Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

УДК 004.942

На правах рукопису

ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Тема: “ Система захисту від реалізації підробок в аптеках”

(назва згідно з наказом ректора)

Спеціальність – 121 “Інженерія програмного забезпечення”

ПОЯСНЮВАЛЬНА ЗАПИСКА

БР.ІПЗ - 31.00.00.000

(позначення)

Студентка

ІПЗ-43_____/Тетяна КУЗІВ/

(шифр групи) (підпис) (дата)(розшифровка підпису)

Науковий керівник

д.т.н._____/Генадій ПОРЄВ/

(посада) (підпис) (дата) (розшифровка підпису)

Допускається до захисту

Консультант

з питань нормоконтролю

фахівець_____/Тамара ЧАПОВСЬКА/

(посада) (підпис) (дата) (розшифровка підпису)

Завідувач кафедри

д.т.н., проф._____/Олексій БИЧКОВ/

(посада) (підпис) (дата) (розшифровка підпису)

Київ – 2021

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра програмних систем і
технологій Освітньо-кваліфікаційний
рівень бакалавр
Спеціальність 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖЕНО

Зав. кафедри програмних систем і технологій

_____ (Олексій БИЧКОВ)

(підпис) (прізвище та ініціали)

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУРОБОТУ СТУДЕНТУ

Кузів Тетяні Романівні

(прізвище, ім’я, по-батькові)

1. Тема бакалаврської роботи “ Система захисту від реалізації підробок в аптеках”

керівник проекту (роботи) Порєв Генадій Володимирович, д.т.н., с.н.с, доцент

затверджені наказом вищого навчального закладу від “11” листопада 2020 р. №

2. Строк подання студентом роботи _____

3. Вихідні дані до проекту (роботи) Система захисту від реалізації підробок в аптеках

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)

1. Аналіз існуючих рішень проблеми фальсифікації ліків
2. Порівняння штрих-кодів
3. Розгляд двовимірного штрих-коду DataMatrix та його впровадження в маркування ліків
4. Створення бази даних
5. Розробка модулів системи
6. Реалізація API

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

1. Директива ЄС DataMatrix(рис. 1.1, ст. 14)
2. GS1 DataMatrix USA (рис. 1.2, ст. 16)
3. Лінійні штрих-коди (рис. 1.3, ст. 20)
4. DataMatrix (рис. 1.4, ст. 21)
5. QR-код (рис. 1.5, ст. 21)
6. Aztec (рис. 1.6, ст. 22)
7. Модель Finder і дані(рис 2.1, ст. 24)
8. Квадратна форма проти прямокутної форми (рис. 2.2, ст. 25)
9. Складові штрих-коду(рис 2.3, ст. 27)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняла
Основна частина	Доцент, д.т.н., с.н.с. Порєв Г.В.		

7. **Дата видачі завдання** 11.11.2020

Керівник_____(Генадій ПОРЄВ)

Завдання прийняла до виконання _____ (Тетяна КУЗІВ)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури	25.11.2020	виконано
2	Аналіз існуючих рішень проблеми	09.01.2021	виконано
3	Порівняння штрих-кодів	20.01.2021	виконано
4	Розгляд двовимірного штрих-коду DataMatrix та його впровадження в	03.02.2021	виконано

	маркування ліків		
5	Розробка модулів системи	17.02.2021	виконано
6	Створення бази даних	10.03.2021	виконано
7	Реалізація API	31.03.2021	виконано
8	Опис розробки в пояснювальній записці	12.04.2021	виконано
9	Затвердження пояснювальної записки роботи завідувачем кафедри		

Студент – бакалавр_____(Тетяна КУЗІВ)

Керівник роботи_____(Генадій ПОРЄВ)

АНОТАЦІЯ

Випускна кваліфікаційна бакалаврська робота: 69 с., 20 рис., 1 табл., 14 додат., 14 джерел.

Тема: Система захисту від реалізації підробок в аптеках

Об'єкт дослідження : проблема фальсифікації ліків

Мета роботи: розробка системи захисту від реалізації підробок в аптеках. Система повинна передбачати використання однозначних ідентифікаційних ключів для ідентифікації ліків.

Результати дослідження: Досліджено можливості захисту від реалізації фальсифікації ліків. Запропоновано використання реалізованої системи для вирішення проблеми.

Висновок

В результаті розробки, було створено систему, яка дозволяє захистити від продажу підробок в аптеках за допомогою двовимірних штрих-кодів, які включають в себе унікальний ідентифікатор кожної упаковки ліків.

ANNOTATION

Graduation qualifying bachelor's thesis: 69 c., 20 pic., 1 table, 14 addit., 14 sources.

Topic: System of protection against sale of counterfeits in drugstores

Object of research : the problem of falsification of medicines

The goal of thesis: development of a system of protection against the sale of counterfeits in pharmacies. The system should provide for the use of unique identification keys to identify medicines.

Results of the research: Possibilities of protection against falsification of medicines are investigated. It is proposed to use the implemented system to solve the problem.

Conclusion

As a result of the development, the system was created. It protects against the sale of counterfeits in pharmacies with two-dimensional barcodes, which include a unique identifier for each package of medicines.

ЗМІСТ

Стр.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ - програмне забезпечення

БД - база даних

ТЗ - технічне завдання

GTIN- Global Trade Item Number

ЄС - Європейський Союз

ШК - Штрих-код

СУБД - Система управління базами даних

ІС - Інформаційна система

ВСТУП

Актуальність роботи

За інформацією Всесвітнього конгресу з фармації та фармацевтичних наук FIP, проблема підроблених ліків є глобальною по всьому світу.

Ключові факти :

- Неякісні та сфальсифіковані медичні вироби можуть завдати шкоди пацієнтам та не лікувати хвороби, для яких вони призначені.
- Вони призводять до втрати довіри до ліків, медичних працівників та систем охорони здоров'я.
- ВООЗ повідомляли про неякісні та фальсифіковані медичні вироби з усіх основних терапевтичних категорій, включаючи ліки та вакцини.
- Протималярійні засоби та антибіотики є одними з засобів, про які найбільш часто повідомляється як неякісні та фальсифіковані медичні вироби.
- Як загальні, так і інноваційні ліки можна фальсифікувати, починаючи від дуже дорогих ліків проти раку і закінчуючи недорогими ліками для зняття болю.
- Їх можна знайти на нелегальних вуличних ринках, через нерегульовані веб-сайти до аптек та лікарень.
- За оцінками, 1 із 10 медичних виробів у країнах з низьким та середнім рівнем доходу є неякісними або фальсифікованими.
- Неякісні та фальсифіковані медичні вироби сприяють протимікробній стійкості та інфекціям, стійким до лікарських засобів.

Сфальсифіковані медичні вироби не можуть містити діючої речовини, неправильної діючої речовини або неправильної кількості правильної діючої речовини. Також виявлено, що вони зазвичай містять кукурудзяний крохмаль, картопляний крохмаль або крейду. Деякі неякісні та сфальсифіковані медичні вироби були токсичними за своєю суттю, або з фатальним вмістом неправильної діючої речовини, або з іншими токсичними хімічними речовинами.

Неякісні та фальсифіковані медичні вироби за своєю природою важко виявити. Вони часто розроблені таким чином, щоб виглядати ідентично справжньому продукту, і можуть не викликати явної побічної реакції, однак вони часто не можуть належним чином лікувати хворобу або стан, для яких вони були призначені, і можуть призвести до серйозних наслідків для здоров'я, включаючи смерть.

На мою думку, найкращим способом, як насправді захиститися від підробок в майбутньому - це створити систему, яка буде вирішувати це питання швидко і ефективно.

Мета і задачі дослідження

Метою бакалаврської роботи є розробка системи захисту від реалізації підробок в аптеках. Система повинна передбачати використання однозначних ідентифікаційних ключів для ідентифікації ліків.

Досягнення мети включало розв'язання таких задач:

1. Ознайомлення з проблемою фальсифікації ліків і існуючими розв`язаннями її.
2. Ознайомлення з принципом роботи штрих-кодів та їх різновидами.
3. Ознайомлення з видами архітектури систем
4. Ознайомлення з видами баз даних
5. Ознайомлення з штрих-кодом DataMatrix, його складовими і принципом його створення.
6. Написання модулів системи
7. Створення бази даних.
8. Реалізація API
9. Тестування системи

Методи дослідження : пошук та аналіз інформації, дослідження існуючих вирішень проблеми фальсифікації ліків, порівняння видів штрих-кодів та принципами їх створення.

Практичне значення одержаних результатів : розроблену систему можна використовувати у цілях захисту від підробок.

Особистий внесок студента

Розроблена система, яка дозволяє захистити від продажу підробок в аптеках за допомогою двовимірного штрих-коду DataMatrix, які включають в себе унікальний ідентифікатор кожної упаковки ліків.

Публікації

За результатами наукових досліджень, проведених в бакалаврській роботі, було підготовлено статтю для доповіді у восьмій Східно-Європейської конференції «Математичні і програмні технології Internet of Everything» на тему: «Система захисту від реалізації підробок в аптеках»

Структура та обсяг роботи

Робота викладена на 69 сторінках друкованого тексту, який складається із вступу, трьох розділів, висновків, списку використаних джерел (14 найменувань). Робота містить 1 таблицю, 20 рисунків та 16 додатки,.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд існуючих вирішень проблеми фальсифікації ліків

Вирішення проблем з підробленими ліками являється однією з ключових завдань для багатьох країн, тому держава повинна забезпечувати громадян якісним засобом перевірки препаратів. Вже кілька років законодавці рішуче виступають проти небезпеки фальсифікованих ліків для споживачів та компаній, приймаючи національні та наднаціональні директиви щодо фальсифікації. Швидке дотримання вимог дозволяє фармацевтичним виробникам та постачальникам пакувальних послуг гарантувати, що їх продукція продовжує безперешкодно розподілятися, а підроблені недорогі товари надійно виводяться з ринку. В даний час існує більше 20 директив про фальсифікацію на різних етапах впровадження, з частково різними вимогами.

