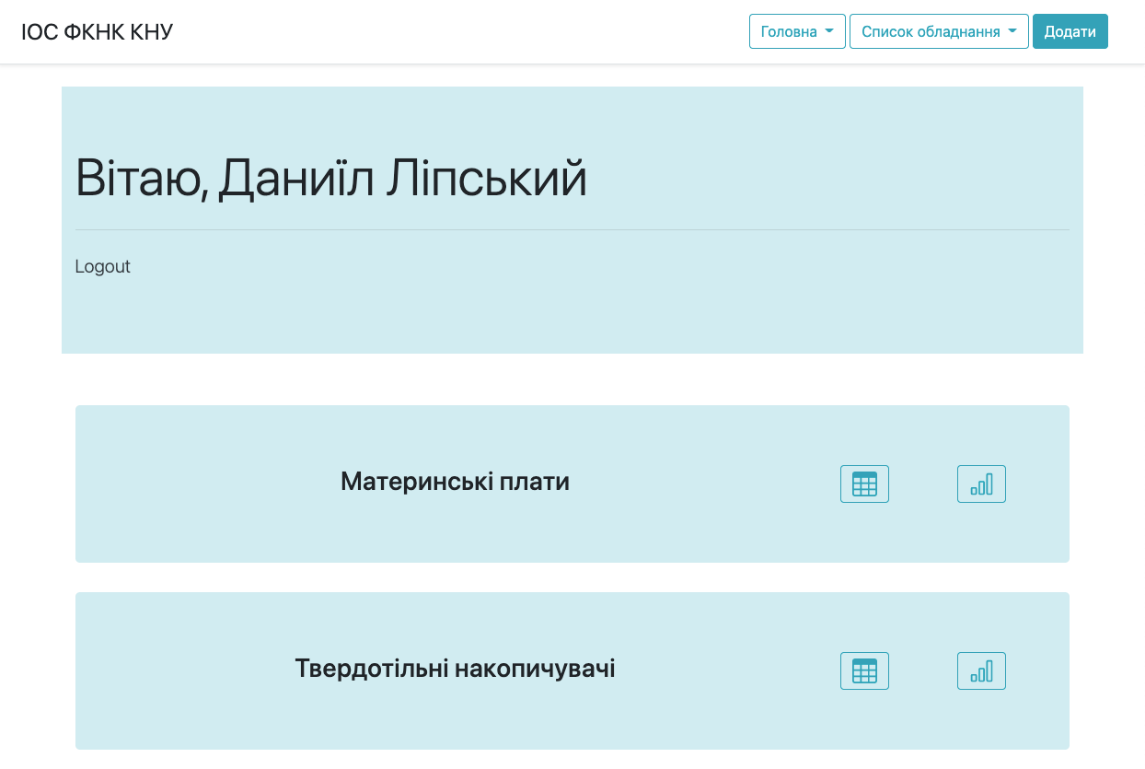
Користувачі даного веб-сервісу поділені на групи - кожна група має свої функціональні можливості. З метою підтвердження прав на певний функціонал в рамках даного додатку була розроблена авторизація в веб- сервісі.



Ілюстрація 1

Головне вікно програми

Відправна точка веб-додатку : в даному вікні представлена статистика по кожній з компонент персонального комп’ютера та таблиці з переліком всіх компонент.

