Київський національний університет імені Тараса Шевченка
Факультет радіофізики, електроніки та комп'ютерних систем
Кафедра комп'ютерної інженерії

«Управління емулятором фармацевтичного робота-пакувальника»

Випускна кваліфікаційна робота бакалавра
студента 4-го курсу
за спеціальністю 123 «Комп'ютерна інженерія»
Демід СЕМЕНЧЕНКО
_____(підпис)

Науковий керівник, кандидат технічних наук, асистент
Євген СЛЮСАР
_____(підпис)

Рецензент, кандидат фізико-математичних наук, доцент
Віктор ВОЛОХОВ
_____(підпис)

До захисту допускаю
Протокол засідання кафедри від
"____"____2022 р. №_____

Завідувач кафедри
Кандидат фізико-математичних наук, доцент
Юрій БОЙКО

КИЇВ - 2022

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра містить 69 сторінок, 33 ілюстрації, 20 джерел посилань, 6 таблиць, 4 додатки, 9 лістингів коду.

ФАРМАЦЕВТИЧНИЙ РОБОТ-ПАКУВАЛЬНИК, АВТОМАТИЗАЦІЯ АПТЕЧНИХ ПУНКТІВ, АВТОМАТИЗАЦІЯ ФАРМАЦЕВТИКИ, ЕМУЛЯТОР ФІЗИЧНОЇ МАШИНИ, TCP – СЕРВЕР, КОМАНДА, ПРОТОКОЛ КОМУНІКАЦІЇ, МЕРЕЖЕВИЙ ПОРТ, МЕРЕЖЕВИЙ КОНТРОЛЕР, ВАЛІДАЦІЯ

Об’єктом роботи є процес розробки комплексу управління емулятором фармацевтичного робота-пакувальника. Функціонал змодельованої машини повторює процеси реального апарату, що є складовою сфери автоматизації аптечних пунктів.

Метою роботи є створення протоколу комунікації, емулятора фармацевтичного робота-пакувальника та керуючого додатка, який виконує функції управління аналогом робота.

Методи розроблення: визначення вимог до комплексу, розробка протоколу комунікації, що складається з команд – спеціальної послідовності кодів, які можуть бути оброблені моделлю фізичної машини.

Результати роботи: виконано програмну реалізацію фізичної машини, що може виконати процес пакування таблетки з зазначеної касети у вказану комірку упаковки. Програмне забезпечення керується через додаток управління, використовуючи обмін TCP-повідомленнями.

Розроблене рішення може бути застосовано у якості демонстрації для потенційних користувачів роботів автоматизації аптечних пунктів, як навчальний емулятор для фармацевтів у закладах та як приклад кінцевої системи для розробників, що працюють над інтеграційними рішеннями.

ЗМІСТ

ВСТУП

Фармацевтичні роботи-пакувальники є важливим продуктом у сфері автоматизації аптечних пунктів. Автоматизація аптечних пунктів значно спрощує задачу обліку та розповсюдження медичних засобів, покращує умови праці фармацевту, дозволяє зменшити споживання ресурсів на транспортування та пакування ліків. Із використанням подібного рішення, швидкість обслуговування клієнтів також вдається підвищити. Так, наприклад, сучасні апарати Litrea III японської компанії Yuyama надають можливості у виконанні 60 операцій пакування у хвилину [1]. Роботи-пакувальники є досить комплексним пристроєм, тому для їх інтеграції у аптечні пукти потрібне розуміння користі функціоналу зі сторони керівництва та вміння експлуатації зі сторони фармацевтів, що працюють у закладах.

Процес інтеграції апаратів для автоматизації аптечних пунктів супроводжується навчанням персоналу вмінню користуватися складним обладнанням. Використання фізичної машини для подібного, інколи, не є можливим. Розробка комплексу емулятора та керуючого веб-додатка допоможе фармацевтам вивчити функціонал апарату, зрозуміти загальні принципи користування та процедури, самостійно реалізувати сценарії пакувального процесу. Також, використання комплексу буде мати користь для розробників, що працюють над інтеграційними рішеннями для роботів-пакувальників різних виробників, адже всі додатки подібного призначення мають спільні підходи до вирішення задачі спілкування із роботом. Важливою підставою для розробки емулятору є скорочення часу навчання та надання можливості отримати більш глибокі знання із процесу користування пакувальним роботом. Однією із функцій комплексу також є демонстрація функціональних можливостей рішення потенційним користувачам.

Об’єктом розробки є програмна реалізація пакувального робота та керуючого веб-додатка, що буде повторювати функції фізичної машини та процеси роботи із нею. Таким чином, комплекс зможе представити систему, що встановлюється у аптечні пункти та використовується для автоматизації фармацевтики у сенсі процесу обслуговування клієнтів.

Одним за найважливіших понять у роботі є протокол комунікації, як спосіб зв’язку між керуючими додатками та емулятором. Його розробці та програмній реалізації також буде приділена значна частина уваги.

Комплекс може бути застосований у якості демонстраційного об’єкту для потенційних користувачів фармацевтичного роботу-пакувальника, в якості навчальної моделі для фармацевтів аптечних пунктів, у які буде інтегруватись система автоматизації та для ознайомлення розробників інтеграційних рішень із підходами роботи з апаратами такого класу.

РОЗДІЛ 1. АНАЛІЗ ГАЛУЗІ АВТОМАТИЗАЦІЇ ФАРМАЦЕВТИКИ

1.1 Вплив фармацевтичного робота на стан автоматизації аптеки

Ключовою метою встановлення засобу у аптечний пункт є прискорення та покращення обслуговування клієнтів. Ця задача виконується завдяки швидкості роботи машини, що як було зазначено раніше, може досягати 60 операцій пакувань у хвилину [1]. Клієнти отримують більш якісне обслуговування і за рахунок того, що використанням подібного рішення, аптеки можуть виконувати гнучкі замовлення і віддавати препарати у штучній кількості, ігноруючи, наприклад, блістерні упаковки виробників ліків. Замість пакування від виробників, користувачі роботів використовують свої упаковки (рисунок 1.1), із якими здатна працювати машина. Такі упаковки, зазвичай маркуються часовими мітками, що також дозволяють використовувати їх як органайзер для клієнтів. Це зазначення є критично важливим, наприклад у випадку обслуговування літніх людей, які приймають велику кількість ліків.

Рисунок 1.1 – приклад упаковки, із якою працюють роботи

Також слід зауважити, що однією із переваг є зменшення коштів на транспортування препаратів, адже організація–користувач роботу більш не є залежною у фасувальних можливостях виробників препаратів (рисунок 1.2). Після отримання розсипу ліків, вони зберігаються у спеціальних касетах, що встановлюються у апарат. Згодом, з цих касет, препарати потрапляють до механічних частин усередині роботу та розфасовуються в упаковку.

Рисунок 1.2 – вигляд касети, у якій зберігаються препарати

Зі сторони фармацевту, що працює із апаратом у аптеці, процес пакування представлений введенням даних замовлення у відповідний додаток, що підключений до фізичної машини. Після цього, як тільки операції роботу завершені, фармацевт забирає готову упаковку та передає її клієнту.

1.2 Проблема інтеграції рішення у аптечний пункт

Організації, що мають намір встановити робота-пакувальника у свої аптеки, інколи, зустрічаються із труднощами даного рішення, адже воно, дійсно є комплексним і потребує розуміння, як зі сторони персоналу, що безпосередньо працює із відповідним обладнанням, так і зі сторони керівництва, відповідального за процеси управління. Більш того, компанії, що замовляють роботів, інколи потребують модифікацій у додатках керування апаратами. Це все призводить до ускладнення процесу інтеграції рішення у аптеки, внаслідок якого клієнти відмовляються від автоматизованих засобів та погоджуються із ручним обслуговуванням фармацевтом своїх клієнтів.

У свою чергу, зі сторони компанії-постачальника роботів, не завжди вдається продемонструвати переваги їх рішення, адже замовники отримують лише поверхні уявлення про кінцеву систему. Можливості для навчання персоналу замовників зумовлені фізичним доступом до відповідних машин. У контексті модифікацій, які інколи потрібні потенційним користувачам, також є свої труднощі - через специфічність обладнання, розробники, інтегруючі фізичні машини із додатками, потребують відповідних прикладів.

Таким чином, підсумовуючи, можна побачити наявність проблеми загальної інтеграції рішення у аптечні пункти, вирішення якої можна досягти за рахунок засобів, що її спрощують, тобто, наприклад, навчально-демонстраційного комплексу, що здатний промоделювати процеси роботи із реальними машинами.

1.3 Постановка задачі

У якості засобу, що може спростити інтеграцію рішення у аптечні пункти пропонується комплекс емулятора та керуючого веб-додатка. Емулятор – програмний додаток, що повторює функціонал реальної машини та набір її основних комплектуючих. Керуючий веб-додаток – програма, що підключається до емулятору та відповідає за функції управління. Фактично, керуючий додаток – аналог системи, що розгортається у аптеці, із допомогою якої фармацевт задає інформацію замовлення для машини.

Протокол комунікації – важлива складова комплексу, яка являє собою перелік текстових команд, отримуючи які, емулятор виконує потрібні процедури. Команди, відповідно, формуються керуючим додатком.

Задля виконання задачі спрощення інтеграції фармацевтичного робота-пакувальника у аптечні заклади, потрібно розробити архітектуру та на її основі створити відповідний навчально-демонстраційний комплекс, комунікація між складовими якого буде відбуватись із використанням побудованого протоколу комунікації. Спілкування між складовими проходить по мережі, аналогічно до того, як це роблять реальні машини [2], [3]. Користувацькі інтерфейси для роботи із рішенням мають бути простими та зрозумілими для фармацевтів, що не мають технічного фаху.

Використовуючи розроблене рішення, персонал, не маючи доступу до реальної машини, зможе заздалегідь практикуватись у роботі з аналогом роботу та досліджувати процес пакування. Керівництво, організації, що має намір встановити відповідне рішення у свої заклади отримує чітку демонстрацію кінцевої системи та одразу побачить переваги автоматизації. Розробники, що працюють над модифікаціями та додатковими інтеграційними програмами для апаратів, зможуть проаналізувати роботу протоколу комунікації між емулятором та веб-додатком, та ознайомитись із принципами роботи внутрішніх підсистем апарату.

РОЗДІЛ 2. ВНУТРІШНЯ БУДОВА ЕМУЛЯТОРА

Кінцевим образом системи, яка представлена комплексом емулятора робота-пакувальника та керуючого додатка є дві зв’язані мережею програми. Цей комплекс, власно, і є демонстрацією повного рішення, яке пропонують компанії-постачальники зі сфери автоматизації фармацевтичної галузі.

2.1 Вимоги до емулятору робота-пакувальника

Виходячи з представлення кінцевої системи, емулятор повинен являти собою додаток із наступними складовими:

— промодельовані основні комплектуючі фізичної машини, наприклад такі як відсік у якому міститься упаковка або панель із касетами, у яких містяться препарати. Стан цих компонентів повинен змінюватись у ході виконання емулятором тих чи інших процедур.

— система отримання команд, наведених у протоколі, причому при отриманні команди з мережі, вона повинна перевірятись на коректність та можливість виконання у вибраний момент часу;

— система обробки вхідних команд, що розбирає їх параметри та в залежності від них змінює стан промодельованих комплектуючих машини. Наприклад, в залежності від отриманого переліку команд для пакування, стан промодельованої упаковки змінюється відповідним чином.

— інтерфейс індикації поточного стану машини. Інтерфейс емулятору повинен компенсувати візуальний контакт із апаратом, що має місце у випадку роботи із реальною машиною.

Після отримання повідомлення-запиту (наприклад, запит на пакування

таблетки із вказаного контейнеру у вказану комірку упаковки), сервер емулятору повинен обробити запити, виконуючи всі зв’язані із цим функції та надати клієнту повідомлення-відповідь. У більшості випадків, після виконання запиту, емулятор змінить стан машини (наприклад, при запиті на розблокування дверці відсіку, де зберігається адаптер із упаковкою).

Перед прийняттям до обробки повідомлення, програма робота-пакувальника повинна провалідувати його. Це є важливою складовою комунікації, адже іноді можливі ситуації, коли до машини надійдуть повідомлення, що не підпадають під формат протоколу або у собі несуть коди команд, які наразі не можуть бути виконані пристроєм.

Якщо повідомлення виявиться невалідним, емулятор все-одно повинен на нього відповісти. Для цього у протоколі комунікації повинні бути визначені команди із спеціальними кодами, які вкажуть на невірний формат команди або на неможливість машини виконати її у даний момент часу. Якщо ж команда не була розпізнана як наведена у протоколі, то у відповідь повинно надійти повідомлення вигляду “Нерозпінена команда”.

Одна із вимог до емулятору – можливість відслідкувати фізичний стан машини у певний момент часу. Так як машина реалізована програмно, ми не маємо візуального контакту із роботом, як це б було при взаємодії із реальним апаратом, але ми можемо промоделювати її стан програмним об’єктом. Так наприклад, при вилученні адаптеру з пакувального відсіку, програмний об’єкт змінює статус адаптеру на “Відсутній”. Доступ до описаного машинного стану ми можемо отримати від контролера веб-додатка та побачити на інтерфейсі емулятора, що має безпосередній зв'язок із контролером.

Подібним чином можна також реалізувати і фізичну взаємодію із апаратом. Так, при наявності фізичного доступу, ми могли би вийняти упаковку з машини, але у випадку емулятора, ми виконаємо запит до контролера, натиснувши на відповідну кнопку інтерфейса.

2.2 Основні елементи у фізичному представленні машини

Для розуміння роботи емулятора, важливим є фізичне представлення машини. Як можна побачити, на рисунку 2.1 виділені основні компоненти роботу, які мають бути промодельовані у емуляторі.

Так, наявні медикаменти зберігаються у касетах, з яких вони при надходженні відповідної команди будуть переміщені у комірки упаковки. У машині, що моделюється, розміщено 40 касет, а на упаковці, що буде використовуватися – 28 комірок.

Упаковка поміщується у спеціальний адаптер та зберігається у відсіку, доступ до якого закритий спеціальною дверцею. Напочатку пакування, дверця блокується магнітним замком, команду на відкривання якого можна надіслати після закінчення пакування.

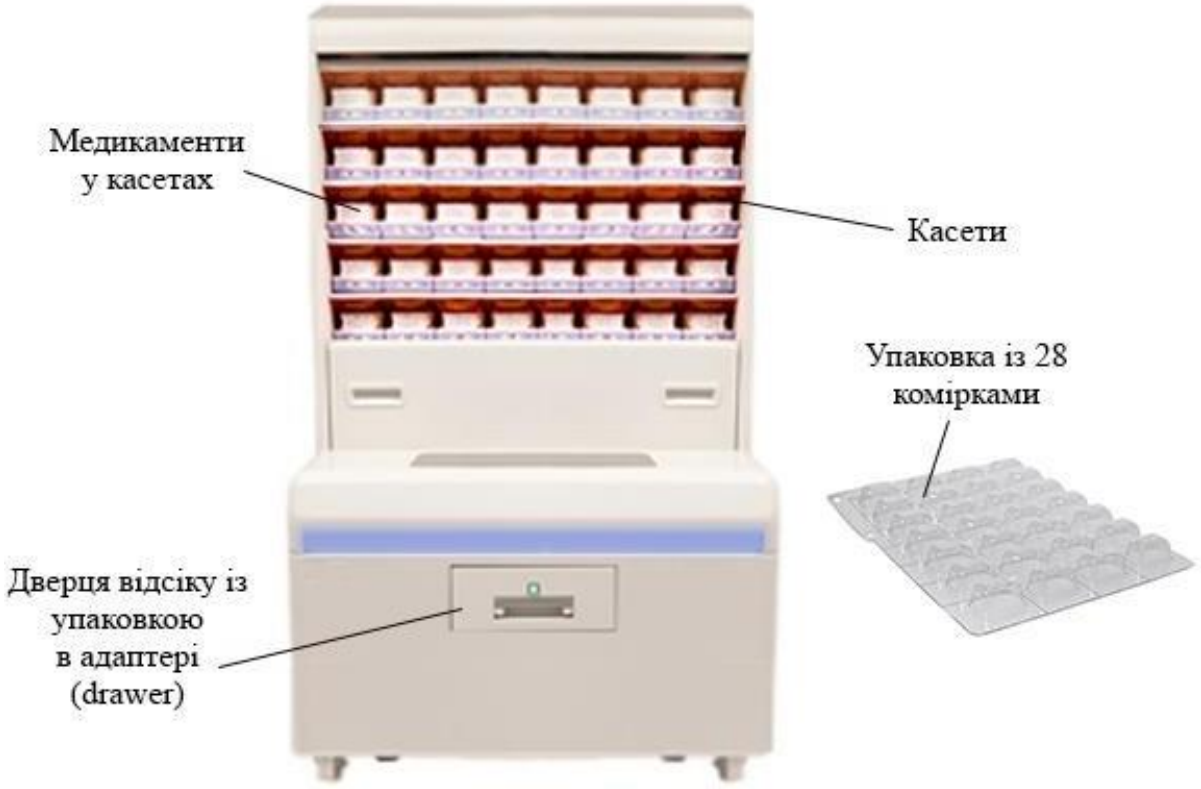


Рисунок 2.1 – представлення умовної фізичної машини

2.3 Програмні складові емулятора

Емулятор містить у собі декілька важливих програмних сутностей, що можуть знадобитися для розуміння його роботи. Серед них:

— Об’єкт поточного стану машини;

— Об’єкт упаковки;

— Зареєстрований рецепт;

— Контролер емулятору;

— Інтерфейс емулятору;

Об’єкт поточного стану машини – програмна сутність емулятора, що відображує поточний фізичний стан машини та зберігає дані з поточної сесії роботи із емулятором (рецепти).