У 2011 році була опублікована Директива ЄС про фальсифіковані ліки 2011/62 / ЄС. Це поширюється на всі ліки, що відпускаються за рецептом та до відбірних безрецептурних препаратів в Європейському Союзі з лютого 2019 року. На країни, що не входять до ЄС, Ісландію та Норвегію також поширюється дія Європейської директиви про фальсифіковані ліки. Італія, Бельгія та Греція серіалізувались навіть до лютого 2019 року; однак, не відповідно до директиви ЄС, а з використанням наклейок. У них є ще шість років, протягом яких вони можуть повністю реалізувати директиву ЄС. Ліки, які не мають необхідних захисних елементів згідно з Директивою ЄС про фальсифіковані ліки, більше не можуть продаватися в ЄС. На практиці фармацевт сканує окрему упаковку перед тим, як розподілити її пацієнту. Препарат не можна видавати, якщо сканування повертає повідомлення про помилку.

Деталі Директиви ЄС 2011/62/ЄС:

Згідно з Директивою ЄС про фальсифіковані ліки кожна упаковка повинна містити наступну інформацію у вигляді простого тексту та закодованого у машиночитаному 2D-матричному коді:

(01) Product code (e.g. GTIN/NTIN/PPN)
(21) Serial Number
(10) Batch/Lot Number
(17) Expiry Date
National reimbursement number
(applicable to DE, FR, BR, ES, PT,)

Рис. 1.1 DataMatrix

- Ідентифікаційний номер продукту у форматі GTIN, NTIN (або PPN для Німеччини)
- Серійний номер
- Індивідуальний серійний номер однієї упаковки
- За необхідності, національні номери відшкодування витрат на охорону здоров'я, NHRN, для відшкодування збитків національним страховикам

Крім того, окремі упаковки повинні бути захищені від маніпуляцій непошкодженою герметичною печаткою.

З листопада 2018 року на кожній окремій упаковці у вигляді 2D-матричного коду відображається така інформація:

- Національний ідентифікаційний номер товару у форматі GTIN
- Індивідуальний серійний номер окремої упаковки
- Термін придатності препарату
- Назва партії ліків

Закон 61-ФЗ: Запровадження в Росії директиви про фальсифікацію:

Фальсифікація ліків та чорний ринок є серйозною проблемою в Росії. У відповідь на це Державна дума прийняла Закон 61-ФЗ про обіг лікарських засобів, який покликаний підвищити безпеку ліків. Пілотний проект, обмежений кількома основними діючими речовинами, який контролював виробництво та ланцюжок поставок фармацевтичної продукції, що продається у вільному продажу в районах Москви та Санкт-Петербурга, тривав до кінця 2017 року. Метою було надійне

виявлення фальсифікованих ліків. З січня 2020 року всі ліки, що відпускаються за рецептом, мають бути серіалізованими та агрегованими.

Директива про фальсифіковані ліки в Бразилії: Track & Trace with SNCM :

Бразильський орган охорони здоров'я ANVISA - крок вперед. У 2014 році він прийняв скорочені норми SNCM щодо системи Track & Trace, яка використовується для моніторингу виробництва та ланцюга постачання ліків. Подібно до європейської та американської систем, на упаковці слід надрукувати ідентифікаційний номер препарату, серійний номер, номер партії та термін придатності ліків. Після деяких затримок із впровадженням, в даний час застосовуваний регламент визначає 1 квітня 2022 року як кінцевий термін для всебічного впровадження "Національної системи контролю за медичними засобами".

DSCSA - Закон про безпеку лікарських засобів забезпечує безпечні ліки в США:

Директива США про фальсифікацію DCSA є частиною Закону США про безпеку лікарських засобів (FQSA). Він був підписаний федеральним законом у 2013 році і, як очікується, буде повністю впроваджений до 2023 року. З 2015 року виробники та роздрібні продавці ліків, що відпускаються за рецептом та відшкодовуються, зобов'язані надавати необхідну інформацію на рівні партій та реєструвати всі дані про транзакції до ліки остаточно відпускаються в аптеці. Повна відслідковуваність номерів партій ліків дозволяє швидко розпізнати та вилучити не тільки сфальсифіковані, але й викрадені партії наркотиків. У цьому випадку поступові терміни застосовуються до різних зацікавлених сторін, виробників, постачальників пакувальних послуг, оптових продавців та продавців. З листопада 2023 року зобов'язання буде поширюватися на маркування лікарських засобів в одній упаковці. Перехідний період для маркування окремих упаковок розпочався в листопаді 2018 р. Відповідні технічні вимоги, які раніше застосовувались в окремих штатах, і зараз замінюються загальноамериканськими правилами. З листопада 2018 року на кожній окремій упаковці у вигляді двовимірного коду матриці повинна відображатися така інформація:

- Національний ідентифікаційний номер товару у форматі GTIN
- Індивідуальний серійний номер окремої упаковки
- Термін придатності препарату
- Назва партії ліків

Рис 1.2 GS1 DataMatrix USA

Директива про підробку в Південній Африці: серіалізація вторинної та третинної упаковки

Південна Африка долучилася до глобальної боротьби з контрафактними ліками. Експорт фармацевтичної продукції з ЄС до Південної Африки перевищує 1 млрд. Євро на рік, що робить останні директиви щодо кодування та серіалізації у ПАР великою зміною для європейських виробників фармацевтичних препаратів. Компаніям, що перевозять товари до Південної Африки, доведеться реорганізувати процеси упаковки, щоб задовольнити вимоги. Очікується, що серіалізація вторинної та третинної упаковки буде завершена до 2022 року. Поступове впровадження цього регулювання має на меті дати фармацевтичним компаніям достатньо часу для підготовки.

1.2 Огляд існуючих аналогів

Українськи аналоги :

- “Ліки контроль”. Мобільний додаток, з допомогою якого можна знайти аналоги ліків, переглянути інструкцію до ліків, перевірити реєстрацію препарату в Україні, перевірити заборонені серії препаратів, переглянути документи Держлікслужби, що регламентують заборону того чи іншого лікарського

засобу. Перевірка на справжність ліків відбувається за допомогою введення реєстраційного номеру.

- “Ліки”. Веб-сайт, з допомогою якого можна перевірити реєстрацію і якість ліків. Для кожного варіанту є різні поля вводу. Щоб дізнатись інформацію щодо реєстрації ліків, потрібно ввести їхній реєстраційний номер. Щоб дізнатись інформацію про якість щодо лікарського засобу потрібно ввести реєстраційний номер і серію товару.

Закордонні аналоги :

- “Честный знак” російський мобільний додаток. Перевірка справжності ліків відбувається за допомогою штрих-коду DataMatrix. DataMatrix код ділиться на дві частини: код ідентифікації, який визначає позицію товару в системі і єдиному каталозі товарів, і код перевірки або крипто-хвіст, який генерує оператор за допомогою вітчизняних технологій криптографії
- “Scan Pharma” російський мобільний додаток. Перевірка справжності ліків відбувається за допомогою штрих-коду DataMatrix або реєстраційного номеру
- “MilliporeSigma Scan Now”, мобільний додаток. За допомогою сканера в ньому відображається інформація, що зберігається у штрих-коді - включаючи код товару, номер партії, термін зберігання та глобальний номер товарної позиції (GTIN) - разом із посиланнями на документи такі як сертифікати аналізу (Certificates of Analysis) та паспорти безпеки (Safety Data Sheets)
- “Smart Rx Manager” , мобільний додаток. При скануванні ліків з’являється “Scan result”, в якому відображається код товару, серійний номер, номер партії і термін зберігання, а також статус товару “Active” та “Destroyed”

При дослідженні різних аналогів було зроблено висновок, що більшість якісних розробок створено за кордоном, проте вони не діють в Україні. Порівнюючи вітчизняні продукти з розробленою системою дипломної роботи, можна сказати, що вони відрізняються. “Ліки контроль” - мобільний додаток, він орієнтований на перевірку реєстраційного номеру, а не унікального коду. “Ліки” - веб-сайт, орієнтований на перевірку реєстраційного номеру і номеру серії товару.

1.3 Види штрих-кодів

Штриховий код - це графічне чорно-біле зображення, яке наноситься на поверхню товару для маркування та подальшої ідентифікації. У кожного коду - свій унікальний малюнок, в якому зашифровані числові і текстові дані. Залежно від малюнка і обсягу пам'яті виділяють наступні види штрих-кодів: одновимірний (з вертикальними смугами) і двовимірний (поєднання смуг з іншими геометричними фігурами, наприклад, точками). Велике значення мають стандарти виробництва, вони ж - типи штрих-кодів. Ось найбільш популярні: лінійні 13-ти символьні EAN-коди, які зустрічаються на упаковках продуктів, і двовимірні, використовувані для передачі інформації з одного пристрою на інший.

Штрихове кодування інформації використовується у всіх галузях, де є необхідність в оперативній обробці даних, швидкому обліку, приймання та відпуску товарів і їх ефективної інвентаризації. Технологія широко поширена в сфері оптової та роздрібної торгівлі, промисловості, на складах, в логістиці. Для генерації і друку штрих-кодів використовують спеціальні принтери, а для читання - сканери, що значно знижує ймовірність людської помилки під час процедури введення даних.

Штрихові коди поділяють на дві категорії - одномірні (лінійні) і двовимірні. Вони відрізняються графічним зображенням, ємністю, тобто «Місткістю» записуються символів, способом шифрування інформації і читання. Також вибір залежить від виду даних, які необхідно записати - цифри, букви (заголовні і стічні, кирилиця і латиниця). Виходячи від цих критеріїв одномірні і двовимірні штрих-коди використовують в різних сферах, а для їх зчитування може знадобитися різне устаткування.

Лінійні або одномірні (1D) коди представлені у вигляді ряду вертикальних ліній і пробілів різної ширини. Вони зчитаються строго в горизонтальному напрямку, зліва направо. Вміщають близько 20-30 символів - наприклад, артикул або серійний номер товару. Підходять для автоматичної розшифровки зчитувачем і ручної обробки

відомостей: для цього достатньо ввести в систему буквенну або числову інформацію, яка вказується під малюнком.

Найпоширеніший з лінійних кодів - це міжнародний стандарт кодування EAN. Він складається з 8 або 13 символів і має таку структуру:

- перші 2-3 цифри - країна походження;
- наступні 4-5 - виробник товару;
- що йдуть слідом 5 символів - дані про товар;
- останні 1-2 цифри - контрольні, необхідні для перевірки правильності зчитування всієї інформації сканером.

Деякі коди, також виконані в стандарті EAN, використовуються для маркування конкретних товарів. Наприклад, ISBN використовується для маркування книг і завжди починається з 978, а ISSN, код періодичної друкованої продукції, з 977. Для маркування лікарських препаратів застосовують бінарний фармакод, що представляє собою чергування одиниць і нулів.