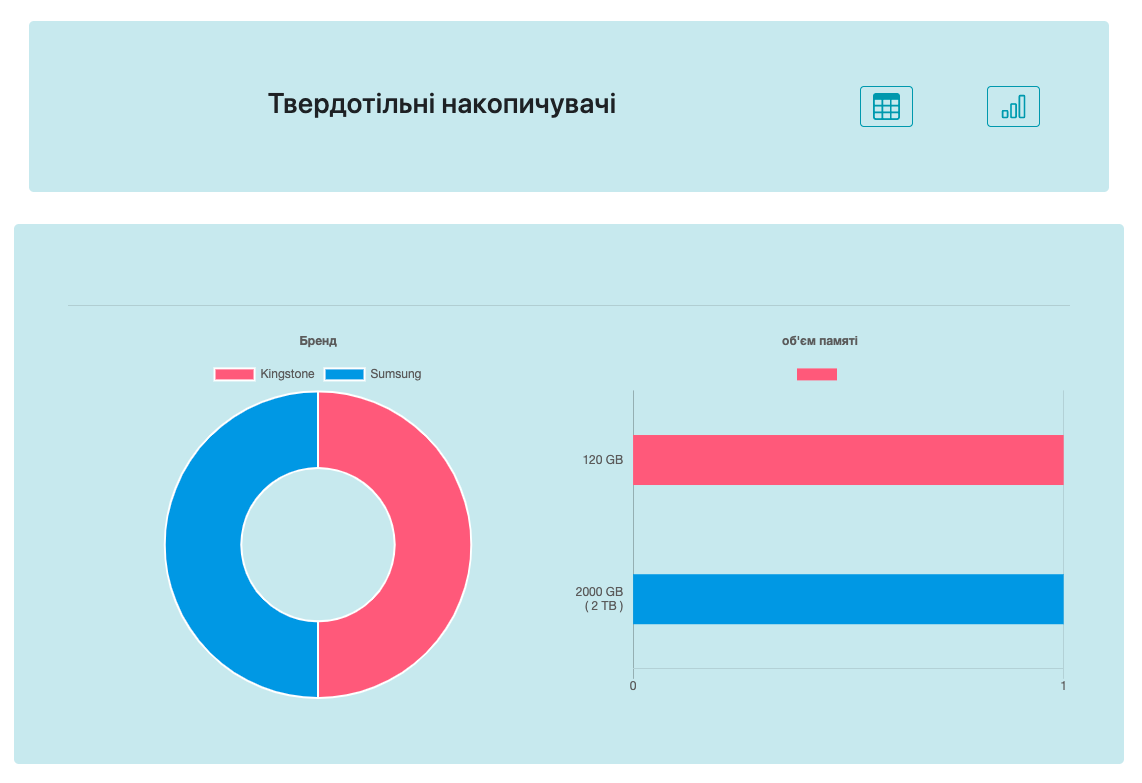


Ілюстрація 2

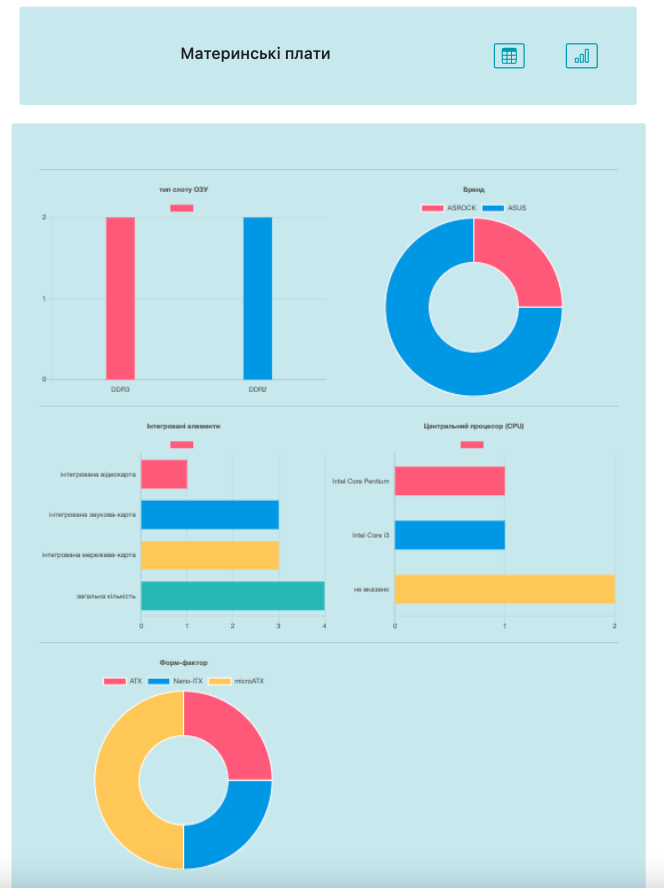
Изображение выглядит как стол

Автоматически созданное описание

Ілюстрація



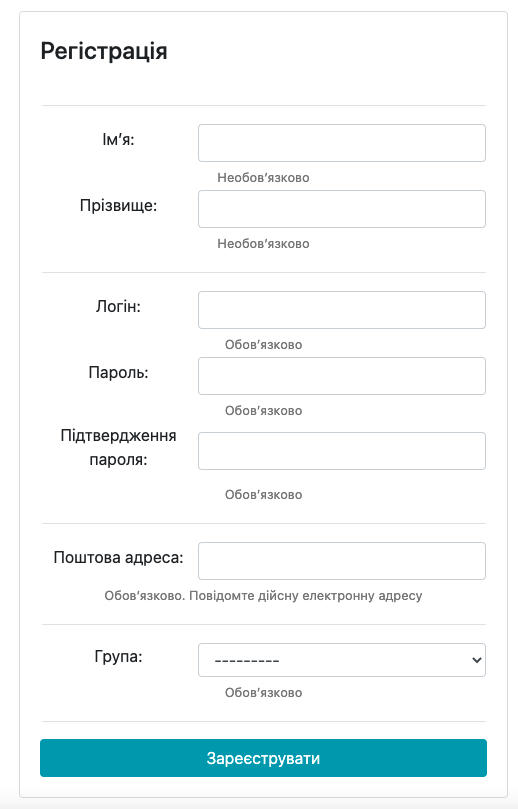
Ілюстрація



Ілюстрація

Вікно додавання нового користувача

Користувач з правами адміністратора має доступ до форми для створення нового користувача. Форма містить поля для введення персональних даних, задання логіну та пароля. Також адміністратор вибирає обмеження у праві доступу до певного функціоналу веб-сервісу користувачем.



Ілюстрація 6

Вікно вибору обладнання та комплектуючих для додавання

Вікно містить типи обладнання, які користувач може додати до бази даних. Структурно перелік обладнання та комплектуючих винесені в окремі фрейми для зручності користувача.