Отримавши цей об’єкт через веб-контролер емулятору можна дізнатися інформацію про: зареєстровані медикаменти та касети у яких вони зберігаються, стан дверці відсіку із адаптером упаковки, поточний стан адаптеру (чи вставлений він у машину), поточний стан об’єкту упаковки, перелік касет, що потребують уваги після пакування.

Об’єкт упаковки містить у себе інформацію про поточний стан упаковки, що закріплена у адаптері. Упаковка розрахована на 28 комірок, кожна з яких має ім’я (Рисунок 2.2) та може бути наповнена таблетками.

Така ж сама упаковка, звичайно має місце й для реальної машини, але у емуляторі вона промодельована програмно.

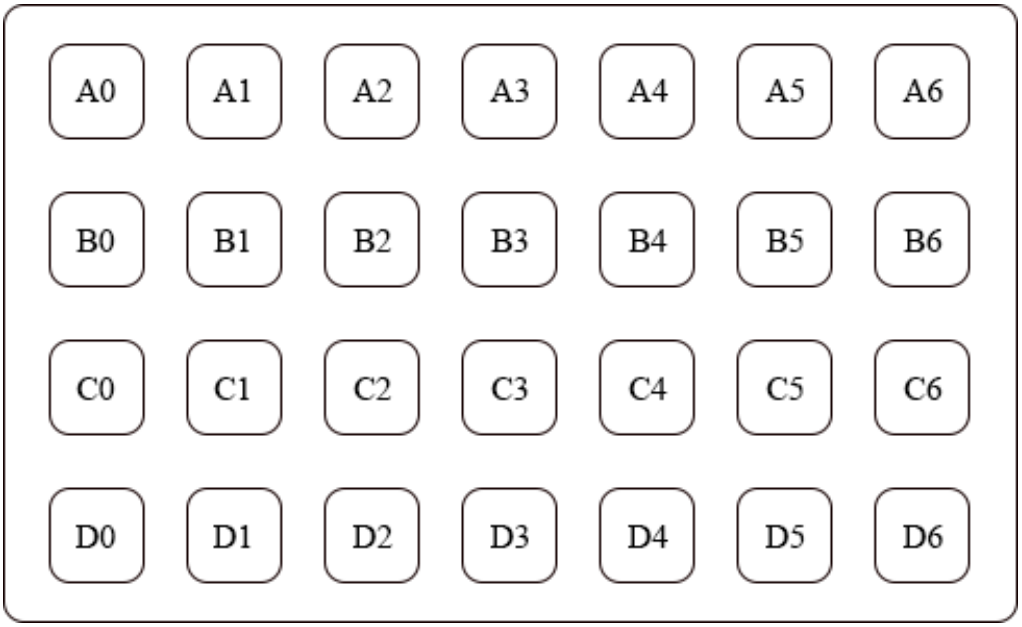


Рисунок 2.2 – схема упаковки на 28 комірок

Інформація про зареєстровані у даній сесії роботи емулятора рецепти міститься у об’єкті машинного стану. Так, щоб виконати пакування (надіслати валідну команду заповнення комірки) потрібно щоб касета, яка вказана як застосована для цього у команді заповнення комірки, була раніше зареєстрована у рецепті. Зареєстрований рецепт містить у себе інформацію про зареєстровані касети, ліки та кількість ліків у касеті.

Контролер емулятору – програмний компонент, що надає можливості через мережу по протоколу HTTP отримати запити на отримання об’єкту поточного стану машини та на фізичну взаємодію із роботом, що неможлива шляхом відправлення команд (наприклад: витягнути упаковку з машини).

Звичайно, такого контролеру немає у реальній машині, але із фізичним фармацевтичним роботом ми маємо візуальний контакт та можемо із ним безпосередньо взаємодіяти. Тому, таке спрощення має місце у програмній реалізації машини.

Інтерфейс емулятору – програмний компонент, який компенсує візуальну індикацію реальної машини, тому як і контролер є спрощенням. Інтерфейс емулятору, насправді представлений вбудованим клієнтським додатком, якій одночасно запускається із основною програмою. Цей клієнтській додаток відповідає за відображення поточного стану машини, отримуючи його через запит до контролера. Всі функції доступні через контролер, наприклад, як запит на відкривання дверці, можна виконати натисканням відповідної кнопки на інтерфейсі.

2.4 Протокол комунікації та вимоги до нього

Протокол комунікації між емулятором та керуючим додатком повинен містити повний перелік текстових команд, що у свою чергу складаються з послідовностей параметрів, в залежності від яких виконуються операції із емулятором. Набір команд має надавати можливість додаткам, що працюють із апаратом усі можливості для виконання головної функції машини – пакування.

2.5 Поняття команди протоколу комунікації із машиною

Ключовим поняттям при комунікації з машиною є команда. Команда – набір спеціальних, визначених протоколом кодів (параметрів), комбінація з яких у випадку роботи з емулятором оброблюється наступним чином (Рисунок 2.3):

— валідується машиною на співпадіння із форматом та на можливість виконання у даний момент часу;

— викликає виконання машиною визначених функцій (якщо команда валідна);

— формує відповідь машини;

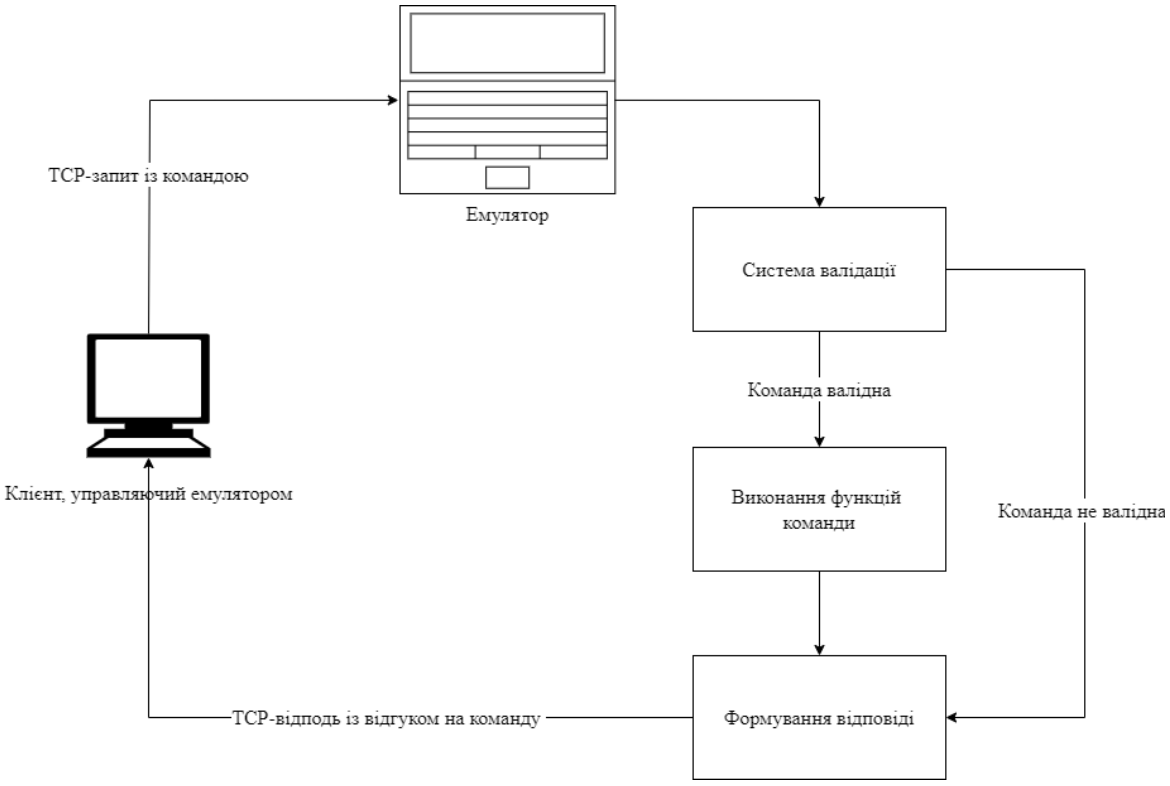


Рисунок 2.3 – процес оброблення емулятором команди

2.6 Перелік команд, що підтримує емулятор

Програмна реалізація фармацевтичного роботу-пакувальника, відповідно до протоколу комунікації (додаток А) підтримує 6 типів команд та відповідей на них, серед команд:

— Команда ініціалізації (індекс команди IN);

— Команда заповнення комірки (індекс команди FL);

— Команда реєстрації рецепту (індекс команди PR);

— Команда запиту машинної активності (індекс команди MR);

— Команда статусного запиту (індекс команди SR);

— Команда невизначеного запиту (індекс команди NR);

2.7 Розбір команди заповнення комірки

Для прикладу розберемо команду заповнення комірки, що підтримується протоколом комунікації із машиною, тобто розглянемо як саме формується послідовність спеціальних кодів.

Приклад реалізації валідної команди заповнення комірки:

“FLC1M100019P0091A0MC0210031102” (2.4)

Для виділення з команди кожного зі спеціальних кодів, звернемося до протоколу. У ньому вказано, під якими позиціями у стрічці знаходиться значення того чи іншого коду та його довжина. Таким чином, маємо:

Назва коду	Значення	Позиції у стрічці
Індекс команди	FL	0 - 1
Відправлено від	C1	2 - 3
Відправлено до	M1	4 - 5
Довжина даних	00019	6 - 11
Початок інформації про рецепт	P	12
Індекс рецепту	0091	13 - 17
Позиція комірки	A0	18 - 19
Початок інформації про видачу	M	20
Початок інформації про касети	C	21
Загальна кількість касет	02	22 - 23
Індекс касети № 1	10	24 - 25
Кількість медикаментів з касети № 1	03	26 - 27
Індекс касети № 2	11	28 - 29
Кількість медикаментів з касети № 2	02	30 - 31

Таблиця 2.5 – розбір кодів команди заповнення комірки

2.8 Спільна частина програмних кодів

Розібравши команду заповнення комірки та звернувши увагу на протокол комунікації (додаток А) із машиною, можна побачити, що всі команди мають спільну початкову частину, у якій в залежності від команди відрізняється тільки ідентифікатор (індекс) команди. Тому, під час проектування емулятору, можна реалізувати програмне наслідування всіх команд від базового класу команд, що буде містити спільний функціонал.

2.9 Опис функціоналу команд

Кожна з команд, що наведена у протоколі спілкування із емулятором має свій функціонал та схему відповіді машини на неї. Функціонал кожної з 6 команд буде наведений у наступних розділах.

2.10 Функціонал команди ініціалізації

Відправляючи команду ініціалізації, користувач може скинути об’єкт поточного стану машини до початкового налаштування. Подібна команда потрібна у випадку коли ми, наприклад, хочемо видалити всі зареєстровані рецепти з машини, що залишилися від попередньої сесії роботи із нею. Успішне виконання команди також видалить зі стану всі касети, що потребували уваги та заблокує замок на дверці до відсіку із адаптером.

Відповідь на команду (додаток А) буде містити код відхилення 01, коли синтаксис команди був невірний.

2.11 Функціонал команди заповнення комірки

Відправляючи команду на заповнення комірки, користувач може виконати наповнення комірки заданою кількістю ліків із касети, що раніше була зареєстрована відповідною командою. Успішне виконання команди заповнить комірку таблетками з заданої касети.

Відповідь на команду (додаток А) буде містити код відхилення 01, коли синтаксис команди був невірний. Код 02 буде міститься у відповіді у випадку:

— Коли дверця відсіку із адаптером відкрита або адаптер відсутній у машині;

— Зареєстрованого рецепту, на основі якого ми виконуємо пакування не існує;

— Касета, яку ми використовуємо для пакування не була зазначена у рецепті;

— Кількість ліків у касеті є меншою, аніж ми хочему запакувати, у цьому випадку касета попадає до списку касет, що потребують уваги у об’єкті поточного стану машини;

2.12 Функціонал команди реєстрації рецепту

Відправляючи команду на реєстрацію рецепту ми вказуємо касети, які ми плануємо використати для пакування, назву ліків, що у них містяться та кількість цих ліків. Успішне виконання команди додає до переліку рецептів поточного стану машини зареєстрований рецепт.

Відповідь на команду (додаток А), буде містити код відхилення 01, коли синтаксис команди був невірний. Код 02 буде міститься у відповіді у випадку коли рецепт із заданим ідентифікатором вже зареєстрований.

2.13 Функціонал команди запиту машинної активності

Відправляючи команду запиту машинної активності ми можемо розблокувати або заблокувати дверцю відсіку із адаптером та очистити перелік касет, що потребують уваги.

Відповідь на команду буде містити код відхилення 01, коли синтаксис команди був невірний.

2.14 Функціонал статусної команди

Відправляючи статусний запит ми хочемо у відповіді отримати деякі дані із поточного стану машини, серед яких: статус блокування дверці відсіку із адаптером, наявність адаптеру із упаковкою усереднені машини, перелік касет, що потребують уваги.

— статус блокування дверці відсіку із адаптером;

— наявність адаптеру із упаковкою усереднені машини;

— перелік зареєстрованих рецептів та їх кількість;

— перелік касет, що потребують уваги та їх кількість;

Відповідь на команду буде містити код відхилення 01, коли синтаксис команди був невірний.

2.15 Функціонал команди невизначеного запиту

Команда невизначеного запиту – є єдиною командою, що вказана у протоколі та не може бути відправлена до машини. До її опису входить тільки відповідь. Відповідь команди невизначеного запиту може надійти клієнту у випадку, коли емулятор отримує повідомлення, яке не може розпізнати як жодну команду з протоколу. Використовується при валідації повідомлення.

2.16 Стандартний алгоритм пакування

На рисунку 2.6 наведено стандартний алгоритм для взаємодії з емулятором із допомогою якого можна виконати пакування зареєстрованих ліків із вказаної касети. Саме цей алгоритм – головна функція керуючого сайту. Сформувавши команди, керуючий сайт передає їх до емулятору на виконання.