Основними стандартами лінійних ШК є:

- UPC/EAN-128;
- EAN-13;
- UPC-E;
- Code39;
- UPC-A;
- EAN-8;
- «Interleaved 2of5».

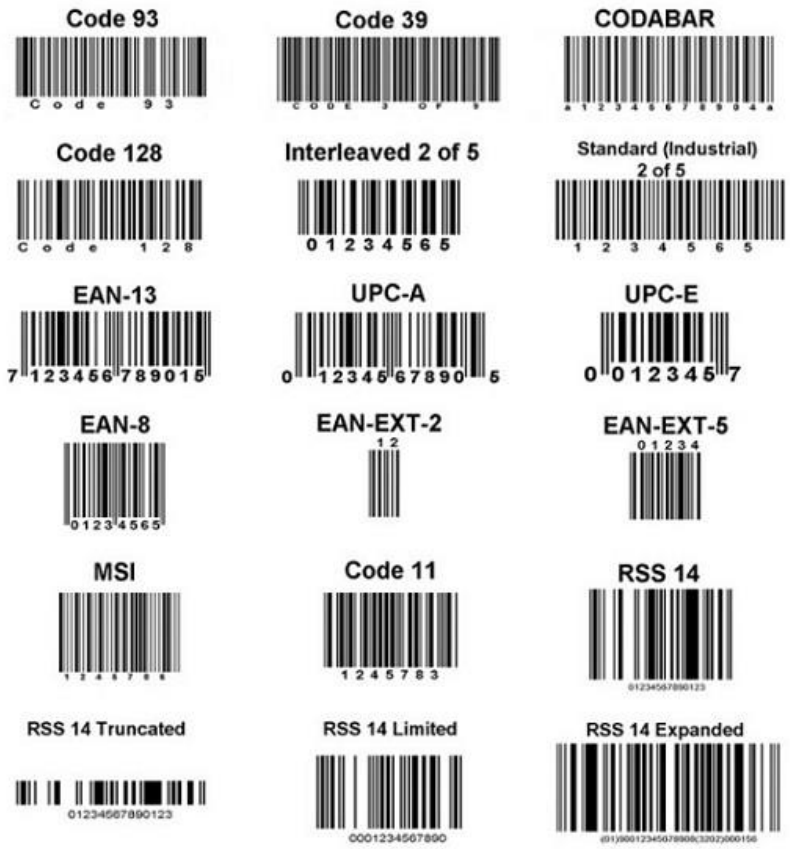

Рис. 1.3 Лінійні штрих-коди

На відміну від лінійних, двовимірні коди зчитуються в двох вимірах - по горизонталі і по вертикалі. Вони складаються з ліній, розташованих в різних напрямках, в тому числі під прямим кутом, точок, клітин і інших фігур і мають яскраво виражені пікселі. Можуть зчитуватися під довільним кутом.

До двовимірних відносяться багаторівневі і матричні коди. Багаторівневі представляють собою «склейку» кількох лінійних, поставлених один на одного, а матричні упаковують інформацію у вигляді чорних і білих «клітин» квадратної або прямокутної форми.

Двовимірні коди зберігають значно більший обсяг даних - до 4 000 символів. Вони підходять для шифрування продукції, що має безліч характеристик і параметрів (до декількох сторінок).

Найвідоміші двовимірні коди - DataMatrix, QR-код і Aztec.

DataMatrix - двовимірний код змінної довжини, який виконаний у вигляді матриці, що складається з білих і чорних крапок різного розміру. Вбудована система корекції помилок дозволяє відновлювати до 30% даних з пошкоджених ділянок штрих-коду.

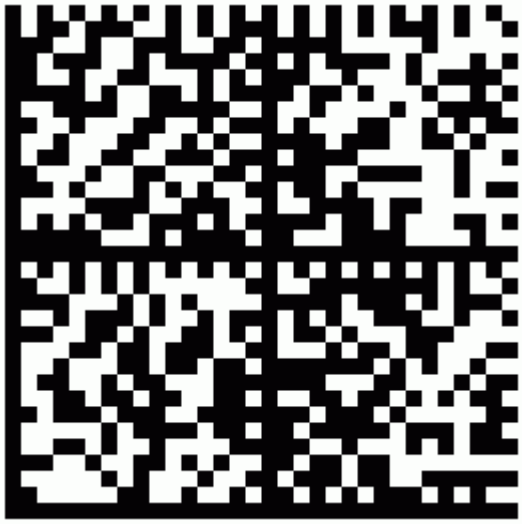

Рис. 1.4 DataMatrix

QR-код - це розробка японської компанії Denso Wave. QR-код легко пізнаваний по зовнішньому вигляду - він оснащений трьома квадратами по кутах (два зліва і один праворуч). Підходить для миттєвої передачі інформації з одного пристрою на інший. У разі зчитування пошкодженого QR-коду, зчитувач здатний «добудувати» і відновити до 30% втрачених даних.

Рис 1.5 QR-код

Код Aztec нагадує мішень - в його центрі розташований багатошаровий квадрат, необхідний для позиціонування. Шари даних представлені білими і чорними точками. Володіє максимальною корекцією помилок - до 95%, і відрізняється високою надійністю.

Рис. 1.6 Aztec

Крім цих двовимірних кодів виділяють менш поширені MaxiCode, PDF417, Microsoft Tag.

Висновки до розділу 1

В цьому розділі було описано існуючі вирішення проблеми фальсифікації ліків, директиви різних країн та їхня відмінність. Було проаналізовано види найбільш застосованих штрих-кодів, їхні відмінності, вигляд і структуру. Також було досліджено та проаналізовано аналоги різних країн для ідентифікації підроблених ліків.

РОЗДІЛ 2
ПРОЕКТУВАННЯ СИСТЕМИ

2.1. DataMatrix

DataMatrix - це матричний (двовимірний) штрих-код, який може бути надрукований у вигляді квадратного або прямокутного символу, що складається з окремих крапок або квадратів. Це подання являє собою впорядковану сітку темних і світлих точок, обрисованих моделлю Finder. Модель Finder частково використовується для вказівки орієнтації та структури символу. Дані кодуються за допомогою серії темних або світлих точок на основі заздалегідь визначеного розміру. Розмір цих точок відомий як X-розмірність. Вважається, що ISO / IEC DataMatrix більш надійний, ніж ISO / IEC QR Code, через більш високий рівень корекції помилок. Відсоток корекції даних для DMC становить близько 30%, а для QR-кодів можливі 4 різних рівня: рівень L (низький) 7%, рівень M 15%, рівень Q 25%, рівень H (високий) 30 % кодових слів можна відновити. Потенційно, єдиною унікальною перевагою використання QR Code замість символіки DataMatrix є більш висока ефективність при кодуванні символів японського ієрогліфічного письма. Тим не менш, це не розглядається в якості конкурентної переваги в Системі, оскільки тільки цифри (числа), латинський алфавіт і набір обраних символів можуть бути закодовані в стандартних символіках. QR-коди комерційно більш відомі, їх видно на різних рекламних роликах, веб-сайтах, гігантських плакатах і роздрібних магазинах. Код матриці даних дуже широко використовується в промисловості для маркування невеликих компонентів, наприклад ліків.
Різниця між носієм даних та структурою даних:

Носій даних - це графічне представлення даних у машиночитаній формі, що використовується для автоматичного зчитування рядків елементів. DataMatrix - це визнана та стандартизована реалізація використання матриці даних ISO / IEC.

DataMatrix складається з двох окремих частин (див. Рис. 2.2): модель Finder, яка використовується сканером для пошуку символу, та самі закодовані дані. Модель Finder визначає форму (квадрат або прямокутник), розмір, розмірність X та кількість рядків і стовпців у символі. Вона має функцію, подібну до допоміжних моделей (моделі Пуск, Зупинка та Центр) у штрих-коді EAN / UPC, і дозволяє сканеру ідентифікувати символ як DataMatrix. Суцільний темний колір називається “L finder

pattern”. Він в основному використовується для визначення розміру, орієнтації та спотворення символу. Інші дві сторони моделі Finder - це чергуються світлі та темні елементи, відомі як "Годинникова доріжка". Це визначає основну структуру символу, а також може допомогти визначити його розмір та спотворення.

Далі дані кодуються в матриці за моделлю Finder. Це переклад у двійкові символи символів DataMatrix (цифрові або буквено-цифрові).

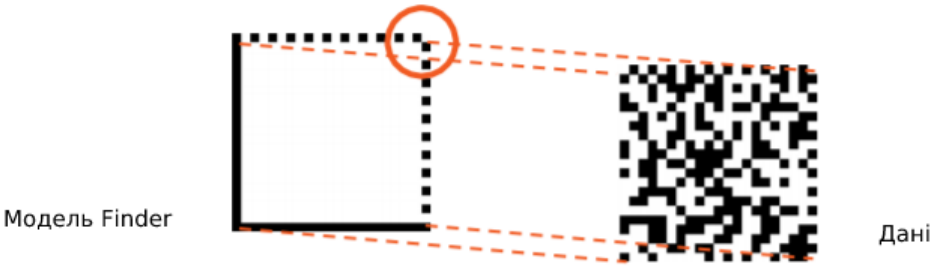

Рис. 2.1 Модель Finder і дані

Як і лінійні (1D) штрих-коди, DataMatrix має обов’язкову спокійну зону. Це світла область навколо символу, яка не повинна містити жодного графічного елемента, який може порушити читання штрих-коду. Він має постійну ширину, рівну X-мірності символу на кожній з чотирьох сторін. Кожен символ матриці даних складається з кількості рядків і стовпців. DataMatrix завжди має парну кількість рядків і стовпців. Тому він завжди має світлий квадрат у правому верхньому куті (обведений на малюнку вище). Очевидно, що цей кут буде темним, якщо символ DataMatrix надрукований негативно (друк із зворотним відбиттям).

При впровадженні DataMatrix необхідно зробити вибір символьної форми (на основі підтримки конфігурації, вільного місця на типі продукту, кількості даних для кодування, процесу друку тощо). Можна кодувати однакові дані у двох формах DataMatrix:

Рис. 2.2 Квадратна форма проти прямокутної форми

Квадратна форма є найбільш часто використовуваною і дозволяє кодувати найбільший обсяг даних відповідно до ISO / IEC 16022 Інформаційні технології - методи автоматичної ідентифікації та збору даних - Специфікація символіки штрих-коду матриці даних. Однак прямокутна форма з обмеженою висотою більше підходить для деяких високошвидкісних методів друку та для незвичних просторів друку.