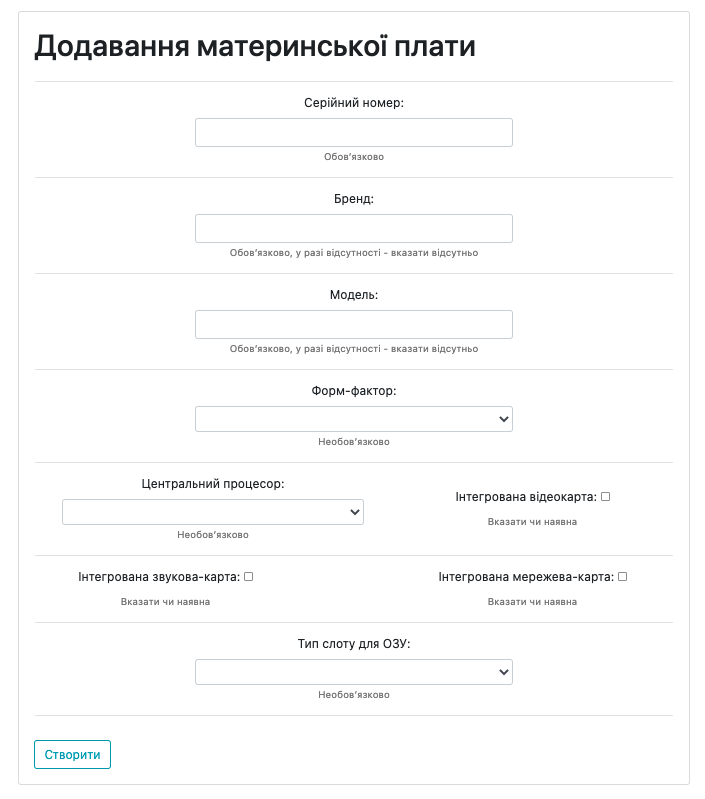
Изображение выглядит как текст

Автоматически созданное описание

Ілюстрація 7

Вікно додавання обладнання

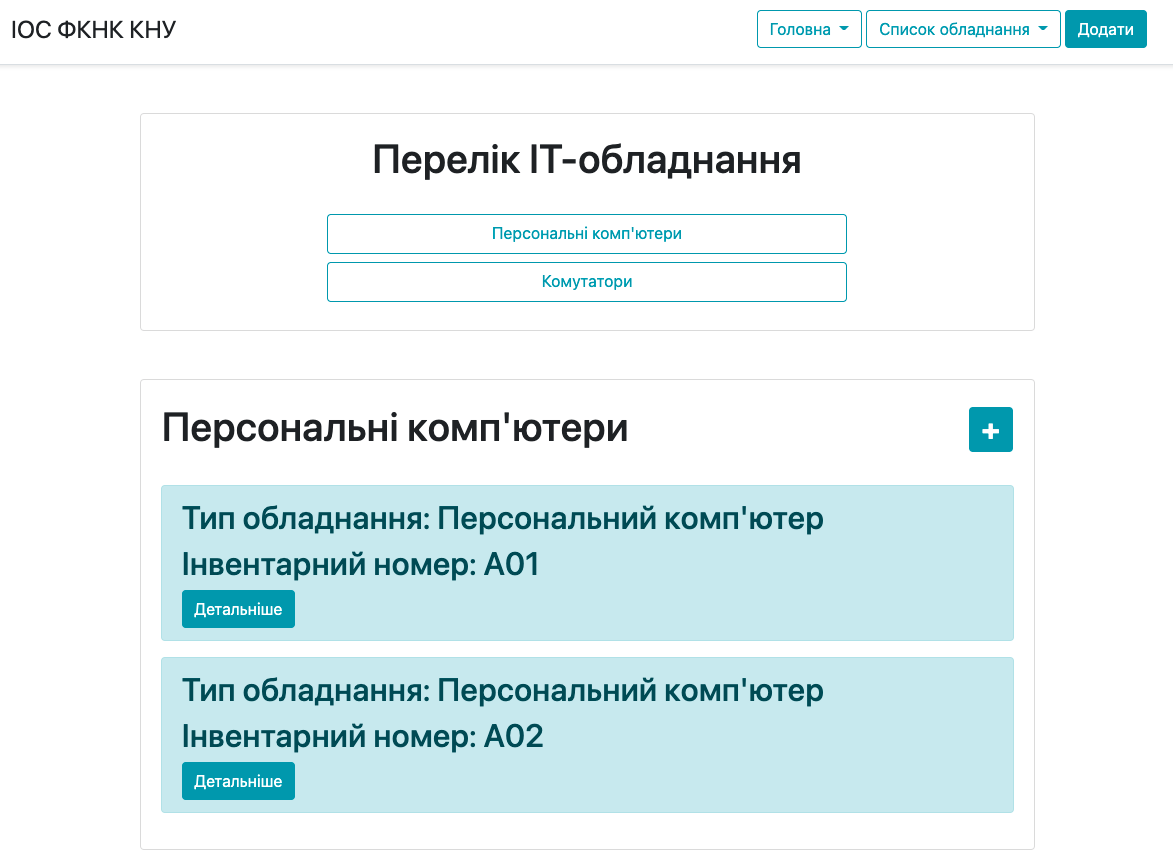
Розроблено форму для додавання нового обладнання. Користувач вводить дані чи вибирає з переліку в залежності від типу поля та зберігає нове сконфігуроване обладнання до бази даних, натиснувши кнопку “Створити”.



Ілюстрація 8

Вікно зі списком обладнання