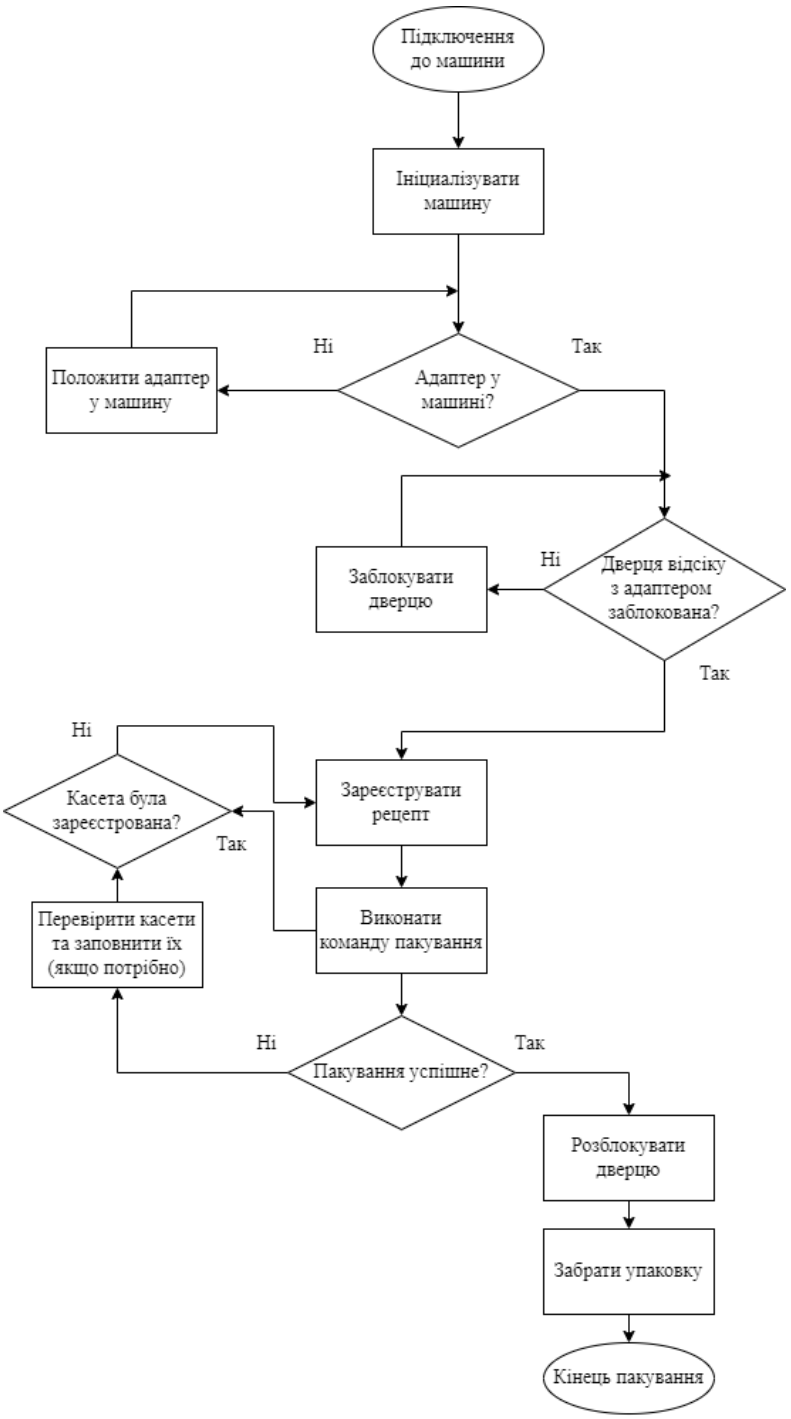


Рисунок 2.6 – стандартний алгоритм пакування

РОЗДІЛ 3. КЕРУВАННЯ ЕМУЛЯТОРОМ ТА АРХІТЕКТУРА КОМПЛЕКСУ

3.1 Вимоги до керуючого додатка

Керуючий веб-додаток, виходячи із призначення кінцевої системи має наступний функціонал:

— створення та підтримка мережевого підключення до емулятору, через яке буде вестись комунікація;

— можливість введення користувачем (фармацевтом) деталей замовлення для подальшого оброблення емулятором та виконання пакування;

— в залежності від поданого у додаток запиту – формування комбінації команд із параметрами, що відтворюють замовлення клієнта, що обслуговується;

3.2 Формування команд керуючим додатком. Поняття операції

Головна функція керуючого додатка – подати команди, необхідні для виконання процедури пакування. Таким чином, кожна з шести команд для роботи із емулятором повинна бути реалізована програмно. Параметри команди задаються користувачем додатку відповідно до замовлення.

Як можна побачити із протоколу (додаток А), будь-яка команда – сутність, що має низькій рівень абстракції, тому, самостійно або у послідовності з іншими командами, скоріш за все, буде не зрозуміла фармацевту, як людині, що не має відповідного рівня ознайомлення із предметною областю. Тому, пропонується ще один рівень абстракції над командою – операція.

Операція – обгортка над послідовностями команд чи інших операцій та логікою роботи із ними, що має вищій рівень абстракції та безпосереднє відноситься до загальних функцій емулятору (або у реальному випадку – фізичної машини). Так, замість того, щоб виконувати пакування ручним введенням послідовності команд до емулятору, користувач лише задає параметри, необхідні для виконання операції, яка у свою чергу в залежності від цих параметрів формує відповідні команди.

Так, керуючий додаток підтримує 3 операції, серед яких:

— операція ініціалізації, що надає функціонал із створення мережевого зв’язку із емулятором та ініціалізації його;

— операція отримання статусу емулятора, що надає функціонал із отримання поточного статусу машини із даними про стан комплектуючих, інформація про які отримується із статусної команди протоколу;

— операція пакування, що формує необхідну послідовність команд. Ця операція, виходячи з параметрів, автоматично реєструє рецепт, та виконує пакування препаратів з вибраних касет у відповідні комірки упаковки. Також вона підготовлює керуючий додаток до роботи із машиною, тобто якщо потрібно – всередині викликає операцію ініціалізації та збирає всю інформацію про стан емулятора до і після пакування операцією отримання статусу;

3.3 Операція ініціалізації

Операція ініціалізації відповідає за функції із створення мережевого зв’язку із емулятором та його ініціалізації, блок-схему її роботи можна побачити на рисунку 3.1:

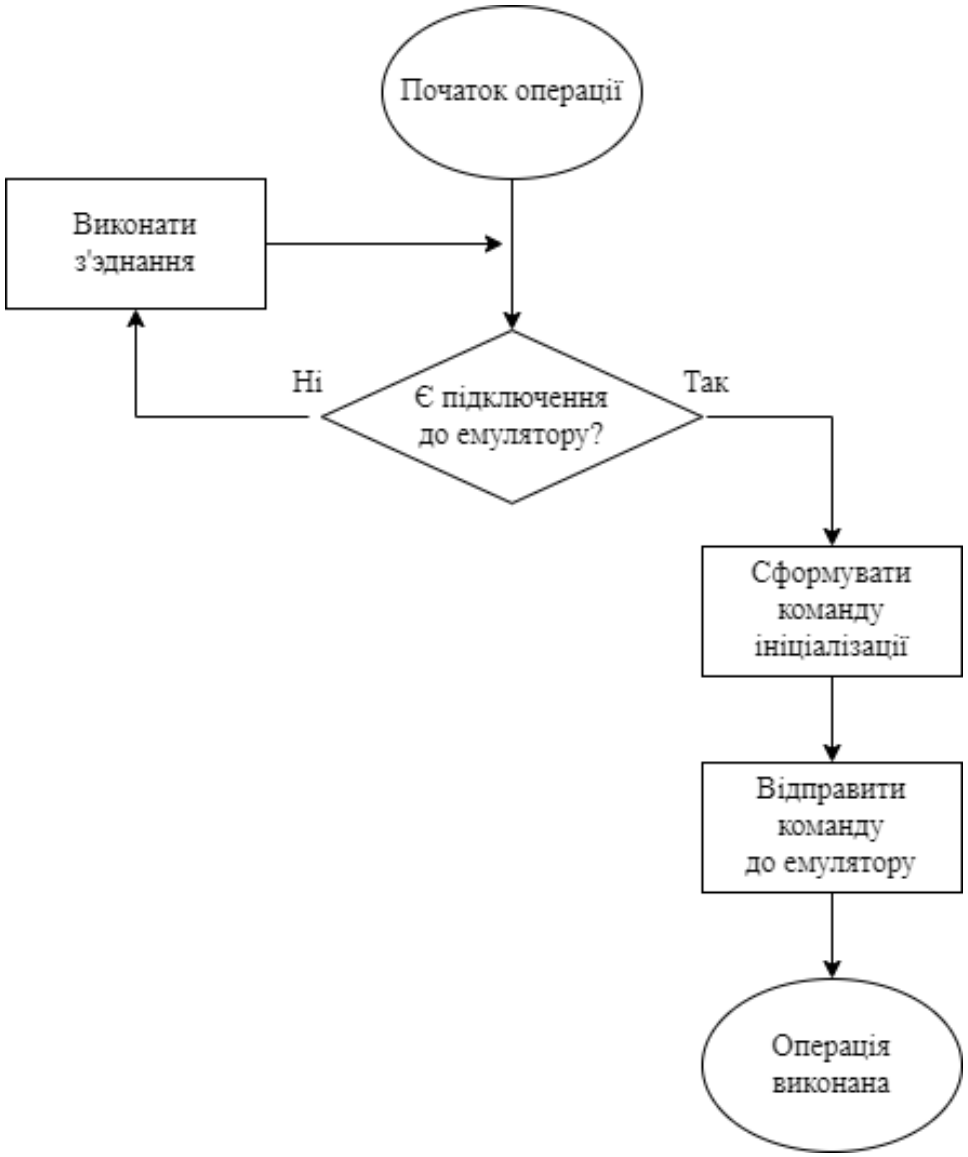


Рисунок 3.1 – операція ініціалізації

3.4 Операція отримання статусу та об’єкт машинного стану

Операція отримання статусу лише формує відповідну команду з протоколу (додаток А), але більш важливим у цьому сенсі є результат виконання цієї операції. Отримавши статусний запит, емулятор у відповідь надсилає статусну відповідь, яка після обробки поновлює об’єкт машинного стану, що містить наступні властивості:

Властивість	Призначення
Дверця відсіку з упаковкою розблокована	Флаг для визначення статусу блокування відсіку з упаковкою, якщо розблокований, то пакування неможливе
Адаптер знаходиться у відсіку з упаковкою	Флаг для визначення присутності адаптера у відсіку, якщо відсутній, то пакування неможливе
Ідентифікатори зареєстрованих рецептів	Колекція значень ідентифікаторів зареєстрованих рецептів. При пакуванні перевіряється, чи вже не присутній у цій колекції рецепт, що намагаються запакувати. Ця колекція запобігає відправленню емулятором відповіді із блокуванням команди через повторну реєстрацію рецепту
Ідентифікатори касет, що потребують уваги	Колекція значень ідентифікаторів касет, що потребують уваги. Ідентифікатор касети потрапляє до колекції, якщо касета не встановлена, але була використана при пакуванні, або пуста і була використана

Таблиця 3.2 – властивості машинного стану

3.5 Операція пакування

Операція пакування є найбільш комплексною, містить у собі додаткову логіку та викликає інші операції – як операцію ініціалізації, так і операцію отримання статусу. У якості параметру, що приймається виступає об’єкт пакувальної операції із наступними властивостями:

Властивість	Призначення
Ідентифікатор рецепту	Ідентифікатор рецепту, що приймається до пакування. Є одним з параметрів команди реєстрації рецепту з протоколу (додаток А)
Препарати для пакування	Колекція з об’єктів препаратів для пакування (таблиця 3.4)

Таблиця 3.3 – об’єкт пакувальної операції

Властивість	Призначення
Ідентифікатор касети	Ідентифікатор касети, що використовується для пакування. Є одним з параметрів команди заповнення комірки та реєстрації рецепту (додаток А)
Назва препарату	Назва препарату, що використовується для пакування. Є одним з параметрів команди реєстрації рецепту
Використані комірки упаковки	Колекція з об’єктів використаних комірок упаковки (таблиця 3.5)

Таблиця 3.4 – об’єкт препарату для пакування

Властивість	Призначення
Ідентифікатор касети	Ідентифікатор касети, що використовується для пакування. Є одним з параметрів команди заповнення комірки та реєстрації рецепту (додаток А)
Назва препарату	Назва препарату, що використовується для пакування. Є одним з параметрів команди реєстрації рецепту
Використані комірки упаковки	Колекція з об’єктів використаних комірок упаковки (таблиця 3.6)

Таблиця 3.5 – об’єкт препарату для пакування

Властивість	Призначення
Позиція упаковки	Назва позиції упаковки (рисунок 2.2), використовується у команді заповнення комірки
Назва препарату	Кількість, шо має бути доставлена до комірки, використовується у команді заповнення комірки та для підрахунку загальної кількості препарату у команді реєстрації рецепту

Таблиця 3.6 – об’єкт використаної комірки

Отримавши об’єкт пакувальної операції ми маємо весь необхідний набір параметрів для формування всіх команд, що використовуються під час пакування. Блок-схема операції пакування вказана на рисунку 3.7:

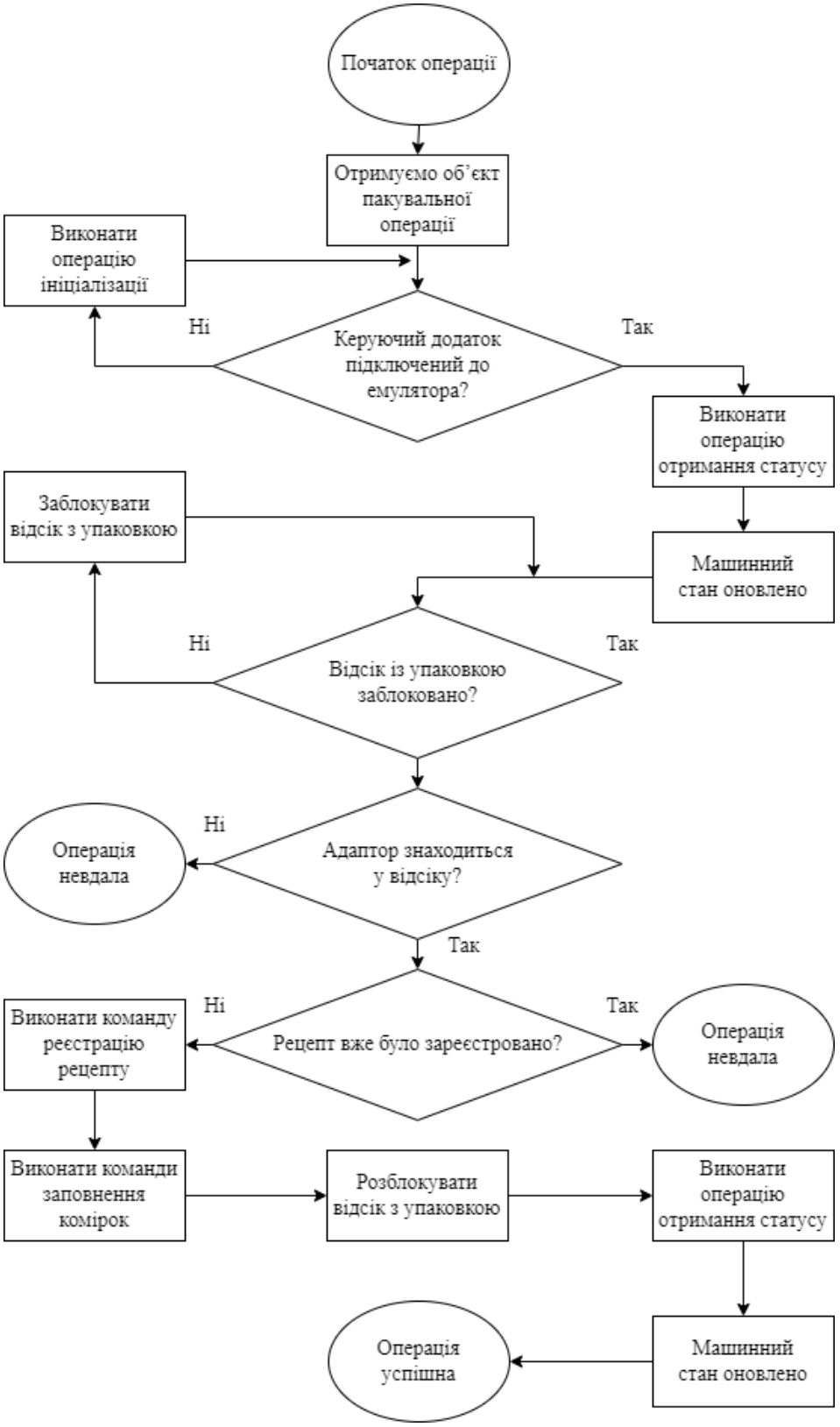


Рисунок 3.7 – блок-схема операції пакування

3.5 Архітектура комплексу

Комплекс емулятору фармацевтичного робота-пакувальника та керуючої програми представлений двома веб-додатками, перший з яких можна розглядати як сервер, другий – як клієнт. Таке твердження є справедливим через аспект мережевої взаємодії розроблених складових. Як і у реальних машинах [2] [3], для обміну повідомленнями використовується TCP - протокол четвертого (транспортного) рівня моделі OSI [4]. Тобто, емулятор у даному випадку повинен містити у себе функціонал TCP-серверу, а керуючий додаток – функціонал TCP-клієнту. Таким чином, у даному випадку рішення будується на базі клієнт-серверної архітектури [5].