2.2.1 Кодування даних

Загальна версія DataMatrix підтримує різні структури кодування, які можуть використовуватися в одному символі одночасно. Приклади включають: ASCII, ISO / IEC 646, C40, Text, X12, EDIFACT та База 256. Ці структури дають можливість максимізувати ефективність кодування необхідних даних у символі матриці даних.

Найпростішим рішенням, передбаченим стандартами GS1, є кодування даних із використанням підмножини ISO / IEC 646 для всієї інформації. Цей обмежений набір символів підтримується майже всіма комп'ютерними системами, доступними сьогодні у всьому світі.

Для глобального використання неоднозначність може виникнути через:

- Один і той же код ASCII використовується для різних розширених символів у різних географічних регіонах
- Неможливість багатьох користувачів ввести розширені символи (через комп’ютерні обмеження та людський фактор).
- Можна використовувати лише символи, що містяться в підмножині інваріантів ISO 646. Слід зазначити, що пробіли не можна кодувати
- Ідентифікатори використовуються для всіх закодованих даних

2.2.2 Зчитування та декодування DataMatrix

Після друку символу для збору закодованих даних потрібен пристрій для зчитування або сканування. Слово «сканування» зазвичай використовується для охоплення двох окремих етапів процесу:

1. Фактичне сканування (зчитування темних і світлих ділянок)
2. Декодування (обробка захопленого зображення для визначення закодованих даних)

У цьому відношенні DataMatrix працює дуже схоже на відомі лінійні штрих-коди, такі як EAN-13, ITF-14, GS1-128 та DataBar. Однак він відрізняється від цих лінійних символів тим, що вимагає сканування на основі камери або зображення, оскільки дані кодуються у двох вимірах.

Після декодування дані передаються в інформаційну систему для подальшої обробки.

Як і інші двовимірні штрих-коди, DataMatrix можна зчитувати лише за допомогою камер обробки зображень або пристроїв CCD (Charge Couple Device). Принцип заснований на тому, що спочатку фіксують зображення символу, а потім аналізують його. Моделі Finder використовуються для відтворення віртуального образу матриці. Як правило, кожна з темних і світлих областей у матриці перетворюється у двійкові значення (1 або 0). Потім це обробляється згідно з еталонним алгоритмом декодування DataMatrix, як визначено у ISO / IEC 16022, на основі ідеального зображення

2.2 Розбір складових частин штрих-коду

Для того, щоб розуміти, яким чином унікальний номер стає унікальним для упаковки ліків, необхідно ознайомитись з його складовими. На рисунку 2.3 перераховані всі символи дозволені для використання в рядках елементів ідентифікатора системи.

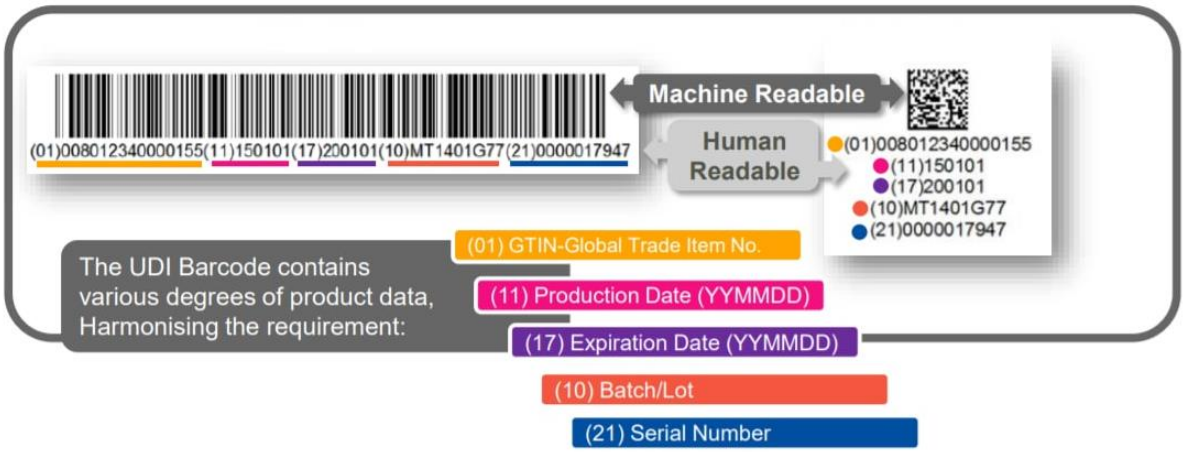

Рис. 2.3 Складові штрих-коду

Кожна окрема упаковка ліків, що відпускається, повинна мати унікальний ідентифікатор, кодований за допомогою двовимірної матриці даних (штрих-коду). Якщо розмір упаковки це дозволяє, упаковка також буде містити ту саму інформацію в зручному для читання тексті, надрукованому поруч із 2D-кодом, де це можливо.

Унікальний ідентифікатор складається з:

1. GTIN
2. Дата виробництва
3. Термін придатності
4. Номер партії
5. Серійний номер

Оскільки ця інформація буде надрукована та закодована на кожній упаковці, вона також може бути використана для деяких видів діяльності, таких як повторне замовлення запасів, ротація запасів та перевірка точності, наприклад в оптових та аптечних магазинах.

2.3.1 Серійний номер(Serial Number)

Серійний номер - це буквено-цифрове поле змінної довжини (до 20), яке несе практично всю інформацію про товар, його виробника, дату випуску і, можливо, про інші характеристики виробу. На сьогоднішній день маркування товарів - обов'язкова умова для виробника, цей процес значно полегшує процедуру ідентифікації того чи

іншого товару, вона необхідна при транспортуванні, митному оформленні та обліку товарів вже безпосередньо у продавця.

Серійний номер присвоюється об’єкту протягом усього його життя. У поєднанні з GTIN серійний номер однозначно ідентифікує окремий товар.

Власник торгової марки та виробник несуть відповідальність за не дублювання серійних номерів GTIN. Для повторного використання серійних номерів із GTIN необхідно враховувати секторальні обмеження. Система не прийме повторний серійний номер, тому що він обмежений 10000 кодами на партію. Дані, передані з пристрою для зчитування штрих-коду, означають, що рядок елемента, що позначає серійний номер, було захоплено. Оскільки цей рядок елемента є атрибутом торгової позиції, він повинен оброблятися разом із GTIN торгової позиції, до якої він відноситься.

2.3.2 Тестування випадковості серійних номерів

Статистичні тести

Набір статистичних випробувань NIST - це пакет з 16 тестів, розроблених для перевірки випадковості (довільно довгих) двійкових послідовностей, створених генераторами випадкових чи псевдовипадкових чисел. Тести фокусуються на різноманітних різних типах невипадковості, які можуть існувати в послідовності. Кожен тест базується на підрахованому статистичному значенні тесту, яке є функцією тестованої послідовності. Тестова статистика використовується для обчислення значення P, яке узагальнює силу доказів щодо випадковості. Кожне значення P можна інтерпретувати як ймовірність того, що ідеальний генератор випадкових чисел створив би послідовність, менш випадкову, ніж послідовність, яку тестували, враховуючи тип невипадковості, оцінений тестом. Використання значень P призначене для того, щоб дозволити індивідуальному випробуванню генератора легко та об'єктивно інтерпретувати результати випробувань та оцінити якість генератора. NIST SP 800-22 надає високорівневий опис та приклади для кожного з 16 тестів у наборі тестів, а також математичну основу для кожного тесту. Крім того,

документ містить вказівки щодо вказівки параметрів, необхідних для тестів, і для інтерпретації результатів тестування, як для однієї послідовності для даного тесту, так і для кількох послідовностей для цього тесту.

16 статистичних тестів, що містяться в наборі тестів:

1. Тест частоти (монобітний) визначає, чи дорівнює кількість одиниць та нулів у тестованій послідовності приблизно ½, як очікується для справді випадкової справедливої двійкової послідовності. Усі наступні тести залежать від проходження цього тесту.
2. Тест на частоту всередині блоку визначає, чи дорівнює частота частот у M-бітовому блоці тестованої послідовності приблизно M / 2, як і слід було очікувати за умови припущення про випадковість.
3. Тест запусків використовується, щоб визначити, чи є загальна кількість прогонів одиниць та нулів різної довжини такою, як очікується для випадкової послідовності. Тест пробігів можна розглядати як визначення чи коливання між нулями та одиницями є занадто швидким чи занадто повільним.
4. Тест на найдовший пробіг одиниць у блоці визначає, чи відповідає найдовший пробіг тих, що знаходяться в M-бітовому блоці тестованої послідовності, довжині найдовшого прогону, який можна було б очікувати у випадковій послідовності з M бітів.
5. Тест випадкової двійкової матриці використовує неперервні підматриці, утворені з усієї послідовності, для перевірки лінійної залежності серед підрядків фіксованої довжини тестованої послідовності.
6. Тест дискретного перетворення Фур'є (спектральний) використовує висоти піків дискретного швидкого перетворення Фур'є випробовуваної послідовності для виявлення періодичних ознак (тобто повторюваних шаблонів), які вказували б на відхилення від припущення про випадковість.
7. Тест на збіг шаблонів, що не перекривається, визначає, чи є занадто багато випадків заздалегідь визначених аперіодичних шаблонів.
8. Тест на збіг шаблонів, що перекривається, також визначає, чи є занадто багато випадків заздалегідь визначених шаблонів. Зауважте, що як для тестів шаблонів, що збігаються, так і для не перекриваються, якщо шаблон не знайдено, вікно збігу ковзає на одну бітну позицію. Якщо шаблон знайдений, вікно збігу ковзає на одну бітову позицію перед відновленням тесту накладання, тоді як вікно збігу ковзає m бітових позицій для неперекриваючого тесту, де m - довжина шаблону.
9. Тест Маурера "Універсальний статистичний" визначає, чи можна перевірену послідовність значно стиснути без втрати інформації. Це досягається шляхом

оцінки розподілу середніх відстаней між шаблонами, що відбуваються в послідовності. Усереднення виконується за числами цифр при двійковому розширенні цих відстаней.