В даному вікні представлено перелік обладнання. Кожен тип обладнання винесено в окремий фрейм. Кожен фрейм має заголовок - назву обладнання та функціональну кнопку - додавання нового об’єкту даного типу до бази даних. На початку сторінки представлено перелік обладнання. Натиснувши на певне обладнання, користувач автоматично пересувається по сторінці на початок того фрейму, в рамках якого йде перелік обладнання даного типу.



Ілюстрація 9

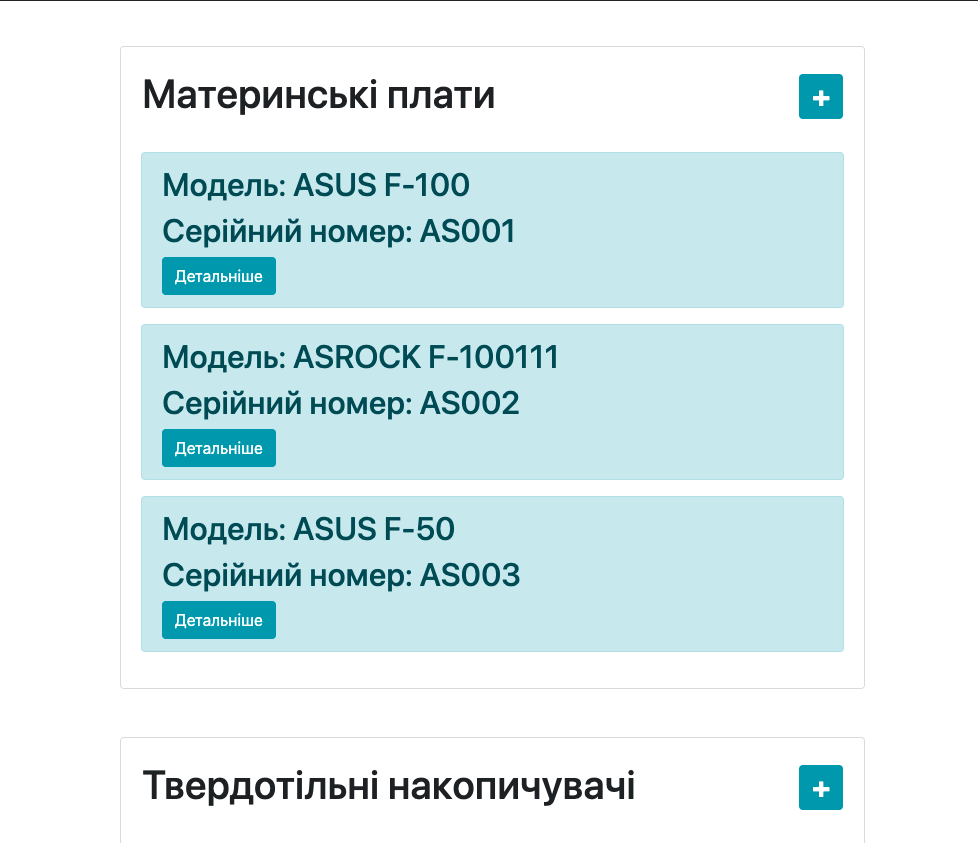
Вікно зі списком комплектуючих до обладнання