Клієнт-серверна архітектура – архітектурний шаблон, що розподіляє компоненти системи на сервери, клієнти та мережу між ними. Архітектура передбачає відсутність жорсткої прив’язки клієнтів до серверів. Характерною рисою є можливість сервера одночасно обробляти запити від декількох клієнтів, які працюють паралельно.

У випадку комплексу, як клієнт, так і сервер представлені програмним забезпеченням.

Що ж стосується взаємодії користувача із емулятором, виклик усіх функцій, тобто, у даному випадку операцій, він робить зі сторони керуючого додатку, якій, як зазначено раніше, є веб рішенням, доступ до інтерфейсу якого, відповідно, можна отримати через браузер.

Також, важливо зазначити складову користувацького інтерфейсу, зі сторони емулятора та керуючого рішення. Інтерфейси двох складових працюють через наявність клієнтських додатків, які стартують одразу із сервером керуючого сайту та сервером емулятора. Сервери у даному випадку надають REST WEB API [6] із яким через протокол HTTP обмінюються повідомленнями клієнтські SPA [7] додатки (рисунок 3.8).

SPA – веб-додаток, що працює на базі документу, що був завантажений лише один раз, а вміст цього документу оновлюється за рахунок, наприклад, API мови програмування JavaScipt. Тобто, інакше кажучи, сторінка у браузері не потребує перезавантаження при зміні свого вмісту.

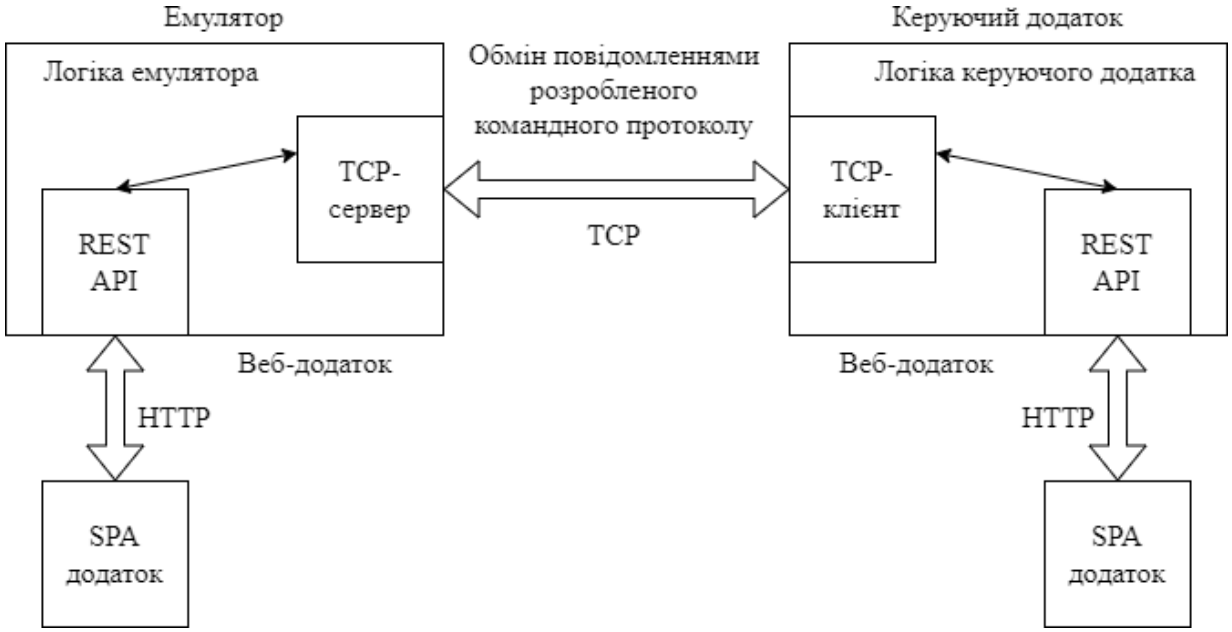


Рисунок 3.8 – архітектура комплексу

Обрана архітектура добре підходить для побудови кінцевої системи через комбінування функціоналу TCP-сервера (TCP-клієнта) та REST WEB API у одному додатку. Веб складові надають можливості для відображення усіх потрібних деталей на вікні браузера, а складові, що працюють із протоколом TCP відповідають за мережевий зв'язок емулятора та керуючого додатка. Звичайно ж, можна було б побудувати цей зв'язок із використанням інших, більш високорівневих протоколів, наприклад таких як HTTP, але, як було зазначено раніше, реальні машини частіше працюють саме із TCP [2] [3].

РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА

4.1 Вибір технологій для побудови комплексу

У якості технології для побудови серверної частини емулятора та додатка була використана мова програмування C# та платформа .NET 5. Так, як функціонал розрахований на взаємодію через ВЕБ, для побудови комплексу застосовувалася платформа ASP.NET Core 5 [8], а саме шаблон для розробки ASP.NET WEB API.

ASP.NET – кросплатформовий фреймворк із відкритим кодом, призначений для побудови веб-застосунків.

ASP.NET WEB API – шаблон для розробки додатків, орієнтований на архітектурний стиль REST.

Мова C# є гнучким інструментом розробки рішень, які працюють із фармацевтичними роботами пакувальниками через наявність широкого функціоналу для роботи із мережевими ресурсами, зокрема із протоколом TCP. Засоби, що допомагають у використанні мережевих сокетів, протоколів UDP та TCP зібрані у просторі імен “System.Net.Sockets” [9].

Клієнтська частина комплексу представлена фреймворком Angular [10], який написаний на мові TypeScript та розрахований на її використання. Фреймворк надає інфраструктуру для побудови SPA [7].

Обрані технології для реалізації WEB API та клієнтської частини не викликають труднощів у поєднанні. Обидві складові розраховані на архітектурний стиль REST тому мають весь спектр можливостей для дотримання його.

4.2 Практична реалізація емулятора

4.2.1 Об’єкт базової команди

Як зазначалося у теоретичній частині, емулятор – програмна складова, основна мета якої полягає у моделюванні стану апарату та зміни його у залежності від команд, що надходять.

Команди протоколу (додаток А), як можна побачити, мають спільну частину кодів, зокрема кожна з команд містить:

— ідентифікатор команди;

— поле відправника;

— поле отримувача;

— довжину інформації, що передається;

На рівні коду, наявність у всіх команд базового набору параметрів можна представити як наслідування від батьківського класу. Об’єкт базової команди на діаграмі класів зображений на рисунку 4.2.1.1.

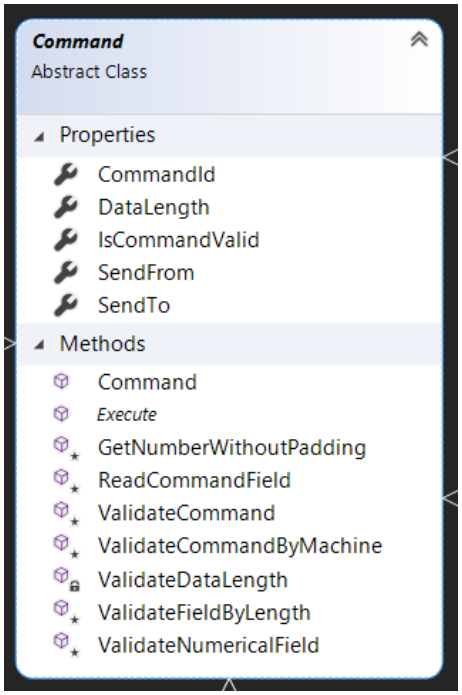


Рисунок 4.2.1.1 – діаграма класу базової команди

Базова команда, окрім стану містить у собі також і поведінку. Таким чином, у ній реалізовані:

— метод виконання команди, що в залежності від стану (параметрів) команди змінює об’єкт, що моделює стан машини;

— методи зчитування параметрів з команди, що надійшла;

— методи валідації команди на коректність формування та на можливість виконання у даний момент часу;

Діаграма класів, що демонструє залежність всіх типів команд від базового зображена рисунку 4.2.1.2. Залежні команди як наслідують батьківські властивості та методи, так і містять свої власні.

Рисунок 4.2.1.2 – діаграма класів команд

4.2.2 Отримання команди

На відміну від керуючого веб-додатку, емулятор отримує та оброблює команди, а не займається їх формуванням. Отримання команди відбувається зчитуванням даних із TCP з'єднання (лістинг 4.2.2.1) та подальшим її відправленням до відповідного обробника у методі "GetResponseFromProcessedMessage" (лістинг 4.2.2.2). Цей метод після обробки команди повертає відповідь емулятора на неї та посилає на клієнт (лістинг 4.2.2.1).

```
public void ReceiveMessage()
{
    byte[] data = new byte[99999];
    try
    {
        while (true)
        {
            StringBuilder builder = new StringBuilder();

            int bytes = 0;
            do
            {
                bytes = _stream.Read(data, 0, data.Length);
                builder.Append(Encoding.ASCII.GetString(data, 0, bytes));
            }
            while (_stream.DataAvailable);

            string message = builder.ToString();
            _logger.LogInformation($"Receive command: {message}");
            string response = GetResponseFromProcessedMessage(message);
            _stream.Write(Encoding.ASCII.GetBytes(response), 0, response.Length);
            _logger.LogInformation($"Send command: {response}");
        }
    }
    catch (Exception ex)
    {
    }
    finally
    {
        if (_stream != null)
            _stream.Close();
        if (_client != null)
            _client.Close();
    }

}
```

Лістинг 4.2.2.1 – метод, у якому відбувається зчитування команди з TCP з'єднання

```
public string GetResponseFromProcessedMessage(string message)
{
   string messageToReturn = _commandHandler.ExecuteCommand(MachineState, message).Response;
   return messageToReturn;
}
```

Лістинг 4.2.2.2 – метод, у якому команда відправляється до обробника

4.2.3 Оброблення команди

Після отримання командного повідомлення, вона потрапляє до обробника, який містить метод “ExecuteCommand” (лістинг 4.2.3.1), у якому відбувається класифікація команди за типом на основі ідентифікатору (додаток А) для її подальшого виконання. Якщо ідентифікатор не вдалося валідувати (ідентифікатор не збігся з жодною командою протоколу), у відповідь на клієнт буде направлена нерозпізнана команда, що починається з індексу “NR” (додаток А). З методу “ExecuteCommand” повертається відповідь емулятора на команду, тобто значення властивості “Response” об’єкту “CommandResponse”.

```
public CommandResponse ExecuteCommand(MachineState machineState,
                string commandString)
{
   string index = GetCommandStringIndex(commandString);
   bool indexIsValid = ValidateCommandIndex(index);
   if (!indexIsValid)
   {
      return GetNotRecongnizedCommand();
   }

   return GetCommand(index, commandString).Execute(machineState);
 }
```

Лістинг 4.2.3.1 – метод, у якому відбувається обробка команди та повернення відповіді емулятора на неї

4.2.4 Валідація та виконання команди

Коли класифікація команди відбулася успішно, викликається метод конструктору команди відповідного типу. У конструкторі кожній з властивостей команди присвоюється значення на основі повідомлення, що раніше надійшло із TCP з'єднання (лістинг 4.2.4.1). Коли всі властивості мають значення, команда перевіряється на валідність (рисунок 4.2.4.2). Якщо вона відповідає синтаксису команди із протоколу (лістинг 4.2.4.3) та можлива для виконання машиною у даний момент часу (лістинг 4.2.4.4), властивість "IsValid" (рисунок 4.2.1.1), успадкована від базової команди отримує значення "True".

```
public MachineActivityRequestCommand(string commandString) : base(commandString)
{
   DataStartNotification = ReadCommandField(commandString,
      MachineActivityRequestValues.DataStartNotificationStartIndex,
      MachineActivityRequestValues.DataStartNotificationLength);
   DrawerStatus = ReadCommandField(commandString,
      MachineActivityRequestValues.DrawerStatusStartIndex,
      MachineActivityRequestValues.DrawerStatusLength);
   ClearingWarningsInitiated = ReadCommandField(commandString,
      MachineActivityRequestValues.ClearingWarningsInitiatedStartIndex,
      MachineActivityRequestValues.ClearingWarningsInitiatedLength);

   string command = CommandId + SendFrom + SendTo + DataLength
   + DataStartNotification + DrawerStatus + ClearingWarningsInitiated;
   IsCommandValid = ValidateCommand(commandString);
}
```

Лістинг 4.2.4.1 – конструктор команди запиту машинної активності (команди із індексом "MR")

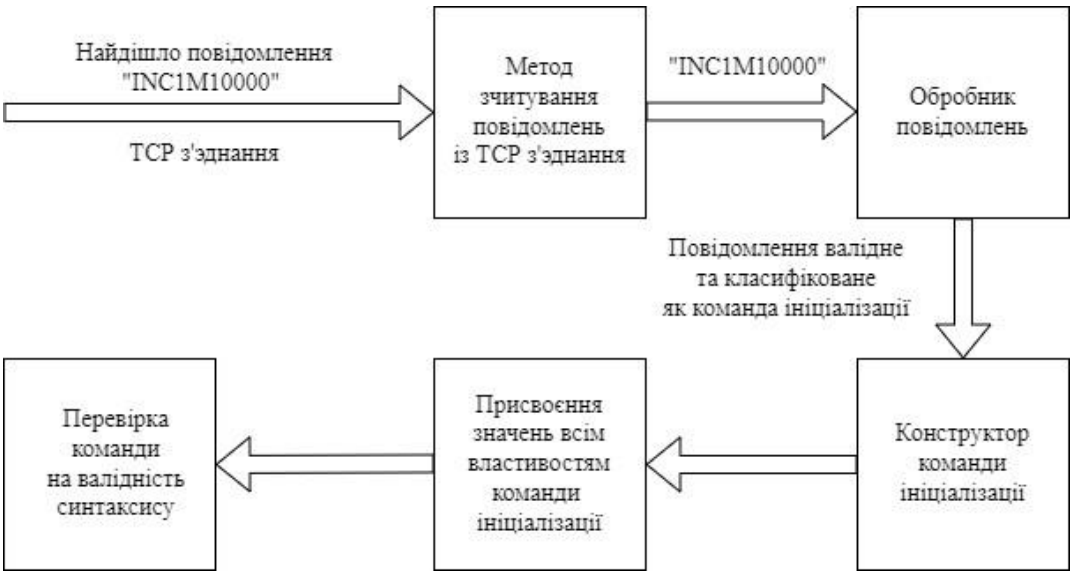


Рисунок 4.2.4.2 – процес оброблення повідомлення
після зчитування із TCP

```
protected override bool ValidateCommand(string commandString)
{
   if (!base.ValidateCommand(commandString))
      return false;
   if (CommandId != InitializationCommandValues.CommandId)
      return false;

   return true;
}
```

Лістинг 4.2.4.3 – валідація синтаксису команди ініціалізації (індекс “IN”)

```
protected override bool ValidateCommandByMachine(MachineState machineState)
{
   int prescriptionId =Convert.ToInt32(GetNumberWithoutPadding(PrescriptionId));

   bool prescriptionExist = machineState.RegistredPrescriptions
            .Any(prescription => prescription.Id == prescriptionId);

   return !prescriptionExist;
}
```

Лістинг 4.2.4.4 – валідація команди реєстрації рецепту (індекс “PR”)
на можливість виконання машиною у даний момент часу

Створення команди має на увазі виклик методу конструктору. Будь-яка створена команда має метод “Execute” (лістинг 4.2.4.5), яка в залежності від значення властивості “IsValid” повертає об’єкт типу “CommandResponse”, що містить відповідь емулятора на команду.

```
public override CommandResponse Execute(MachineState machineState)
{
   if (!IsCommandValid)
   {
      return new FillCommandResponse(CommandResponseCodes.WrongCommandFormat);
   }
   if (!ValidateCommandByMachine(machineState))
   {

      return new FillCommandResponse(CommandResponseCodes.MachineBlockedCommand);
   }
   ExecuteFilling(machineState);

   return new FillCommandResponse(CommandResponseCodes.Sucess);
 }
```

Лістинг 4.2.4.3 – метод виконання команди заповнення комірки (індекс “FL”)

Об’єкт “CommandResponse” (рисунок 4.2.4.6) так само як і “Command” є базовим для підтипів відповідей на команди. Тобто, кожний із типів команд має співставний тип відповіді. Наприклад типу “InitializationCommand” у відповідність ставиться тип “InitializationCommandResponse”.