10. Тест на стиснення Лемпеля-Жива вивчає кількість кумулятивно різних візерунків у тестовій послідовності, щоб визначити, наскільки послідовність може бути стиснута. Справді випадкова послідовність матиме характерну кількість різних моделей.
11. Тест лінійної складності визначає, чи є послідовність достатньо складною, щоб вважати її випадковою. Це досягається вивченням послідовності для визначення довжини лінійного регістру зсуву зворотного зв'язку (LFSR), який створював би послідовність. Короткий регістр зворотного зв'язку передбачає невипадковість.
12. Серійний тест перевіряє рівномірність розподілу (-ів) перекриваюття m-бітових шаблонів для різних довжин шаблонів, m. Випадкові послідовності демонструють однорідність: кожен m-бітовий шаблон повинен з'являтися в середньому так само часто, як і будь-який інший m-бітовий шаблон.
13. Приблизний тест на ентропію порівнює частоту перекриття блоків двох послідовних довжин (m та m + 1) із очікуваним результатом для випадкової послідовності.
14. Тест накопичувальних сум визначає, чи є максимальне абсолютне значення сукупної суми часткових послідовностей, що зустрічаються в тестованій послідовності, занадто великим чи замалим щодо очікуваної поведінки такої кумулятивної суми для випадкових послідовностей. Сукупна сума може розглядатися як випадкова прогулянка. Для випадкової послідовності випадкова прогулянка повинна коливатися навколо нуля. Великі значення прогулянки вказують на занадто багато нулів або одиниць біля початку послідовності. Невеликі значення вказують на те, що нулі та одиниці перемішуються занадто рівномірно.
15. Тест на випадкові екскурсії визначає, чи розподіляється кількість відвідувань випадкової прогулянки із сукупною сумою до цілих рівнів (“станів”) між послідовними переходами нульового рівня, як очікувалося, для справді випадкової послідовності.
16. Тест "Випадкові екскурсії" розширює тест "Випадкові екскурсії", досліджуючи розподіл переходів через безліч переходів між переходами нульового рівня.

2.3.3 GTIN

GTIN - поняття, яке відображає унікальну ідентифікацію різних предметів торгівлі. Будь-яка деталь, яка чим-небудь вирізняється з-поміж інших - кольором, формою, найменуванням або іншим характеристиками, зобов'язана мати свій неповторний код.

Це міжнародно затверджені цифри, які друкуються на товар. Вони використовуються потім в каталогах та інформаційних системах, застосовуються для визначення одиниць в усьому світі. Їх наносять на пачку, коробку у вигляді штрих-коду. Число може бути довжиною в 8, 12, 13 або 14, але структура при цьому завжди буде на рівні 14. Просто порожні місця будуть закриватися нулями. Тому стандартне програмне забезпечення може зчитувати тільки 14 цифр.

Сам по собі GTIN не має елементів класифікації. Це не означаючий нічого цифровий код, під поняття якого потрапляють всі існуючі типи нумерації - ITF-14, EAN-13 і EAN-8.

Кожному товару з торгової мережі присвоюється свій номер. Він не повторюється більше у інших продуктів, не може бути ідентичним для двох різних позицій. Статус у них міжнародний - це зроблено, щоб виріб можна було однозначно визначити навіть в інших країнах.

Кілька ознак GTIN-продукції:

- спеціальні позначення - префікс фірми, великі червоні літери;
- діє у всіх державах;
- легко інтегрувати в будь-яку систему і електронний каталог;
- можна автоматично зчитувати відповідними сканерами;
- максимально точно визначається, якщо завдано чорним чорнилом на білу матову папір.

Поняття GTIN використовується виключно для маркування, з його допомогою ведеться логістичний облік.

Товарної одиницею можна назвати будь-який предмет, який братиме участь в торгівлі. Головне - він повинен володіти особливостями, які потрібні продукту, який задіяний в операціях між різними фірмами. Виріб можна замовити, оцінити або продати в будь-якій частині ланцюга. Це стосується не тільки поодиноких і штучних продуктів, але і цілих коробок, і продукції в інший транспортній тарі.

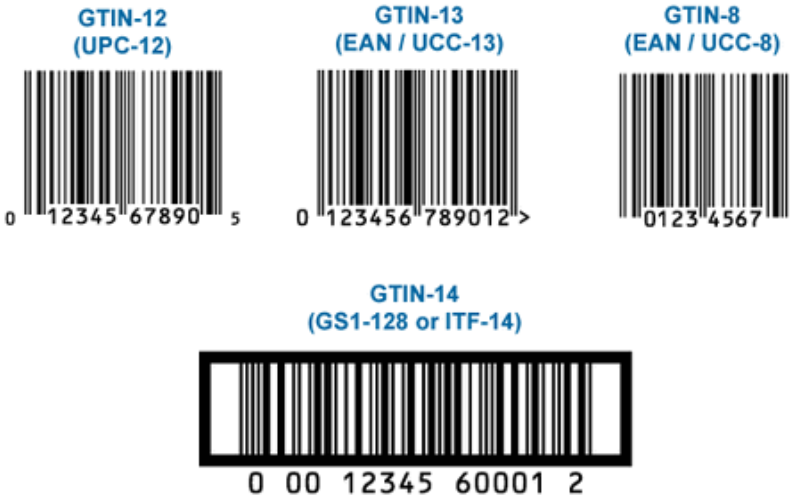

Рис 2. 4 Типи GTIN

Таблиця 2.1

Таблиця глобальних номерів

Тип	14-розрядний GTIN													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
GTIN-14 (ITF-14)	1	4	6	0	9	5	3	3	3	3	4	2	3	6
GTIN-13 (EAN-13)	0	4	6	0	9	4	3	2	8	7	1	2	4	7
GTIN-8 (EAN-8)	0	0	0	0	0	0	4	6	0	9	8	7	6	5

Особливості:

Є кілька нюансів, які виділяють GTIN серед інших ШК:

- Не відноситься до значних номерів. У ньому немає ніякої інформації про товар. Він просто вказує на продукт.
- З його допомогою можна однозначно визначити продукцію. У жодній країні світу не може бути прикріплений цей же числовий номер до іншого виробу. Для адміністрування GTIN EAN code працює спеціальна організація під назвою GS1.

- Будується виключно по заздалегідь прописаним правилам. Довжина може бути від 8 до 14 знаків в залежності від того, для чого призначений товар.

Види кодування:

Загальне правило для всіх компаній світу полягає в тому, що без унікального коду продукт не може продаватися. З моменту присвоєння він не змінюється до тих пір, поки не зміниться назва або інші властивості.

Існує багато типів - EAN-8 і 13, UCC-12 і 14, UPC-A і E, ITF-14 і інші. Для книг придумали окремий ISBN, але все це різні способи видати одиниці продукту GTIN.

Стандартні розрахунки контрольної цифри для структур даних:

Цей алгоритм ідентичний для всіх числових структур даних із фіксованою довжиною (включаючи GDTI, GLN, GRAI тощо), для яких потрібна контрольна цифра.

	Позиції цифр																	
GTIN-8											N1	N2	N3	N4	N5	N6	N7	N8
GTIN-12							N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12
GTIN-13						N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13
GTIN-14					N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	N14
17 цифр		N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	N14	N15	N16	N17
18 цифр	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	N14	N15	N16	N17	N18
	Помножте значення кожної позиції на :																	
	x3	x1	x3	x1	x3	x1	x3	x1	x3	x1	x3	x1	x3	x1	x3	x1	x3	
	Накопичені результати = сума																	
	Відніміть суму з найближчого рівного (або вище) кратного десяти = контрольна цифра																	

Рис. 2.5 Алгоритм розрахунку

Приклад розрахунку контрольної цифри для 18-значного поля																		
Позиції	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	N14	N15	N16	N17	N18
Номер без контрольної цифри	3	7	6	1	0	4	2	5	0	0	2	1	2	3	4	5	6	
	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
Крок 1 : помножити на	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	
	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	
Крок 2 : підсумувати результати	9	7	18	1	0	4	6	5	0	0	6	1	6	3	12	5	18	Результат 101
Крок 3 : відняти суму з найближчого рівного (або вище) кратного десяти(110) = контрольна цифра (9)																		
Номер з контрольною цифрою	3	7	6	1	0	4	2	5	0	0	2	1	2	3	4	5	6	9

Рис. 2.6 Приклад алгоритму розрахунку

2.3.4 Номер партії (batch/lot number)

Номер партії асоціює товар з інформацією, яку виробник вважає важливою для відстеження торгової позиції, до якої застосовується рядок елементів. Дані можуть стосуватися самої торгової позиції або предметів, що містяться. Номер може бути, наприклад, номером виробничої партії, номером зміни, часом або внутрішнім

виробничим кодом. У випадках, коли один і той же товар виготовляється в різних місцях, власник торгової марки та виробник несуть відповідальність за забезпечення не дублювання номерів партій для GTIN. Для повторного використання номерів партій з GTIN необхідно враховувати секторальні обмеження.

2.3.5 Дата виробництва (Production Date)

Поле даних ідентифікатора системи містить дату виробництва. Датою виробництва є дата виготовлення або складання, визначена виробником. Дата може стосуватися самої торгової позиції або предметів, що містяться там. Структура:

- Рік: десятки та одиниці року (наприклад, 2003 = 03), що є обов'язковим.
- Місяць: число місяця (наприклад, січень = 01), яке є обов'язковим.
- День: число дня відповідного місяця (наприклад, другий день = 02); якщо немає необхідності вказувати день, поле необхідно заповнити двома нулями.

2.3.6 Термін придатності (Expiration Date)

Поля даних ідентифікатора системи містять дату закінчення терміну дії. Термін придатності - це дата, яка визначає ліміт споживання або використання товару. Його значення визначається виходячи з контексту товарної позиції (наприклад, для харчових продуктів дата вказуватиме можливість прямого ризику для здоров'я, що виникає внаслідок використання продукту після дати, для фармацевтичної продукції вона вказуватиме на можливість непрямого здоров'я ризик, обумовлений неефективністю продукту після дати).

Структура:

- Рік: десятки та одиниці року (наприклад, 2003 = 03), що є обов'язковим.
- Місяць: число місяця (наприклад, січень = 01), яке є обов'язковим.
- День: число дня відповідного місяця (наприклад, другий день = 02); якщо немає необхідності вказувати день, поле необхідно заповнити двома нулями.

2.3 Структурна схема інформаційних потоків в системі

З метою протидії реалізації підробок в аптеках було розроблено систему з клієнт-серверною архітектурою, яка запобігає потраплянню підроблених ліків до законного потоку постачання.