В даному вікні представлено перелік комплектуючих до персонального комп’ютера. Логіка візуалізації та функціонал розроблено аналогічно до вікна зі списком обладнання. Це зроблено з метою підтримання єдиного дизайну веб-сервісу та для зручності користувачів.

Изображение выглядит как стол

Автоматически созданное описание

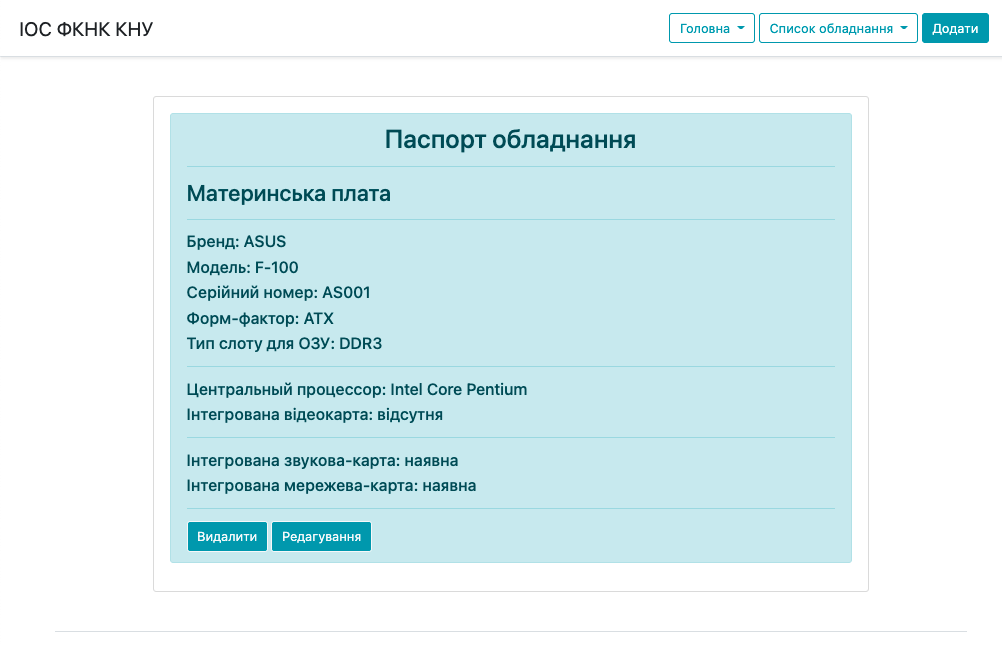
Ілюстрація 10



Ілюстрація 11

Паспорт обладнання