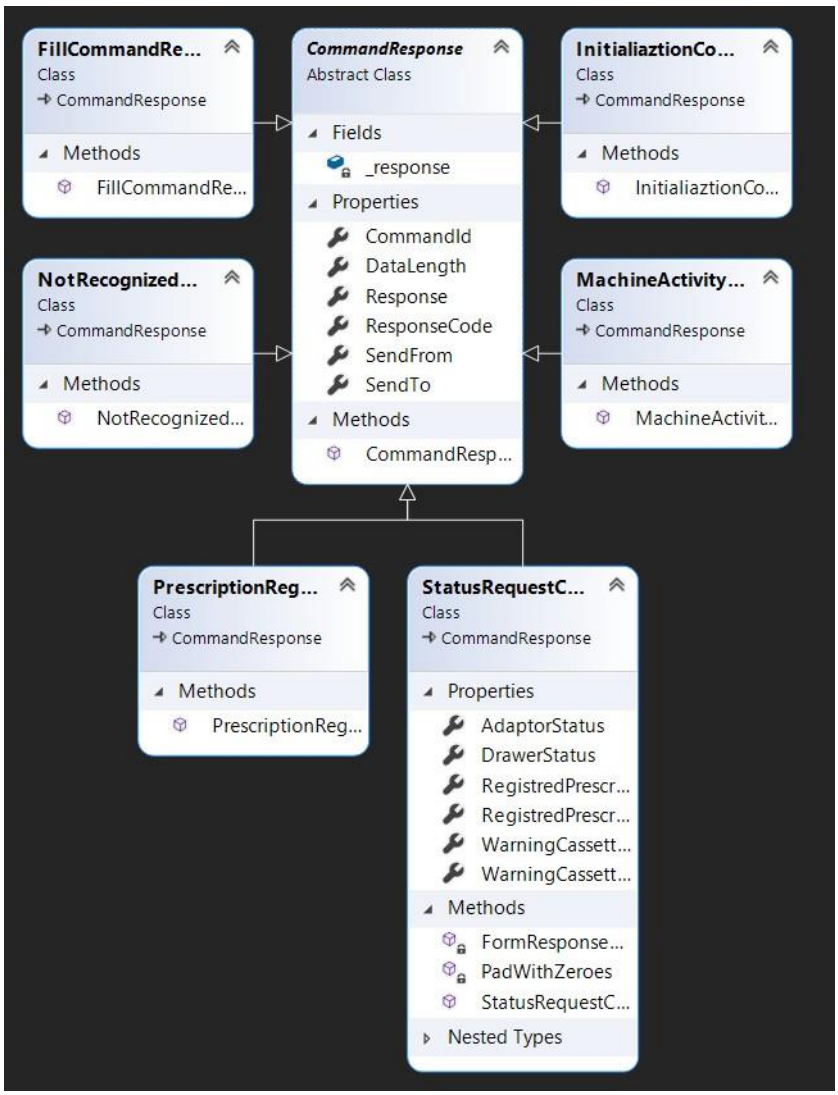


Рисунок 4.2.4.6 – діаграма класів типів відповідей на команди

4.2.5 Реалізація поточного стану машини та фізичних компонентів

Команди, що надходять до емулятора змінюють його поточний стан. Наприклад, після виконання команд, передбачуваних операцією пакування, об’єкт, що моделює упаковку повинен змінитись. Саме через це, ключовою сутністю проекту є клас “MachineState”, який моделює машинний стан. Як було зазначено у теоретичній частині, емулятор також моделює такі фізичні сутності як сама упаковка, касета, адаптер, комірка упаковки, тощо. Діаграма класів, що відображує дані комплектуючі зображена на рисунку 4.2.5.1.

Рисунок 4.2.5.1 – діаграма класів основних складових емулятора

Інформація про поточний стан емулятора є критично важливою для користувача. Вона містить у собі як відображення візуального стану машини (наприклад, у реальному випадку, фармацевт може поглянути через спеціальне вікно на фізичній машині, скільки і де в упаковці наразі лежать препарати), так і дані, що клієнти отримують із статусної команди. У випадку комплексу, ці дані відображуються на стороні клієнтського додатка емулятора, хоча їх можна і отримати керуючому сайтові. Отримання інформації для відображення на веб-сторінці (рисунок 4.2.5.2) SPA відбувається через контролер, HTTP запитами.

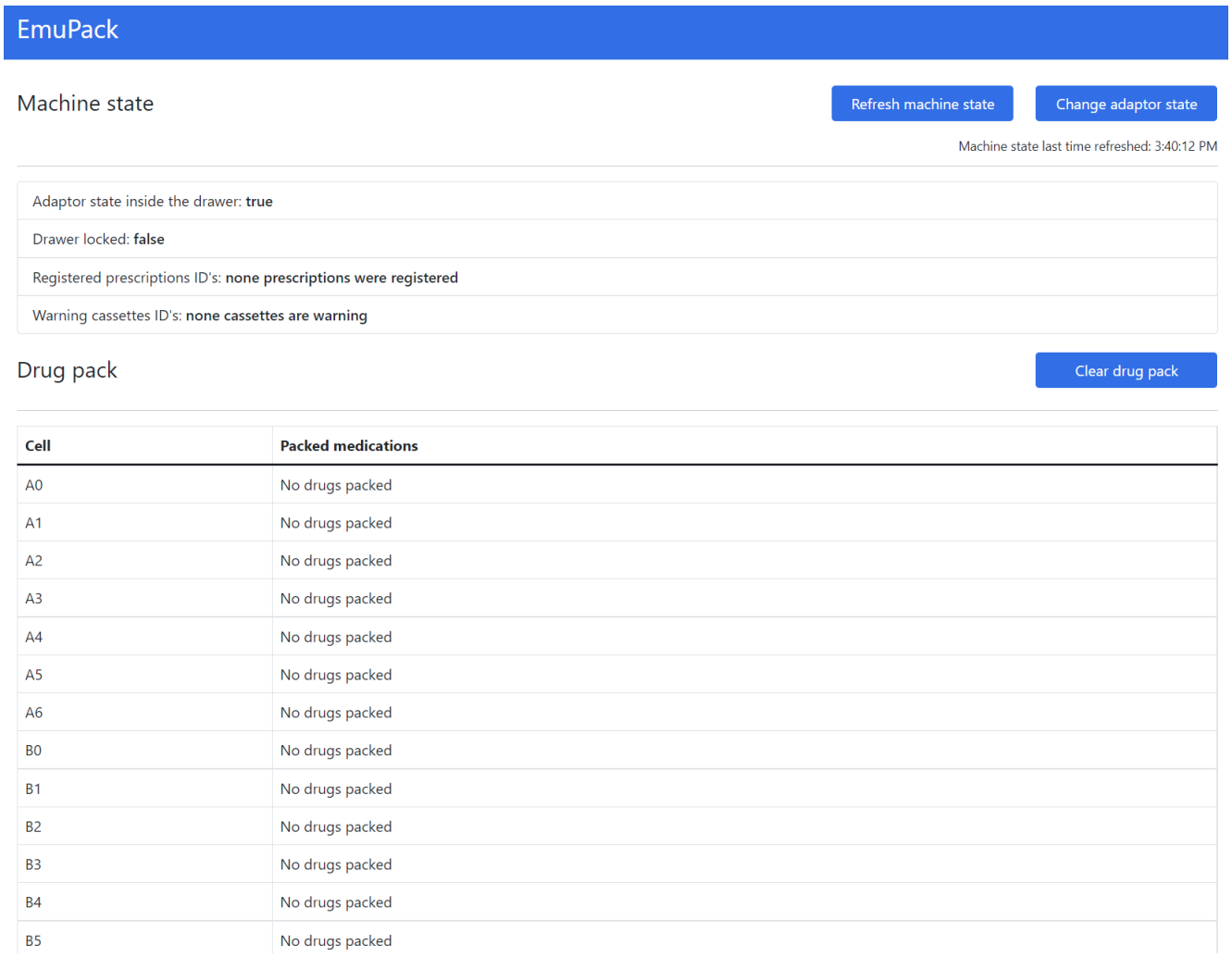


Рисунок 4.2.5.2 – сторінка із відображенням стану емулятора

На веб-сторінці містяться дані про поточний стан адаптера, блокування відсіку із упаковкою, колекція зареєстрованих рецептів та касет, що потребують на увагу. Також, присутня таблиця із співставленням комірок упаковки та їх вмісту. Серед інтерактивних елементів: кнопка оновлення стану, кнопка зміни статусу адаптеру (вийняти адаптер із відсіку), кнопка очищення упаковки. Якщо ж зареєстровано якийсь з рецептів, можна подивитися, які саме препарати були додані до нього (рисунок 4.2.5.3). Таблиця заповнених препаратами комірок зображена на рисунку 4.2.5.4.

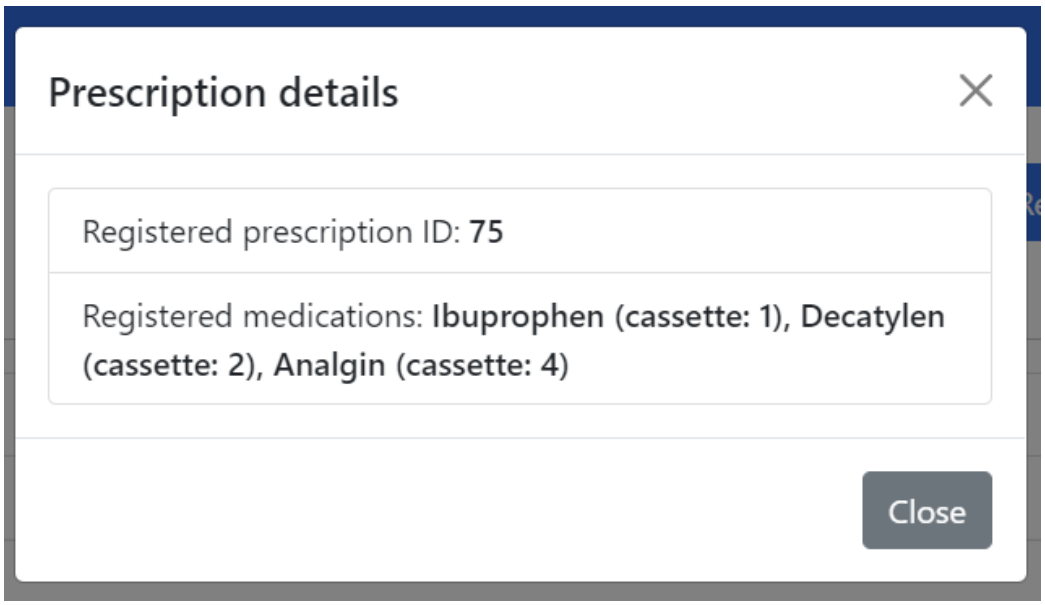


Рисунок 4.2.5.3 – вікно із деталями рецепту

Drug pack

Clear drug pack

Cell	Packed medications
A0	Decatylen (quantity: 3)
A1	Analgin (quantity: 3)
A2	No drugs packed
A3	No drugs packed
A4	No drugs packed
A5	No drugs packed
A6	No drugs packed
B0	Ibuprophen (quantity: 2)
B1	No drugs packed
B2	No drugs packed
B3	No drugs packed
B4	No drugs packed
B5	No drugs packed
B6	No drugs packed
C0	Decatylen (quantity: 4)

Рисунок 4.2.5.4 – таблиця упаковки із препаратами

4.2.6 Тестування ручного введення команд до емулятора

Для того, щоб переконатися у коректності роботи емулятора, спробуємо власноруч, а не за допомогою керуючого додатка, відправити низку команд до емулятора. Для відправлення команд до емулятора будемо використовувати консольний додаток, що є простим TCP-клієнтом, що лише відправляє введений текст до TCP-серверу.

Для перевірки правильності роботи функціоналу команди ініціалізації, візьмемо для початку машинний стан, що вже містить у собі інформацію від попередньої сесії роботи із машиною (рисунок 4.2.6.1). Зокрема, маємо зареєстрований рецепт під ідентифікатором “75”. Команду відправлено (рисунок 4.2.6.2). За інформацією з теоретичної частини, команда ініціалізації повинна скинути машину у початковий стан, тобто у даному випадку видалити зареєстровані рецепти. Тест підтвердив функціонал команди (рисунок 4.2.6.3)

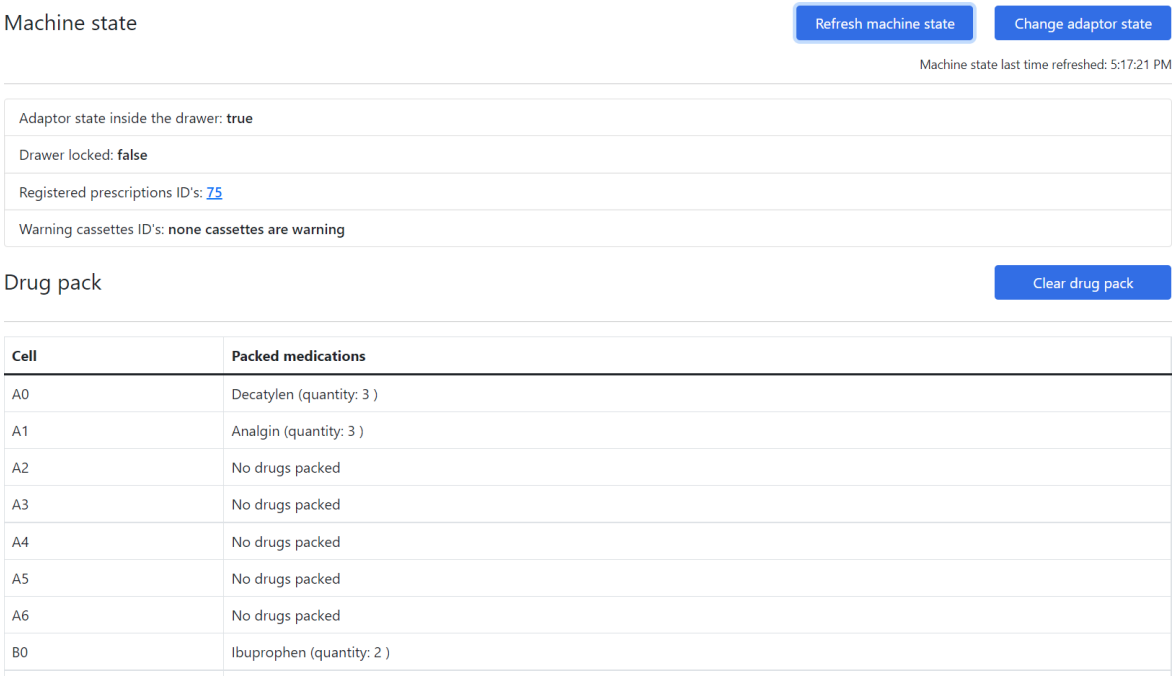


Рисунок 4.2.6.1 – стан машини на початок відправлення команд

```
Введіть ip-адресу машини: 127.0.0.1
Введіть мережевий порт машини: 8888

Адреса: 127.0.0.1; Порт: 8888

!!!!!!!!!!!!!!!ПОЧАТОК ВІДПРАВКИ КОМАНД ДО МАШИНИ!!!!!!!!!!!!!!!
INC1M100000
Отримано повідомлення: INM1C10000200
```