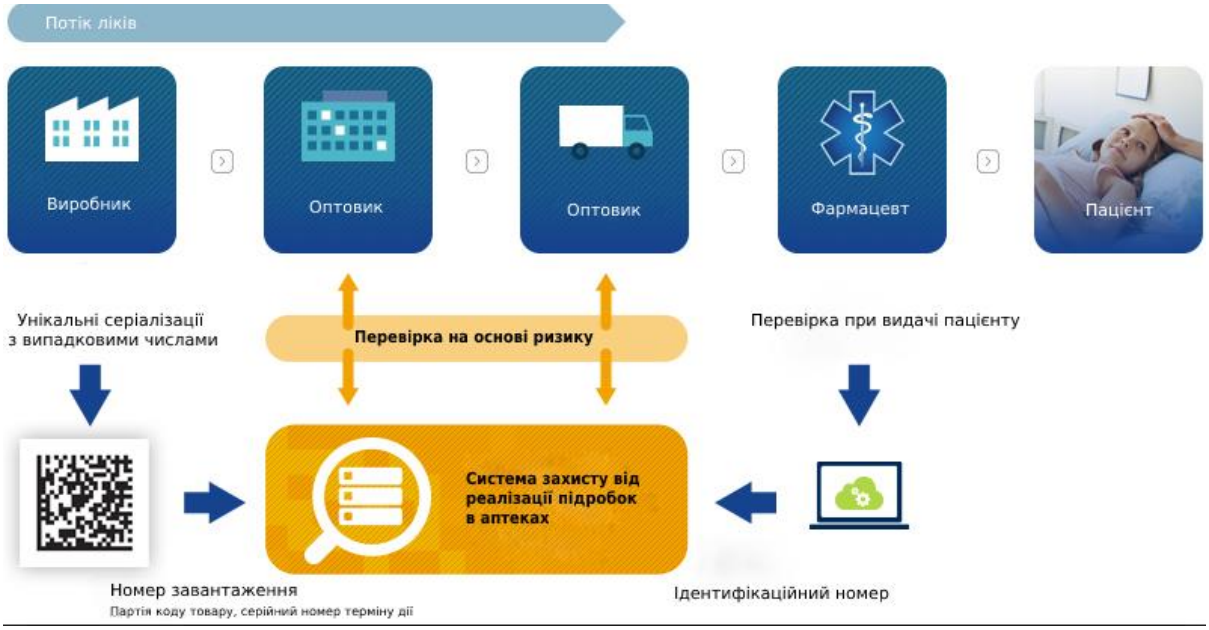

Рис 2.7 Структурна схема інформаційних потоків в системі

Перед розміщенням партій на ринку виробник повинен надати коди, які будуть вноситься в базу системи, надрукувати 2D-код(DataMatrix), який включає унікальний ідентифікатор та застосувати пристрій, що захищає від фальсифікації, на зовнішній упаковці всіх лікарських засобів для кожної окремої упаковки. Перед продажом кожної упаковки, фармацевт повинен сканувати штрих-код ліків. Система розбирає код на елементи GTIN, серійний номер, термін придатності, дата виробництва, номер партії і звіряє з базою даних. Перевірка також відбувається на :

1. Чи не минув термін придатності?
2. Чи були продані такі ліки?
3. Чи були такі ліки повернуті?

Користувач аптеки(фармацевт) отримує помилку тільки в тому разі, якщо продукт прострочений, у всіх інших випадках інформація записується в базу даних, яку ДСУ з лікарських засобів та контролю за наркотиками може переглянути і вчинити міри.

Висновки до розділу 2

В цьому розділі було описано принцип роботи розробленої системи, штрих-код DataMatrix, в якому зашифрований унікальний код продукту та його складові : Серійний номер та тестування випадковості серійних номерів, GTIN, його особливості і алгоритм розрахунку контрольної цифри, а також Номер партії, Дата виробництва і Термін придатності.

РОЗДІЛ 3
РОЗРОБКА СИСТЕМИ

3.1 Середовища і мова розробки системи

IntelliJ IDEA

Для розробки системи було обрано IntelliJ IDEA. IntelliJ IDEA - інтегроване середовище розробки програмного забезпечення для багатьох мов програмування, зокрема Java, JavaScript, Python, розроблена компанією JetBrains. Перша версія з'явилася в січні 2001 року і швидко набула популярності як перше середовище для Java з широким набором інтегрованих інструментів для рефакторінга, які дозволяли програмістам швидко реорганізовувати вихідні коди програм. Дизайн середовища орієнтований на продуктивність роботи програмістів, дозволяючи сконцентруватися на функціональних завданнях, в той час як IntelliJ IDEA бере на себе виконання рутинних операцій. Починаючи з версії 9.0, середовище доступне в двох редакціях: Community Edition і Ultimate Edition. Community Edition є повністю вільною версією,

доступною під ліцензією Apache 2.0, в ній реалізована повна підтримка Java SE, Kotlin, Groovy, Scala, а також інтеграція з найбільш популярними системами управління версіями. В редакції Ultimate Edition, доступною під комерційною ліцензією, реалізована підтримка Java EE, UML-діаграм, підрахунок покриття коду, а також підтримка інших систем управління версіями, мов та фреймворків.

PostgreSQL

Для створення бази даних було обрано PostgreSQL. Це популярна вільна об'єктно-реляційна система управління базами даних. PostgreSQL базується на мові SQL і підтримує численні можливості.

Переваги PostgreSQL:

- Підтримка БД необмеженого розміру;
- Потужні і надійні механізми транзакцій і реплікації;
- Розширювана система вбудованих мов програмування і підтримка завантаження С-сумісних модулів;
- Спадкування;
- Легка розширюваність.

Поточні обмеження PostgreSQL:

- Немає обмежень на максимальний розмір бази даних
- Немає обмежень на кількість записів в таблиці
- Немає обмежень на кількість індексів в таблиці
- Максимальний розмір таблиці - 32 Тбайт
- Максимальний розмір запису - 1,6 Тбайт
- Максимальний розмір поля - 1 Гбайт
- Максимум полів в записі 250-1600 (в залежності від типів полів)

Особливості PostgreSQL:

Функції в PostgreSQL є блоками коду, виконуваними на сервері, а не на клієнта БД. Хоча вони можуть писатися на чистому SQL, реалізація додаткової логіки, наприклад, умовних переходів і циклів, виходить за рамки власне SQL і вимагає використання деяких мовних розширень. Функції можуть писатися з використанням різних мов програмування. PostgreSQL допускає використання функцій, які повертають набір записів, який далі можна використовувати так само, як і результат виконання

звичайного запиту. Функції можуть виконуватися як з правами їх творця, так і з правами поточного користувача. Іноді функції ототожнюються з збереженими процедурами, проте між цими поняттями є різниця.

Тригери в PostgreSQL визначаються як функції, що ініціюються DML-операціями. Наприклад, операція INSERT може запускати тригер, перевіряючий додану запис на відповідності певним умовам. При написанні функцій для тригерів можуть використовуватися різні мови програмування. Тригери асоціюються з таблицями. Множинні тригери виконуються в алфавітному порядку.

Механізм правил в PostgreSQL є механізм створення користувацьких обробників не тільки DML-операцій, а й операції вибірки. Основна відмінність від механізму тригерів полягає в тому, що правила спрацьовують на етапі розбору запиту, до вибору оптимального плану виконання і самого процесу виконання. Правила дозволяють перевизначати поведінку системи при виконанні SQL-операції до таблиці.

Індекси в PostgreSQL наступних типів: B-дерево, хеш, R-дерево, GiST, GIN. При необхідності можна створювати нові типи індексів, хоча це далеко не тривіальний процес.

Багатоверсійність підтримується в PostgreSQL - можлива одночасна модифікація БД кількома користувачами за допомогою механізму Multiversion Concurrency Control (MVCC). Завдяки цьому дотримуються вимоги ACID, і практично відпадає потреба в блокуванні читання.

Розширення PostgreSQL для власних потреб можливо практично в будь-якому аспекті. Є можливість додавати власні перетворення типів, типи даних, домени (призначені для користувача типи з самого початку накладеними обмеженнями), функції (включаючи агрегатні), індекси, оператори (включаючи перевизначення вже існуючих) і процедурні мови.

Спадкування в PostgreSQL реалізовано на рівні таблиць. Таблиці можуть успадковувати характеристики і набори полів від інших таблиць (батьківських). При

цьому дані, додані в породжену таблицю, автоматично будуть брати участь (якщо це не зазначено окремо) в запитах до батьківської таблиці.

Trello

Для планування і розбиття задач було використано Trello. Trello - це одна з найпопулярніших систем управління проектами в режимі онлайн, яка користується особливим попитом серед невеликих компаній і стартапів. Вона дозволяє ефективно організовувати роботу по японській методології канбан-дощок. Вона створена Fog Creek Software в 2011 році на базі MongoDB, Backbone.js і Node.js. Головні переваги, які дозволили Trello домогтися популярності - це:

- Простий інтерфейс
- Майже необмежений безкоштовний доступ
- Зручність в роботі і можливість інтеграції з іншими популярними інструментами для онлайн-роботи

Для організації завдань використовується дошка з картками, які розподіляються за типами. Як правило, завдання розбиваються на:

- Заплановані
- Поточні
- Виконані

Bitbucket

Bitbucket Cloud - це інструмент для розміщення коду та спільної роботи на основі Git, призначений для команди. Найкращі у своєму класі інтеграції Jira і Trello для Bitbucket створюються для всієї команди розробників єдиного простору, в якій її учасники разом працюють над проектом.

Наразі всім користувачам безкоштовно пропонуються наступні можливості:

- Дисковий простір у 2 ГБ на репозиторії.
- Необмежена кількість публічних репозиторіїв.
- Необмежене число приватних репозиторіїв для команди до п'яти людей.
- Доступ до репозиторію за протоколам HTTP та SSH.
- Можливість прив’язати обліковий запис на сервісі до власного домену.
- Вікі (окремо для кожного репозиторію, можна відключити).

- Система обліку помилок (окремо для кожного репозиторію, можна відключити).
- Інтеграція з Google Analytics, Twitter, Basecamp та іншими службами.
- RSS-стрічка історії змін.
- Управління приватністю окремо для кожного репозиторію.
- Для публічних репозиторіїв кількість користувачів не обмежена (BitBucket безкоштовно для проектів відкритого програмного забезпечення).