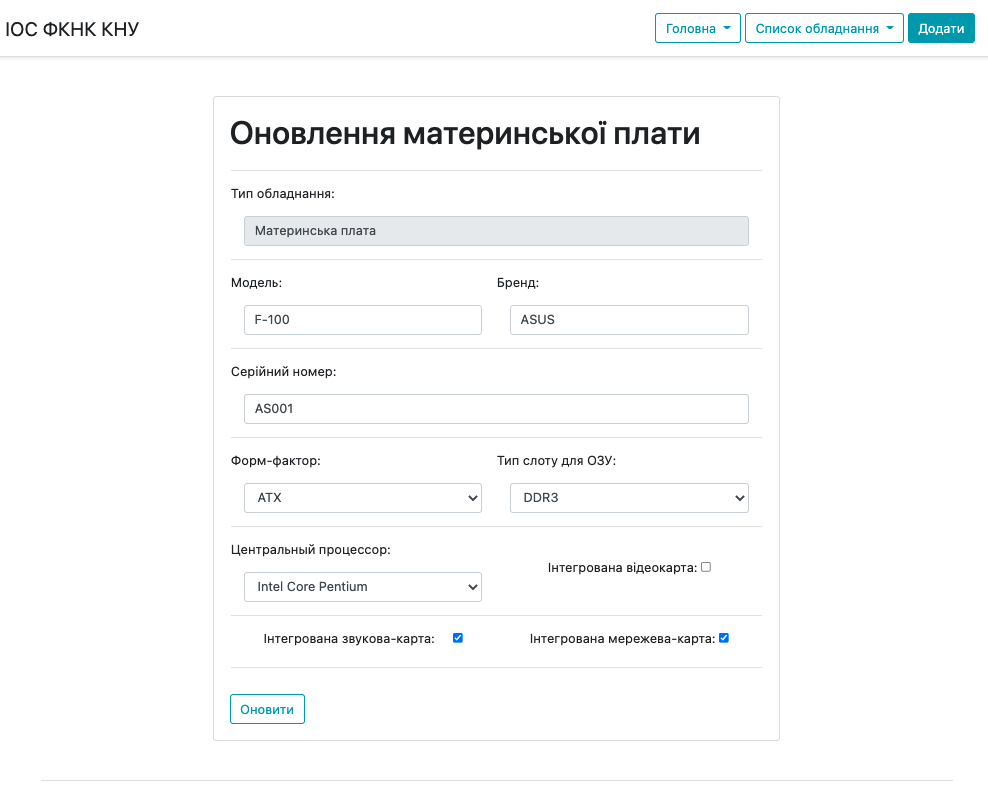
Розроблено окрему сторінку для перегляду повної інформації про обладнання, наявні функціональні кнопки для редагування або видалення обладнання з бази даних.



Ілюстрація 12

Вікно редагування обладнання

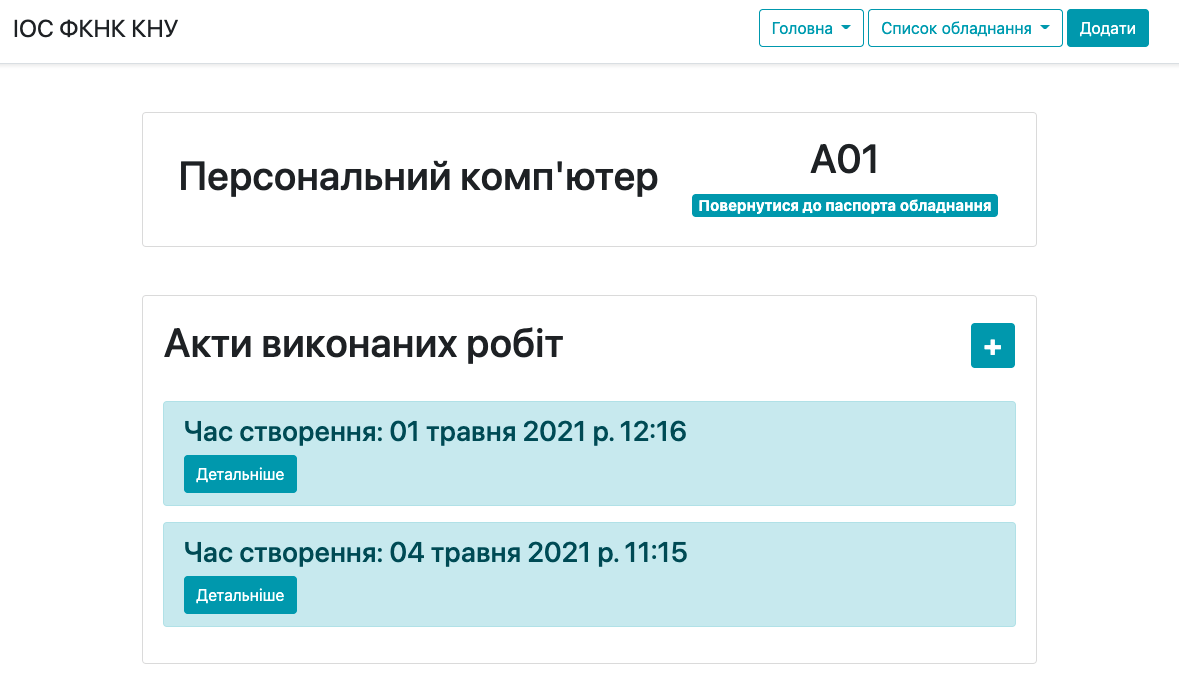
Розроблено форму для редагування інформації про обладнання. При відкритті форма заповнюється поточними даними про обладнання з бази.