Рисунок 4.2.6.2 – відправлення команди та відповідь емулятора на неї

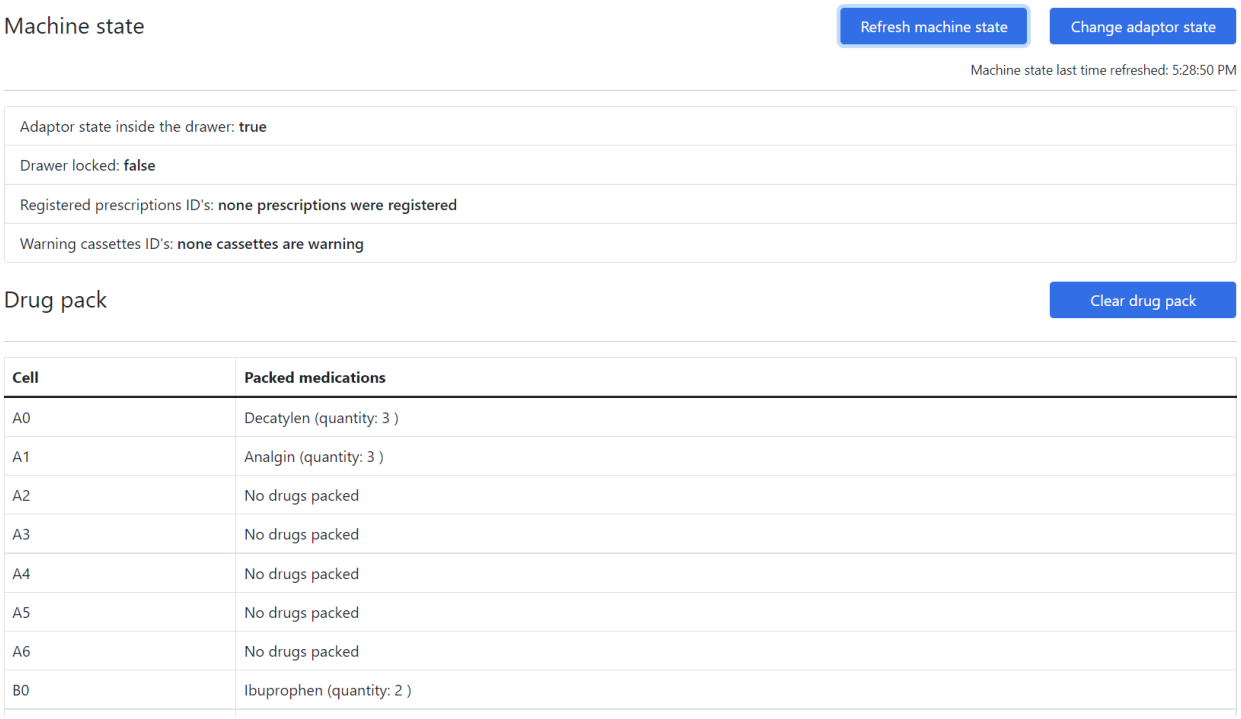


Рисунок 4.2.6.3 – стан емулятора після ініціалізації

Для подальшого тестування натиснемо кнопку очищення упаковки та відправимо команду на реєстрацію рецепту під ідентифікатором “75”, яка зареєструє (рисунок 4.2.6.4):

— препарат “DECATYLEN” із касети “1” у кількості “5”;

— препарат “IBUPROPHEN” із касети “2” у кількості “10”;

```
PRC1M100081I00750201DECATYLEN                    0000502IBUPROPHEN                    00010
Отримано повідомлення: PRM1C10000200
```

Рисунок 4.2.6.4 – реєстрація рецепту

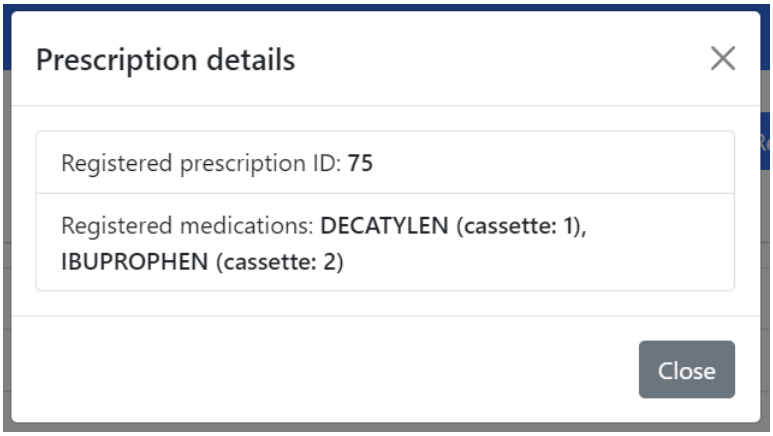


Рисунок 4.2.6.5 – результат реєстрації рецепту

Рецепт зареєстровано, команда спрацювала правильно (рисунок 4.2.6.5). Для подальшого пакування спробуємо відправити команду заповнення комірки “A1”, препаратом “DECATYLEN” у кількості “5” з касети “1”. Команда має бути відхилена валідацією так, як наразі розблокована дверця відсіку. Тобто, у відповіді на команду “FL” повинен міститися результуючий код “02” (додаток А, рисунок 4.2.6.6).

```
FLC1M100015P0075A1MC010105
Отримано повідомлення: FLM1C10000202
```

Рисунок 4.2.6.6 – невдалий результат виконання команди

Якщо ж заблокувати дверцю відсіку та спробувати відправити команду ще раз, відповідь буде містити код “00” (рисунок 4.2.6.7), тобто команда була виконана успішно (рисунок 4.2.6.8) і ми бачимо препарат в упаковці.

```
MRC1M100005D0000
Отримано повідомлення: MRM1C10000200
FLC1M100015P0075A1MC010105
Отримано повідомлення: FLM1C10000200
```

Рисунок 4.2.6.7 – блокування дверці відсіку та вдала відповідь на команду пакування

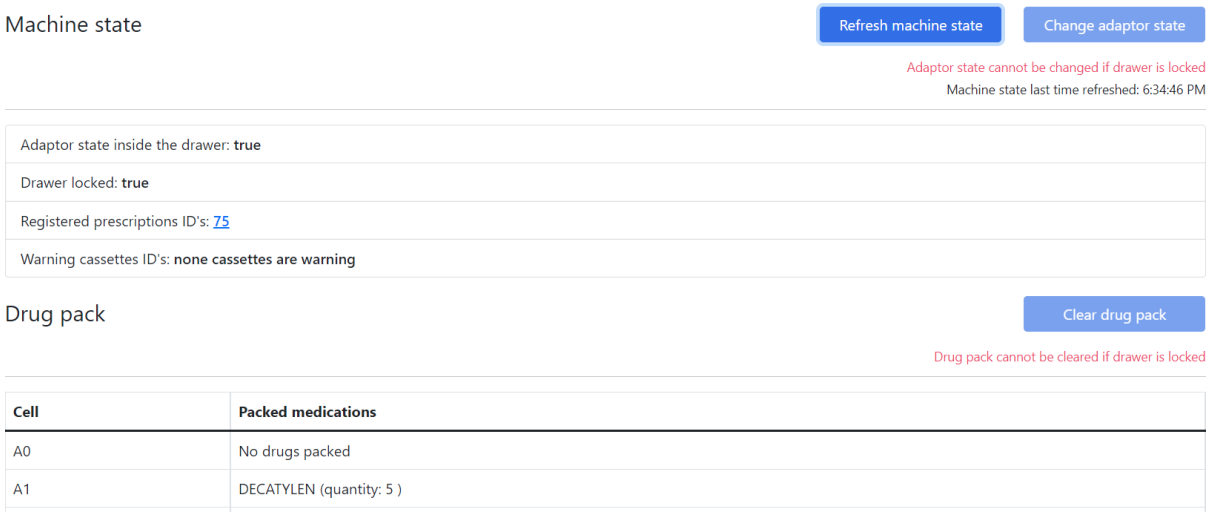


Рисунок 4.2.6.7 – результат виконання команди

4.3 Практична реалізація керуючого додатка

4.3.1 Реалізація команд

На відміну від емулятора, керуюче рішення безпосереднє формує команди. Як і у емуляторі, у ньому є базовий клас команди, від якого успадковуються всі інші реалізації (рисунок 4.3.1.1).

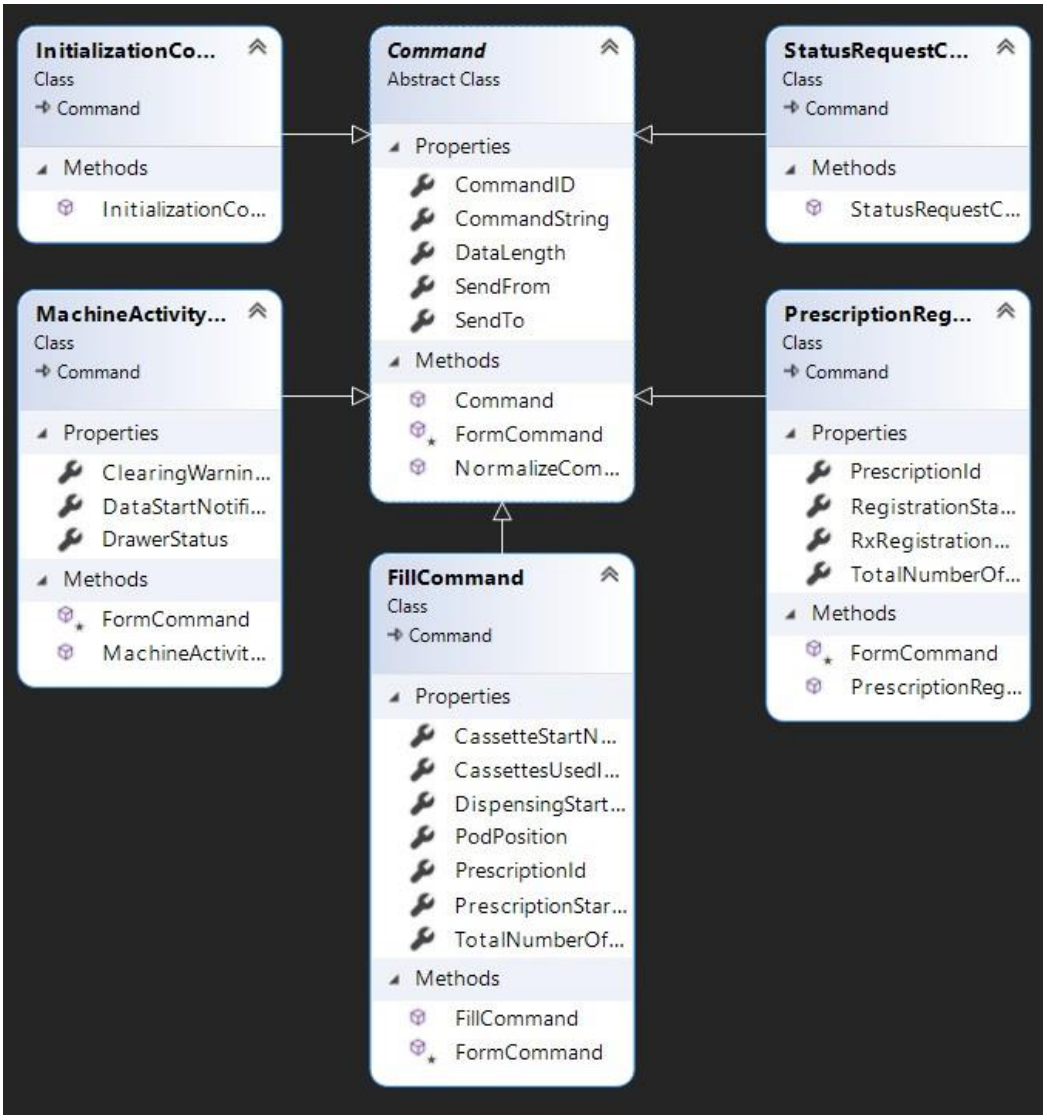


Рисунок 4.3.1.1 – діаграма класів команд

Формування команди відбувається згідно набору набору параметрів, що надходить у DTO [11].

DTO – практика, що являє собою передачу інформації між системами додатку у вигляді спеціального об’єкту, що має стан, але не має поведінки.

У випадку керуючого рішення, DTO – об’єкт, що формується користувачем у SPA клієнтському додатку, після чого потрапляє до описаних у теоретичній частині операцій. Саме операції, зробивши необхідні перетворення DTO, виконують дії по формуванню та відправленню команд. Діаграми реалізованих у додатку операцій можна побачити на рисунку 4.3.1.2.

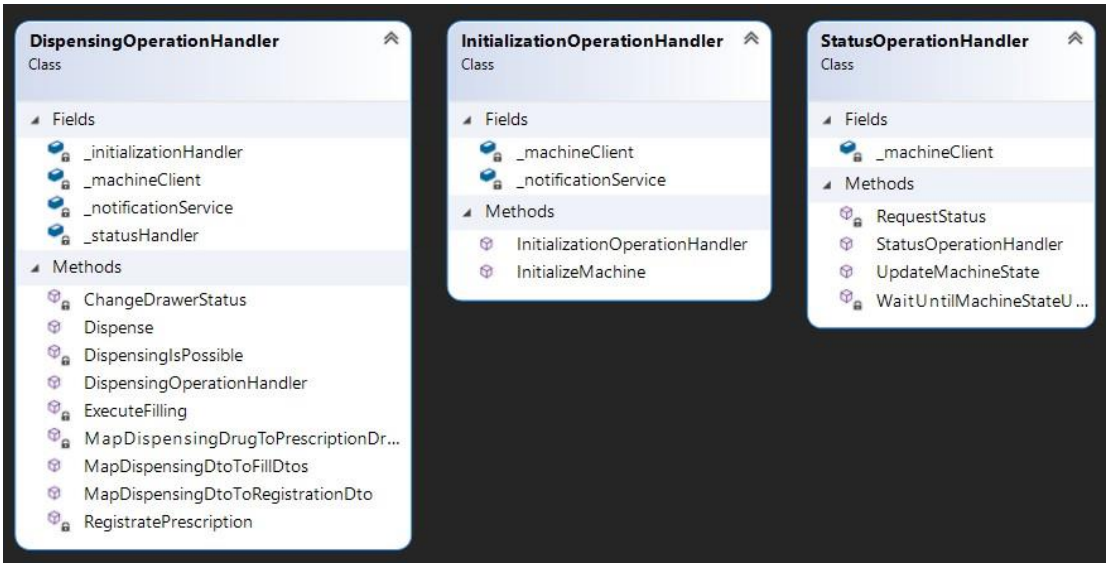


Рисунок 4.3.1.2 – діаграма класів команд

Кожна з операцій, у собі містить метод, у якому реалізується основний її функціонал. Наприклад, для “DispensingOperationHandler” це метод “Dispense” (лістинг 4.3.1.3). У ньому, виконуються дії по формуванню відповідних команд, виходячи з набору властивостей, що надходять у метод від контролеру (а раніше – від SPA). Але, робиться це лише у тому випадку, якщо пакування наразі можливе, тобто якщо метод “DispensingIsPossible” повертає значення “True” (лістинг 4.2.1.4). Слід зауважити, що у методах операції пакування можна побачити виклики інших операцій, зокрема: операції ініціалізації та операції отримання стану. Так, наприклад, операція ініціалізації викликається у тому випадку, коли машина не підключена до керуючого додатку. Починає та завершує операцію пакування – операція отримання статусу. Вона слугує для того, щоб оновити інформацію про стан емулятора зі сторони керуючого додатка до і після пакування. У свою чергу це робиться напочатку, щоб з’ясувати, чи можливе пакування і наприкінці, щоб записати необхідну для подальших пакувань інформацію (наприклад, про рецепт, що щойно був зареєстрований).

```
public void Dispense(DispensingOperationDTO dto)
{
    if (DispensingIsPossible(dto))
    {
        RegistratePrescription(MapDispensingDtoToRegistrationDto(dto));
        MapDispensingDtoToFillDtos(dto).ForEach(fillDto => ExecuteFilling(fillDto));
    }
    if (_machineClient.ConnectedToMachine)
    {
        ChangeDrawerStatus(drawerLocked: false);
        _statusHandler.UpdateMachineState();
    }
}
```

Лістинг 4.3.1.3 – головний метод пакувальної операції

```
private bool DispensingIsPossible(DispensingOperationDTO dto)
{
    bool adaptorInDrawer = true;
    bool prescriptionNotRegistred = true;
    if (!_machineClient.ConnectedToMachine)
    {
        _initializationHandler.InitializeMachine();
    }
    if (!_machineClient.ConnectedToMachine)
    {
        return false;
    }
    _statusHandler.UpdateMachineState();
    if (_machineClient.MachineState.DrawerOpened)
    {
        ChangeDrawerStatus(drawerLocked: true);
    }
    if (!_machineClient.MachineState.AdaptorInDrawer)
    {
        adaptorInDrawer = false;
    }
    if (_machineClient.MachineState.RegistredPrescriptionsIds
        .Contains(dto.PrescriptionId.ToString()))
    {
        prescriptionNotRegistred = false;
    }
    _notificationService.SendDispensingNotification(adaptorInDrawer,
        prescriptionNotRegistred);

    return adaptorInDrawer && prescriptionNotRegistred;
}
```

Лістинг 4.3.1.4 – метод, що визначає можливість пакування у даний момент часу

4.3.2 Тестування автоматичної генерації команд

Кінцева точка керування емулятору – клієнтській SPA додаток, на його інтерфейсі розміщена форма введення замовлення. У неї фармацевт вказує номер рецепту та касети із препаратами, з яких буде вестися пакування. Інтерфейс підтримує введення одразу багатьох касет із вказанням комірок упаковки, до яких буде доставлено препарати. На рисунку 4.3.2.1 зображено введення замовлення, що буде пакуватися одразу з двох касет у різні комірки.