Мова Java

Java - це об'єктно-орієнтована мова програмування, розроблена компанією Sun Microsystems (в подальшому придбана компанією Oracle). Дата офіційного випуску - 23 травня 1995 року. Програми на Java транслюються в байт-код, який потім виконується віртуальною машиною Java (JVM). JVM - це програма, яка обробляє байтовий код і передає інструкції обладнанню. Перевагою такої реалізації є незалежність байт-коду від операційної системи і устаткування, що дозволяє виконувати Java-додатки на будь-якому пристрої, для якого існує JVM.

Іншою важливою особливістю технології Java є гнучка система безпеки завдяки тому, що виконання програми повністю контролюється віртуальною машиною. Будь-які операції, які перевищують встановлені повноваження програми (наприклад, спроба несанкціонованого доступу до даних або з'єднання з іншим комп'ютером) викликають негайне переривання.

Плюси і мінуси Java

У всіх якостей Java, будь то сувора типізація або об'єктна орієнтованість, є свої плюси і мінуси, а ще вони є у самій Java як у мови.

Плюси:

- Незалежність - код буде працювати на будь-якій платформі, яка підтримує Java.
- Надійність - в чималій мірі досягається завдяки суворій статичній типізації.
- Багатофункціональність.
- Порівняно простий синтаксис.
- Java - основна мова для Android-розробки.
- Об'єктно-орієнтоване програмування (ООП) теж приносить багато вигоди:

1. паралельна розробка;
2. гнучкість;
3. одні й ті ж класи можна використовувати багато разів;
4. код добре організований і його легше підтримувати.

Мінуси:

- Низька швидкість (в порівнянні з C і C ++).
- Вимагає багато пам'яті.
- Немає підтримки низькорівневого програмування (Java - високорівнева мова). Наприклад, у неї немає покажчиків.
- З 2019 року оновлення для бізнесу і комерційного використання стали платними.
- Для ООП потрібен досвід, а планування нової програми займає багато часу.

3.2 Види баз даних

База даних - це впорядкований набір структурованої інформації або даних, які зазвичай зберігаються в електронному вигляді в комп'ютерній системі. База даних зазвичай управляється системою управління базами даних (СКБД). Дані разом з СУБД, а також додатки, які з ними пов'язані, називаються системою баз даних, або, для стислості, просто базою даних.

Дані в найбільш поширених типах сучасних баз даних зазвичай зберігаються у вигляді рядків і стовпців формують таблицю. Цими даними можна легко управляти, змінювати, оновлювати, контролювати та організовувати. У більшості баз даних для запису і запитів даних використовується мова структурованих запитів (SQL).

Типи баз даних

- Реляційні бази даних. Реляційні бази даних стали переважати в 1980-х роках. Дані в реляційної базі організовані у вигляді таблиць, що складаються із стовпців і рядків. Реляційна СУБД забезпечує швидкий і ефективний доступ до структурованої інформації.
- Об'єктно-орієнтовані бази даних. Інформація в об'єктно-орієнтованої бази даних представлена у формі об'єкта, як в об'єктно-орієнтованому програмуванні.

- Розподілені бази даних. Розподілена база даних складається з двох або більше частин, розташованих на різних серверах. Така база даних може зберігатися на декількох комп'ютерах.
- Сховища даних. Будучи централізованим репозиторієм для даних, сховище даних являє собою тип бази даних, спеціально призначеної для швидкого виконання запитів і аналізу.
- Бази даних NoSQL. База даних NoSQL, або нереляційна база даних, дає можливість зберігати і обробляти неструктуровані або слабо структуровані дані (на відміну від реляційної бази даних, яка задає структуру даним, які містяться в ній).
- Графові бази даних. Графова база даних зберігає дані в контексті сутностей і зв'язків між сутностями.
- Бази даних OLTP. База даних OLTP - це база даних призначена для виконання бізнес-транзакцій, що виконуються безліччю користувачів.

Це лише деякі з десятків типів баз даних, які використовуються в даний час. Інші, менш поширені бази даних, призначені для дуже специфічних наукових, фінансових та інших завдань. Крім появи нових типів, бази даних розвиваються в абсолютно нових напрямах - змінюються підходи до розробки технологій, відбуваються значні зрушення, такі як впровадження хмарних технологій і автоматизації. Зокрема, останнім часом з'явилися такі бази даних:

- Бази даних з відкритим вихідним кодом. Такі бази даних мають відкритий вихідний код і можуть управлятися засобами як SQL, так і NoSQL.
- Хмарні бази даних. Хмарна база даних являє собою набір структурованих або неструктурованих даних, розміщених на приватній, публічній або гібридній платформі хмарних обчислень. Існує два типи моделей хмарних баз даних: традиційна база даних і база даних як послуга (DBaaS). У моделі DBaaS адміністративні завдання та обслуговування виконуються постачальником хмарних послуг.
- Багатомодельні бази даних. Багатомодельна база даних об'єднує різні типи моделей баз даних в єдину інтегровану серверну СУБД. Це означає, що вона може містити різні типи даних.
- Документні бази даних / JSON. Бази даних документів призначені для зберігання, вилучення та обробки документорієнтованої інформації і надають сучасний спосіб зберігання даних в форматі JSON, а не у вигляді рядків і стовпців.

- Автономні бази даних. Самоврядні бази даних (також звані автономними) - це новітні і самі революційні хмарні бази даних, які використовують машинне навчання для автоматизації налаштування, захисту, резервного копіювання, оновлення та інших стандартних завдань обслуговування, зазвичай виконуваних адміністраторами баз даних.

3.3 Структура бази даних

При розробці системи було створено реляційну базу даних, в якій знаходяться такі таблиці :

- Roles. Таблиця складається з колонок: id, created(дата та час, коли була створена роль), updated(дата та час, коли колонки ролі оновились) та name(назва ролі)
- Manufacturer. Таблиця складається з колонок : id, name(ім’я виробника)
- Users. Таблиця складається з колонок: id, created(дата та час, коли був створений користувач), updated(дата та час, коли колонки користувача оновились), email(пошта зареєстрованого користувача), name(ім’я користувача), password(зашифрований пароль користувача), status(статус користувача), role_id(id ролі з таблиці roles). Статус користувача може бути активним чи не активним. При створенні користувача, адміністратор системи(роль ROLE_ADMIN) повинен підтвердити його активацію, якщо користувач не був активований, йому прийде лист на пошту для активації.
- Product. Таблиця складається з колонок: id, product_name(назва продукту/ліків), manufacturer_id(id виробника), serial_number(серійний номер ліків), description(опис ліків), created(дата та час занесення ліків в базу), updated(дата та час оновлення даних)
- Identified_errors. Таблиця складається з колонок: id, error_type(вид помилки : дублікат, термін придатності закінчився і т.д.) created(дата та час появи помилки), updated(дата та час оновлення даних). При наявності дублікату, в таблицю записується два стовпця : перший про ліки, які були просканові і успішно продані, а другий про повторне сканування. Одні з них підробка, тому перевірити потрібно обидві партії.
- Reset_password_token. Таблиця складається з колонок: id, created(дата та час створення запиту), updated(дата та час оновлення запиту), expiry_date (використати до зазначеної дати), token, user_id

- Verification_token. Таблиця складається з колонок: id, (дата та час створення запиту), updated(дата та час оновлення запиту), expiry_date(використати до зазначеної дати), token, user_id
- Scanned_item. Таблиця складається з колонок : id, created(дата та час запиту), updated(дата та час оновлення таблиці), identified_errors_id(id таблиці identified_errors, в якій описана причина помилки), product_id, user_id(id користувача, від якого прийшов запит на перевірку)

На рисунку 3.1 можна побачити зв’язки таблиць бази даних:

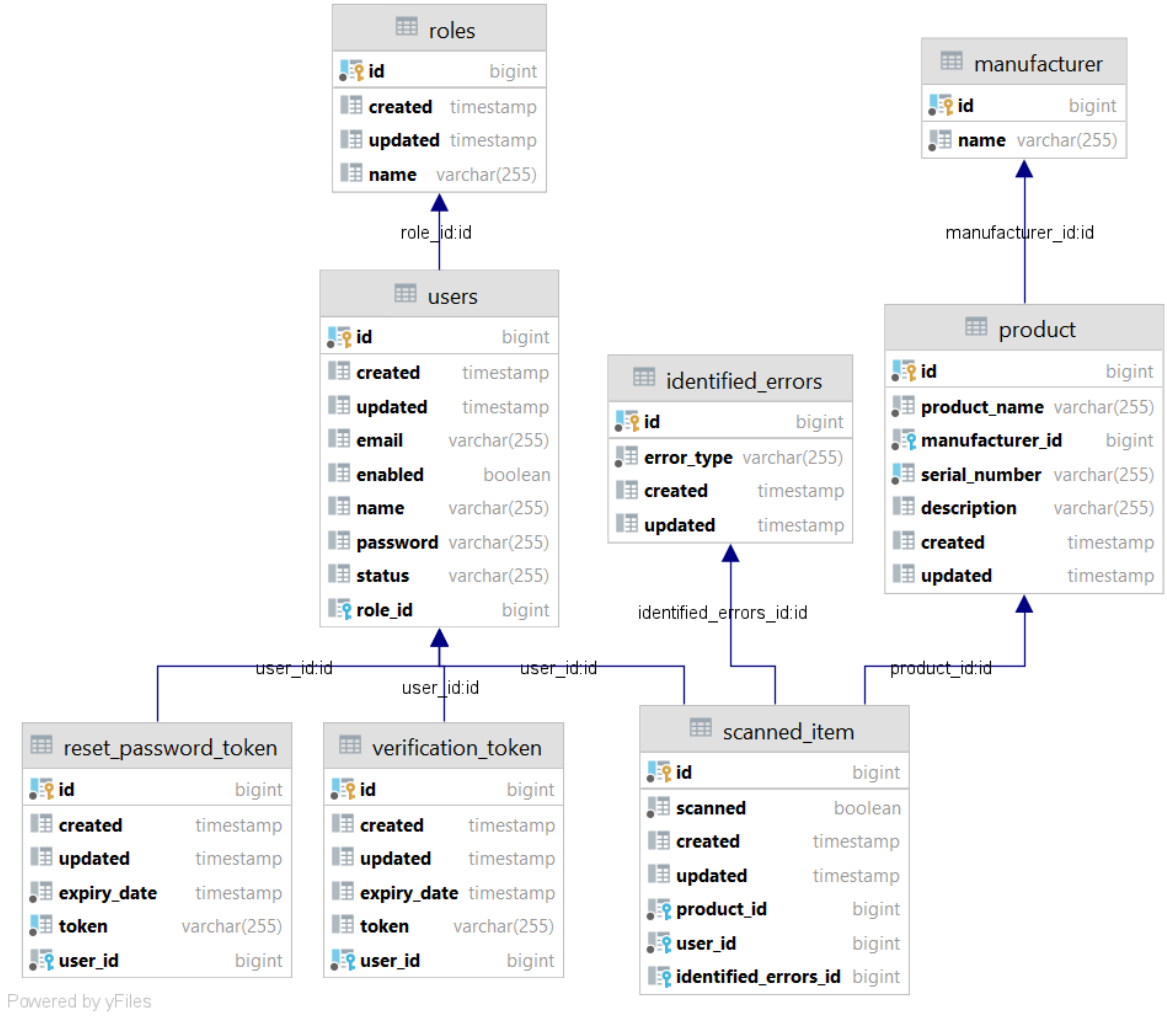

Рис. 3.1 Структура бази даних

3.3.1 Використання бібліотеки Hibernate

Для зв'язку Java-класів з таблицями БД Hibernate використовує Mapping (зіставлення, проектування), яке здійснюється за допомогою конфігураційних XML-