Ілюстрація 13

Вікно перегляду актів виконаних робіт

При розгляді паспорта персонального комп’ютера у користувача є можливість перегляду актів виконаних робіт у відповідному вікні. Механізм перегляду актів подібний до перегляду ІТ-обладнання та комплектуючих до них. Наявна копка для створення нового акту. Кожен акт можна повністю продивитися в окремому вікні.



Ілюстрація 14

Вікно додавання акту виконаних робіт

При відкритті форми для заповнення акту виконаної роботи веб-сервіс автоматично вказує повну дату та час заповнення форми та вказує інвентарний номер персонального комп’ютера. Користувачу залишається тільки описати проведену роботу та зберегти запис до бази даних.

Изображение выглядит как текст

Автоматически созданное описание

Ілюстрація 15

Вікно детального перегляду акту виконаних робіт

Дане вікно надає можливість користувачу переглянути опис виконаних робіт. Також присутній функціонал для видалення запису та формування акту проведеної роботи у форматі PDF для подальшого друкування.

Изображение выглядит как текст

Автоматически созданное описание

Ілюстрація 16

ВИСНОВКИ

В даній кваліфікаційній роботі на здобуття освітнього ступеня бакалавра було розроблено веб-сервіс ведення обліку та систематизації ІТ-обладнання. Для апробації було розглянуто заклади вищої освіти, а саме факультет “Комп’ютерних наук та кібернетики” Київського національного університету імені Тараса Шевченка. На даному факультеті наявна велика кількість інформаційно-технічного обладнання, яке використовується у навчальному процесі та у науковій діяльності.

Результатом дослідження та розробки кваліфікаційної роботи є реалізований веб-сервіс. Даний сервіс створено з метою спрощення ведення обліку ІТ-обладнання та дотримання стандарту акту специфікації. Також користувачі мають можливість продивлятися статистику по комплектуючим персональних комп’ютерів у вигляді відповідних діаграм та таблиць. Статистика являє собою дані сформовані на основі технічних характеристик.

Додаток може бути задіяний для ведення обліку та формування актів виконаних робіт системними адміністраторами обраного факультету. Даний веб-сервіс може бути задіяний як інструмент обліку та систематизації обладнання в рамках підприємств. А також в межах закладів вищої освіти з такою необхідністю.

Метод реалізації даного веб-сервісу не прив’язує користувачів до гаджетів певної платформи чи програмного забезпечення. Необхідні лише підключення до мережі Інтернет та наявність веб-браузера на гаджеті.

В подальшому планується розширювати відповідний функціонал шляхом додавання нових типів ІТ-обладнання: комутатори, інтерактивні дошки, проектори, ноутбуки, блоки безперебійного живлення. Також планується удосконалити оформлення актів виконаних робіт та паспортів обладнання.

Нині важливо вміти систематизувати обладнання для оптимального моніторингу та вчасного проведення ремонтних та профілактичних робіт. Очікується, що сервіс знайде практичне застосування й стане корисним інструментом для менеджерів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. В. Дронов. Django 3.0. Практика создания веб-сайтов на Python - видавництво БХВ-Петербург, 2021 - 704 с.
2. Adrian Holovaty, Jacob Kaplan–Moss. The Definitive Guide to Django: Web Development Done Right 1st Edition, 2008 - 447 с.
3. David Amos, Dan Bader. Python Basics: A Practical Introduction to Python 3, 2021 - 635 с.
4. Addison-Wesley Professional; 2nd edition. Effective Python: 90 Specific Ways to Write Better Python, 2019 - 356 с.
5. Офіційний веб-сайт Django, розділ документація. URL: https://docs.djangoproject.com/en/3.2/