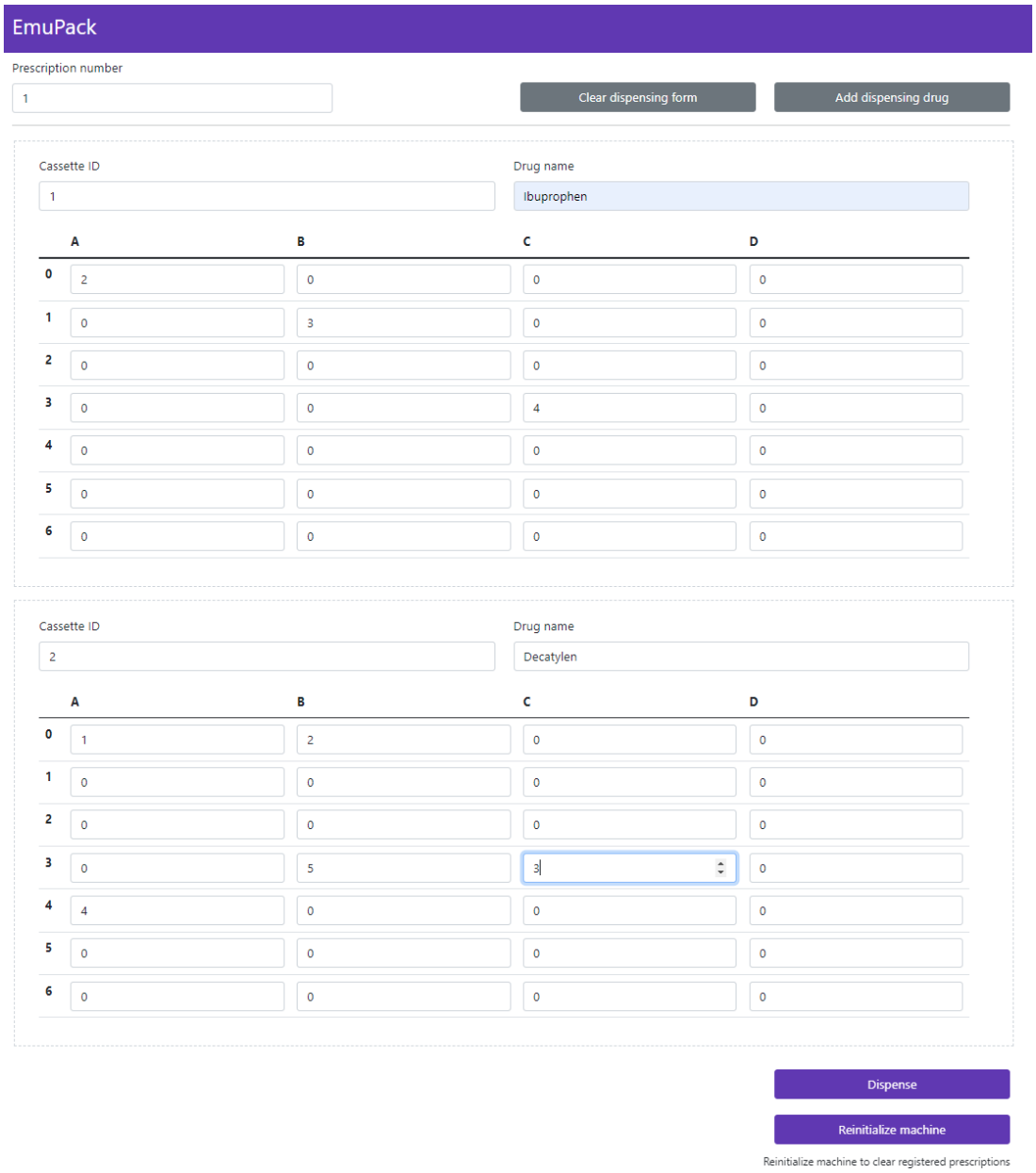


Рисунок 4.3.2.1 – інтерфейс введення замовлення

Форма введення підтримує валідацію усіх полів та обробку повідомлень про помилки, що надходять із серверу. На сторінці також розміщена кнопка для очищення форми та кнопка, що дозволяє виконати ініціалізацію машини у випадку, коли, наприклад, потрібно скинути зареєстровані рецепти.

Тест на виконання замовлення надає можливості на перевірку всього комплексу. Дані введення будуть трансформовані у DTO, що передається до серверу та находить у операцію пакування, внутрішня будови якої була розібрана на рисунку 3.7. Важливо звернути увагу на один з інструментів, що допомагає у тесті. Емулятор підтримує логування команд, що надходять та відправляються. Тому всі команди, що будуть відправлені до емулятора пакувальною операцією, тобто керуючим сайтом, ми побачимо у лог-файлі та зможемо перевірити. Згідно із початковим станом емулятора (додаток В), теоретичний порядок команд має представляти з себе надходження:

1) команду ініціалізації;

2) статусну команду

3) команду запиту машинної активності на блокування дверці

4) команду реєстрації рецепту

5) перелік команд заповнення комірок упаковки

6) команду запиту машинної активності на розблокування дверці

7) статусну команду

Під час тесту буде виконано пакування рецепту під ідентифікатором “10”, який запакує у комірку “A0” препарат “Decatylen” з касети “1”, у кількості “5” таблеток та препарат “Ibuprophen” з касети “2” у кількості “4” таблетки. У комірку “A3” буде запаковано “3” таблетки препарату “Decatylen”. У комірку “B0” буде запаковано “5” таблеток препарату “Ibuprophen” (додаток Г).

Зареєстрований рецепт можна бачити на рисунку 4.3.2.2. Результат пакування зображений на рисунку 4.3.2.3. Вміст лог-файлу із виконаними командами можна бачити на рисунку 4.3.2.4.

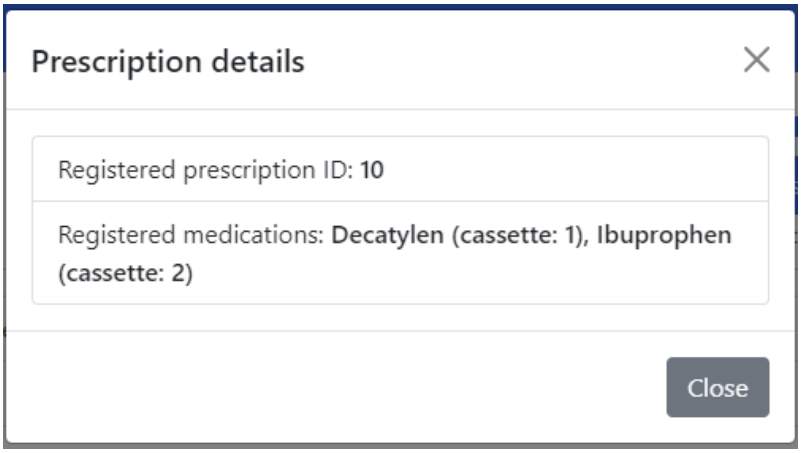


Рисунок 4.3.2.2 – зареєстрований рецепт

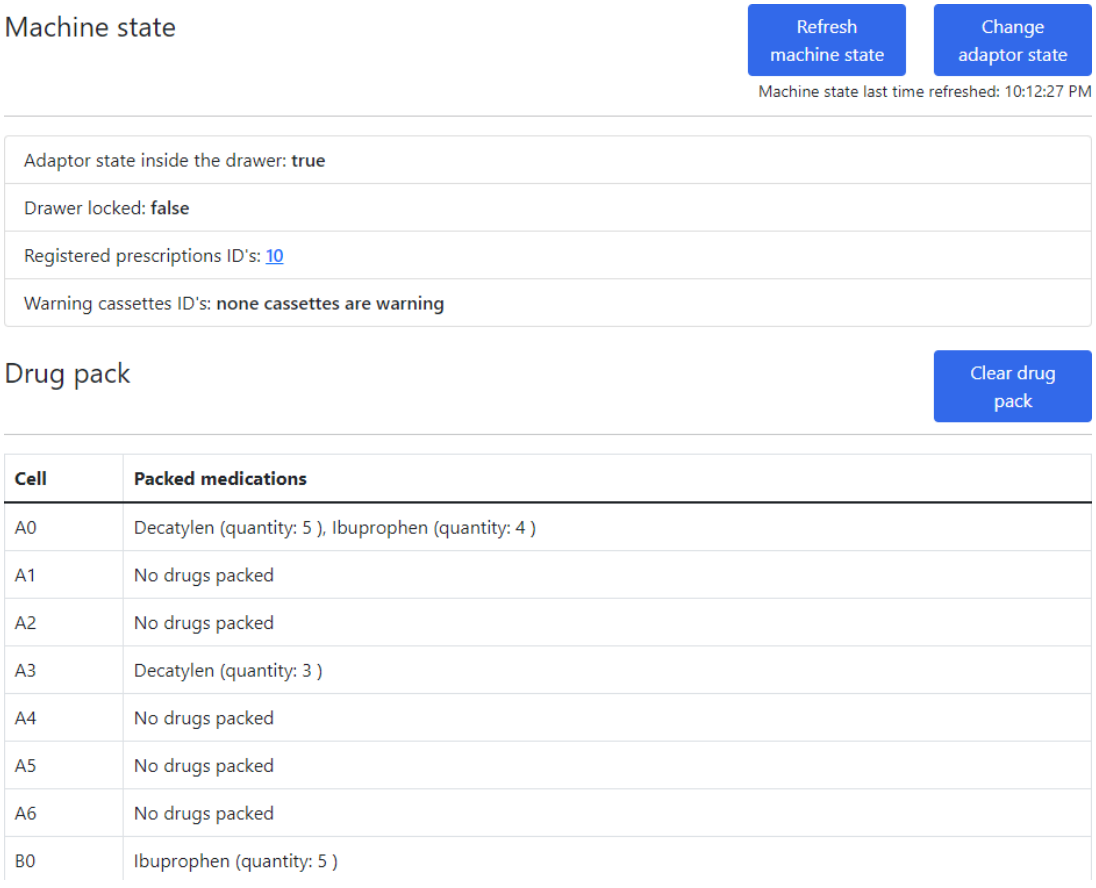


Рисунок 4.3.2.3 – результат пакування

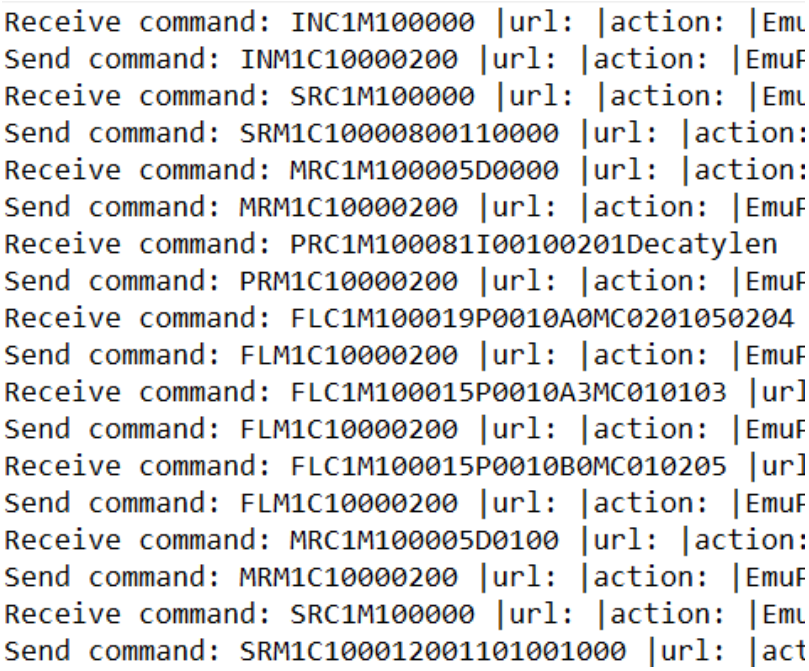


Рисунок 4.3.2.4 – вміст лог-файлу

Таким чином, цим тестом підтверджена правильна взаємодія компонентів комплексу. Керуючий додаток, згідно із функціональними потребами провів успішне генерування команд. Емулятор реагував на надходження повідомлень очікуваним чином. Результуючий набір команд з лог-файлу на рисунку 4.3.2.4 збігся із теоретичним розрахунком. Виходячи з вхідних даних тесту, параметри команд були коректно підставлені, а очікуваний результат тесту збігся із фактичним.

ВИСНОВКИ

У якості засобу для спрощення інтеграції фармацевтичного робота пакувальника може використовуватися розроблений комплекс емулятора та керуючого додатка.

Комплекс повторює ключові особливості роботи із реальними машинами. У рішенні представлено розділення кінцевої системи на аналог реальної машини та додаток із функцією керування. Як і у існуючих роботах, комунікація із машиною відбувається через мережеве з'єднання, засобами розробленого протоколу комунікації.

Протокол комунікації – набір текстових команд, до розпізнавання яких здатний емулятор робота-пакувальника. Набір команд є цілком достатнім для реалізації операції пакування та її гнучкої конфігурації, відповідно до замовлення.

Автоматична генерація команд, представлена у керуючому рішенні, надає можливості для фармацевтів абстрагуватися від поняття команди та мати справу із сутностями, що не потребують ознайомлення із технічними деталями.

Використання командного протоколу, подібного реальним машинам, реалізація принципів валідації, повторення аспекту комунікації між складовими комплексу та логування надає додаткові можливості розробникам, що працюють над інтеграційними рішеннями. Емулятор допомагає у досліджені кінцевої системи та отриманні конкретних прикладів з предметної області для подальших розробок.

Зрозуміла індикація на інтерфейсах та візуалізація процесу слугує демонстрацією для керівництва організацій, що мають наміри у встановленні фармацевтичних роботів.

Фармацевти, що вперше мають справу із роботами-пакувальниками отримують комплекс, у якому промодельовані ключові комплектуючі та

процеси у реальних машинах. Зокрема, є можливість розібратися із алгоритмом користування технікою та розібратися із такими поняттями, як рецепти, касети, адаптери, тощо.

Вибрана архітектура є вдалим рішенням через комбінування TCP серверу (клієнту) та REST API у одному додатку. З одного боку, емулятор та керуючий додаток можуть спілкуватися, використовуючи протокол HTTP, з іншого – здатні до роботи на основі розробленого протоколу повідомлень. Використання архітектурного стилю REST API дозволяє легко поєднати серверну частину обох додатків із клієнтською та використовувати SPA, що э гнучким інструментом для побудови, наприклад, динамічних форм, якою у випадку проекту є форма введення замовлення.

ПЕРЕЛІК ПОСИЛАНЬ

1. Yuyama pharmacy automation manufacterer, Litrea III [Електроний ресурс], Доступ до ресурсу: https://www.yuyamarx.com/litrea%e2%85%a2112/

4. Yuyama pharmacy automation manufacturer, FAQ [Електронний ресурс], Доступ до ресурсу: https://yuyama.com.hk/en/faqs.php

3. Omnicell product equipment specification, FAQ [Електронний ресурс] стр.13, Доступ до ресурсу: http://www.partnershipsbc.ca/wp/wp-content/uploads/2019/09/Omnicell-Product-Equipment-Specifications.pdf

4. CISCO OSI Model Reference Chart, FAQ [Електронний ресурс], Доступ до ресурсу: https://learningnetwork.cisco.com/s/article/osi-model-reference-chart

5. Teach computer science, client-server architecture [Електроний ресурс], Доступ до ресурсу: https://teachcomputerscience.com/client-server-architecture/

6. Red hat, REST API [Електроний ресурс], Доступ до ресурсу: https://www.redhat.com/en/topics/api/what-is-a-rest-api

7. MDN web doc, SPA (Single-page application) [Електроний ресурс], Доступ до ресурсу: https://developer.mozilla.org/en-US/docs/Glossary/SPA

8. MSDN docs, ASP.NET Core overview [Електроний ресурс] , Доступ до ресурсу: https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0

9. MSDN docs, System.Net.Scokets [Електроний ресурс] , Доступ до ресурсу: https://docs.microsoft.com/en-us/dotnet/api/system.net.sockets?view=net-6.0
10. Angular framework [Електроний ресурс], Доступ до ресурсу: https://angular.io/
11. Okta Identity platform, DTO defenition [Електроний ресурс], Доступ до ресурсу: https://www.okta.com/identity-101/dto/
11. Okta Identity platform, DTO defenition [Електроний ресурс], Доступ до ресурсу: https://www.okta.com/identity-101/dto/
12. Anacleto T. A. et al. Medication errors and drug-dispensing systems in a hospital pharmacy //Clinics. – 2005. – Т. 60. – №. 4. – С. 325-332.
13. Fitzpatrick R. et al. Evaluation of an automated dispensing system in a hospital pharmacy dispensary //The pharmaceutical journal. – 2005. – Т. 274.
14. Tsao N. W. et al. Decentralized automated dispensing devices: systematic review of clinical and economic impacts in hospitals //The Canadian journal of hospital pharmacy. – 2014. – Т. 67. – №. 2. – С. 138.
15. Klibanov O. M., Eckel S. F. Effects of automated dispensing on inventory control, billing, workload, and potential for medication errors //American journal of health-system pharmacy. – 2003. – Т. 60. – №. 6.
16. Rochais E. et al. Nursing perception of the impact of automated dispensing cabinets on patient safety and ergonomics in a teaching health care center //Journal of pharmacy practice. – 2014. – Т. 27. – №. 2. – С. 150-157.
17. Noparatayaporn P. et al. Comparison on human resource requirement between manual and automated dispensing systems //Value in health regional issues. – 2017. – Т. 12. – С. 107-111.

18. Uy R. C. Y., Kury F. P., Fontelo P. A. The state and trends of barcode, RFID, biometric and pharmacy automation technologies in US hospitals //AMIA annual symposium proceedings. – American Medical Informatics Association, 2015. – Т. 2015. – С. 1242.

19. Felkey B. G., Fox B. I. Pharmacy automation and technology //Hospital Pharmacy. – 2015. – Т. 44. – №. 8. – С. 708-710.

20. Wang H. et al. Collation delay optimization using discrete event simulation in mail-order pharmacy automation systems //Proceedings of the 2016 Industrial and Systems Engineering Research Conference. – 2016.

Додаток А

Протокол комунікації із емулятором EmuPack

EmuPack communication protocol

Command	Index
Initialization command	IN
Fill command	FL
Prescription registration command	PR
Machine activity request command	MR
Status request command	SR
Not recognized request command	NR

"IN" Initialization command request (PC to Machine)

Field	Detail	Length
Command ID	IN	2
Send from	C1	2
Send to	M1	2
Data length	00000 - 99999	5

"IN" Initialization command response (PC to Machine)

Field	Detail	Length	
Command ID	IN	2	
Send from	M1	2	
Send to	C1	2	
Data length	00000 - 99999	5	
Response code	00, 01	2	00 : Success 01 : Failed. Wrong command format

"FL" Fill command request (PC to Machine)

Field	Detail	Length
Command ID	FL	2
Send from	C1	2
Send to	M1	2
Data length	00000 - 99999	5
Prescription information part	P	7
Dispensing information field	M	8
Prescription information part		
Start notification	P	1
Prescription ID	0000-9999	4
Pod position	A0 - A6 B0 - B6 CO - C6 D0 - D6	2
Dispensing information field		
Start notification	M	1
Cassette part start notification	C	1
Total number of cassettes	01-40	2
Cassette ID	01-40	2
Qty of drug	01-99	2

"FL" Fill command response (Machine to PC)

Field	Detail	Length	
Command ID	FL	2	
Send from	M1	2	
Send to	C1	2	
Data length	00000 - 99999	5	
Response code	00 - 02	2	00 : Success 01 : Failed. Wrong command format 02 : Failed, Data is not valid for prescription

"PR" Prescription registration command request (PC to Machine)

Field	Detail	Length
Command ID	PR	2
Send from	C1	2
Send to	M1	2
Data length	00000 - 99999	5
Registration information part	I	44

Registration information part

Start notification	I	1
Prescription ID	0000-9999	4
Total number of registred cassettes	01-40	2
Cassette ID	01-40	2
Drug name	Character string	30
Quantity per cassette	00001 - 99999	5

"PR" Prescription registration command response (Machine to PC)

Field	Detail	Length	
Command ID	PR	2	
Send from	M1	2	
Send to	C1	2	
Data length	00000 - 99999	5	00 : Success
Response code	00 - 02	2	01 : Failed. Wrong command format 02 : Failed. Data is not valid for the prescription

"MR" Machine activity request command (PC to Machine)

Field	Detail	Length
Command ID	MR	2
Send from	C1	2
Send to	M1	2
Data length	00000 - 99999	5
Machine activity data part	D	5

Machine activity data part

Start notification	D	1	
Drawer status	00 - 01	2	00 : Lock drawer 01 : Unlock drawer
Clearing warnings initiated	00 - 01	2	00: Clearing not initated 01: Clearing initated

"MR" Machine activity request command response (Machine to PC)

Field	Detail	Length	
Command ID	MR	2	
Send from	M1	2	
Send to	C1	2	
Data length	00000 - 99999	5	
Response code	00 - 02	2	00 : Success 01 : Failed. Wrong command format

"SR" Status request command (PC to Machine)

Field	Detail	Length
Command ID	SR	2
Send from	C1	2
Send to	M1	2
Data length	00000 - 99999	5

"SR" Status request command response (Machine to PC)

Field	Detail	Length	
Command ID	SR	2	
Send from	M1	2	
Send to	C1	2	
Data length	00000 - 99999	5	
Response code	00, 01	2	00 : Success 01 : Failed. Wrong command format
Drawer status	0, 1	1	0 : Locked 1 : Unlocked
Adaptor status	0, 1	1	0 : No adaptor 1 : In drawer
Warning cassettes quantity	00 - 40	2	
Cassette ID	01 - 40	2	

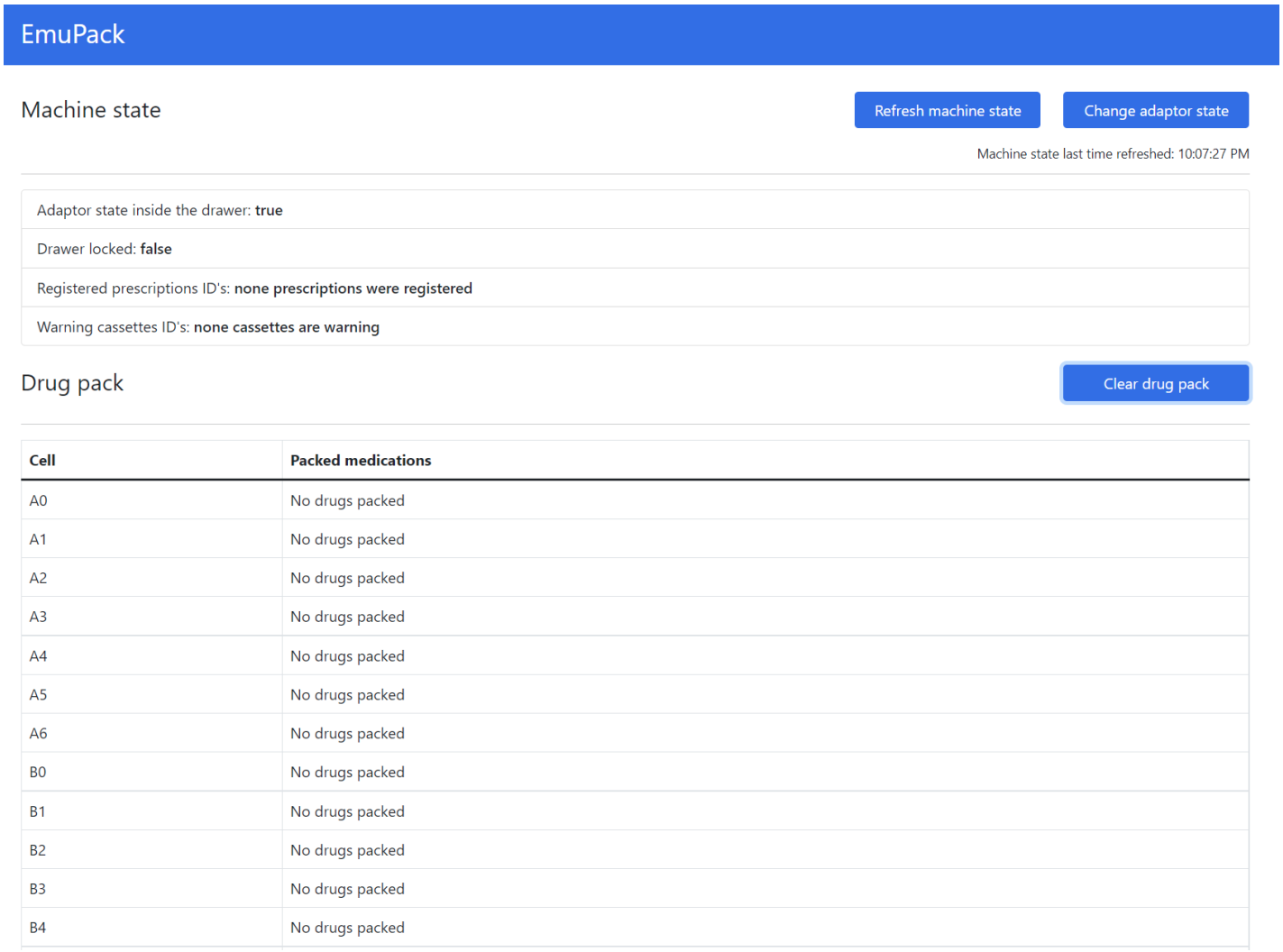
"NR" Not recognized request command response (Machine to PC)

Field	Detail	Length
Command ID	NR	2
Send from	M1	2
Send to	C1	2
Data length	00000 - 99999	5
Response code	00	2

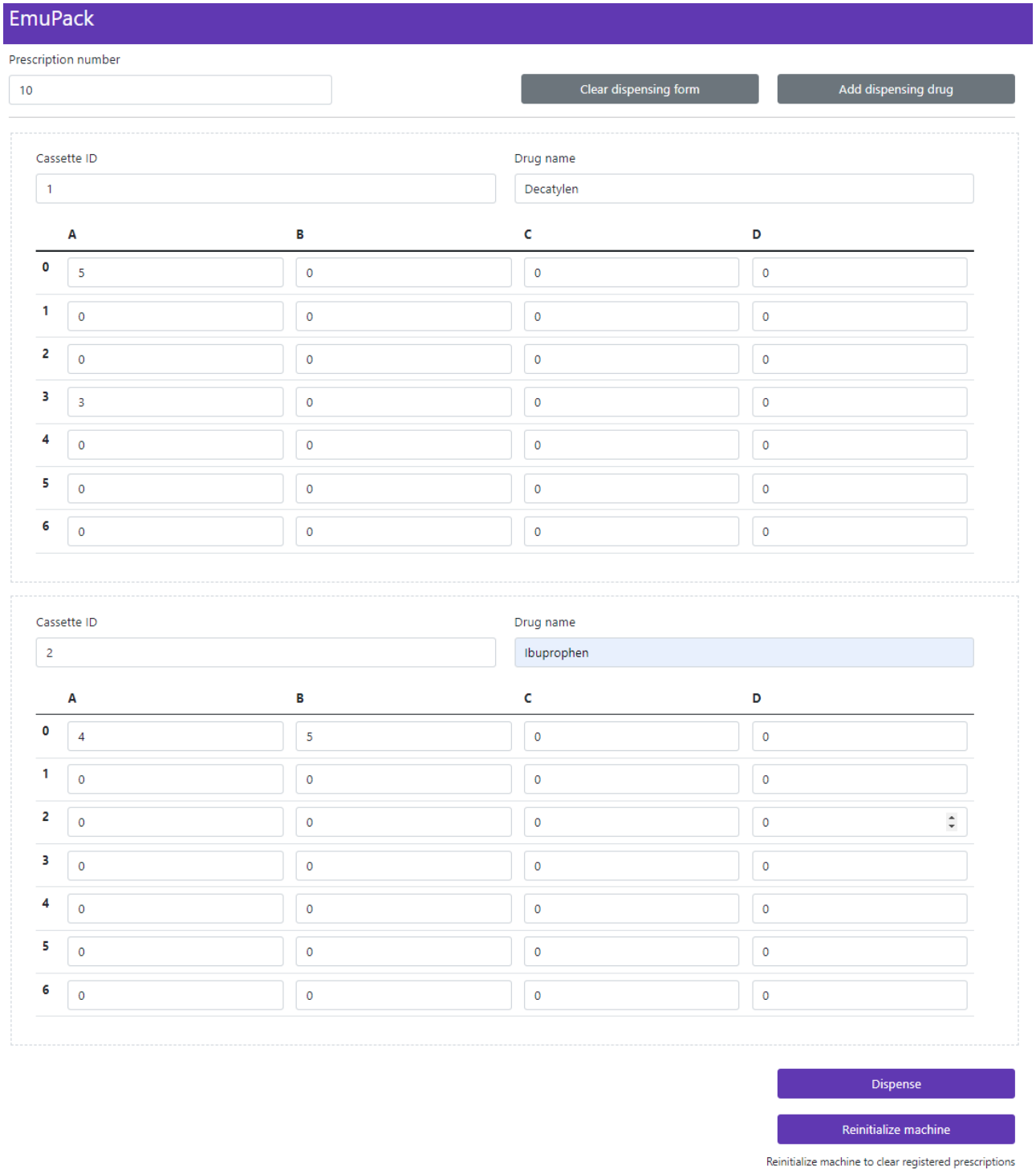
Додаток Б

Посилання на GitHub репозиторій коду проекту: https://github.com/demid-dev/EmuPack.Complex

Додаток В


EmuPack
Machine state
Refresh machine state
Change adaptor state
Machine state last time refreshed: 10:07:27 PM
Adaptor state inside the drawer: true
Drawer locked: false
Registered prescriptions ID's: none prescriptions were registered
Warning cassettes ID's: none cassettes are warning
Drug pack
Clear drug pack
Cell
Packed medications
A0 No drugs packed
A1 No drugs packed
A2 No drugs packed
A3 No drugs packed
A4 No drugs packed
A5 No drugs packed
A6 No drugs packed
B0 No drugs packed
B1 No drugs packed
B2 No drugs packed
B3 No drugs packed
B4 No drugs packed

Додаток Г


EmuPack
Prescription number
10
Clear dispensing form
Add dispensing drug
Cassette ID
1
Drug name
Decatylen
A B C D
0 5 0 0 0
1 0 0 0 0
2 0 0 0 0
3 3 0 0 0
4 0 0 0 0
5 0 0 0 0
6 0 0 0 0
Cassette ID
2
Drug name
Ibuprophen
A B C D
0 4 5 0 0
1 0 0 0 0
2 0 0 0 0
3 0 0 0 0
4 0 0 0 0
5 0 0 0 0
6 0 0 0 0
Dispense
Reinitialize machine
Reinitialize machine to clear registered prescriptions