файлів або Java-анотацій, і забезпечує можливості по організації відносини між класами «one-to-many» і «many-to-many ». На додаток до управління зв'язками між об'єктами Hibernate може також управляти рефлексивними відносинами, де об'єкт має зв'язок «one-to-many» з іншими екземплярами свого власного типу даних.

Таким чином, використовуючи Hibernate, було створено організацію відносин між таблицями. Таблиця users відноситься до таблиці roles як “many-to-one", таблиці reset_password_token та verification_token відносяться до таблиці users як “one-to-one". Таблиця scanned_item відноситься до таблиці users як “many-to-one”, а до таблиці product і identified_errors як “one-to-one”. Таблиця product відноситься до manufacturer як “many-to-one”.

Бібліотека Hibernate, використовувана для розробки Java додатків, призначена для вирішення завдань об'єктно-реляційного відображення ORM (object-relational mapping). Hibernate це вільне програмне забезпечення з відкритим вихідним кодом, яке розповсюджується на умовах GNU Lesser General Public License.

Для використання Hibernate необхідно створити легкий у використанні каркас (фреймворк), що відображає об'єктно-орієнтовану модель даних в традиційні реляційні бази даних. Підключити бібліотеку до додатка можна як в процесі проектування java класів і sql таблиць «з нуля», так і при роботі з уже існуючою базою даних.

Hibernate забезпечує зв'язок між Java класами і таблицями бази даних, відповідність типів даних Java з типами даних SQL. Також бібліотека надає кошти для автоматичної генерації і оновлення набору таблиць, побудови запитів і обробки отриманих даних. Таким чином, при використанні Hibernate можна значно зменшити час розробки, пов'язане з ручним написанням SQL-запитів JDBC-коду.

Одним з основних переваг бібліотеки Hibernate є автоматична генерація SQL-запитів і обробка результуючого набору даних по перетворенню об'єктів, тобто виконання сериализації об'єктів. Таким чином максимально полегшується перенесення додатків на будь-які інші бази даних SQL. Тобто, Hibernate забезпечує

прозору підтримку збереження даних (persistence) для «POJO» (Plain Old Java Object). POJO клас містить тільки поля, без додаткової логіки їх обробки. Доступ до всіх полів такого класу здійснюється тільки через методи get / set.

3.4 Архітектура модулів системи

Spring Framework - один з найпопулярніших фреймворків для створення веб-додатків на Java. По суті Spring Framework являє собою просто контейнер впровадження залежностей, з декількома зручними шарами (наприклад: доступ до бази даних, проксі, аспектно-орієнтоване програмування, RPC, веб-інфраструктура MVC). Це все дозволяє швидше і зручніше створювати Java-додатки.

Spring анотації :

У більшості типових додатків ми маємо різні рівні, такі як доступ до даних, презентації, сервіс, бізнес тощо. Крім того, у кожному шарі ми маємо різні "біни". Для автоматичного виявлення цих компонентів Spring використовує анотації сканування шляху до класу. Потім він реєструє кожен компонент у ApplicationContext.

Біни - це об'єкти, які є основою програми та управляються Spring IoC контейнером. Ці об'єкти створюються за допомогою конфігураційних метаданих, які вказуються в контейнері (наприклад, XML- <bean> ... </ bean>).

Ось короткий огляд кількох із цих анотацій:

- @Component - це загальний стереотип для будь-якого компонента, керованого Spring. Ми можемо використовувати @Component у додатку, щоб позначити компоненти як керовані компоненти Spring. Spring буде збирати та реєструвати компоненти лише в @Component, а не шукати @Service та @Repository загалом.
- @Service коментує класи на рівні обслуговування. Ми позначаємо "біни" @Service, щоб вказати, що вони дотримуються бізнес-логіки. Окрім того, що він використовується в сервісному рівні, для цієї анотації немає жодного іншого спеціального використання.
- @Repository анотує класи на рівні стійкості, які будуть виконувати роль сховища бази даних. Завдання @ Repository полягає в тому, щоб ловити

винятки, характерні для стійкості, і повертати їх як один із уніфікованих неперевірених винятків Spring.

Основна різниця між цими стереотипами полягає в тому, що вони використовуються для різних класифікацій. Коли ми коментуємо клас для автоматичного виявлення, ми повинні використовувати відповідний стереотип.

Spring MVC забезпечує архітектуру паттерна Model - View - Controller (Модель - Відображення (далі - Вид) - Контролер) за допомогою слабо пов'язаних готових компонентів. Патерн MVC розділяє аспекти додатки (логіку введення, бізнес-логіку і логіку UI), забезпечуючи при цьому вільний зв'язок між ними.

- Model (Модель) інкапсулює (об'єднує) дані програми, в цілому вони будуть складатися з POJO (Java-об'єктів, або бінів).
- View (Відображення, Вид) відповідає за відображення даних моделі, - як правило, генеруючи HTML, які ми бачимо в своєму браузері.
- Controller (Контролер) обробляє запит користувача, створює відповідну Модель і передає її для відображення в Вид.

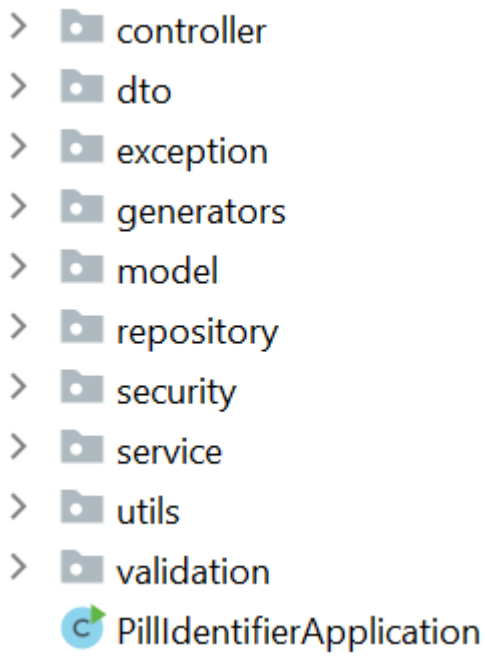

Рис. 3.2 Структура проекту

- DTO. При отриманні з репозиторію об'єкта типу Entity(сутності) ми не можемо використовувати його в контролері. Таким чином, для цього був придуманий шаблон під назвою DTO - data transfer object, який можна використовувати в сервісі або в контролері. Це один із шаблонів архітектури корпоративних

програм, який вимагає використання об’єктів, які об’єднують та інкапсулюють дані для передачі. Він не повинен містити жодної бізнес-логіки, але повинен містити механізми серіалізації та десеріалізації.

- Security - це Java / JavaEE framework, що надає механізми побудови систем автентифікації та авторизації, а також інші можливості забезпечення безпеки для корпоративних додатків, створених за допомогою Spring Framework.

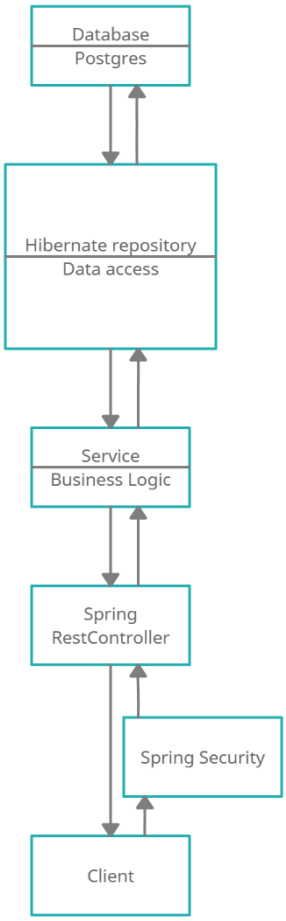

Рис. 3.4 Функціональна схема системи

Зв’язки класів можна переглянути на діаграмі класів(див. Додаток А)

3.5 Реалізація API

API - це інтерфейс прикладного програмування або програмний інтерфейс додатку, за допомогою якого одна програма може взаємодіяти з іншою. API дозволяє передавати інформацію напряму з однієї програми в другу, не зачіпаючи інтерфейс

взаємодії з користувачем. API може бути внутрішнім, приватним - коли програмні компоненти пов'язані між собою і використовуються всередині системи. А може бути відкритим, публічним - в такому випадку він дозволяє зовнішнім користувачам або іншим програмам отримувати інформацію, яку можна інтегрувати в свої додатки. Щоб програми спілкуватися між собою, їх API потрібно побудувати за єдиним стандартом. Одним з них є REST - стандарт архітектури взаємодії додатків і сайтів, що використовує протокол HTTP. Особливість REST в тому, що сервер не запам'ятовує стан користувача між запитами. Іншими словами, ідентифікація користувача (авторизаційний токен) і всі параметри виконання операції передаються в кожному запиті. Цей підхід настільки простий і зручний, що майже витіснив всі інші.

Тестування API проводять, ґрунтуючись на бізнес-логіці програмного продукту. Тестування API відноситься до інтеграційного тестування, а значить в ході нього можна знайти помилки взаємодії між модулями системи або між системами. Для тестування використовують спеціальні інструменти, де можна відправити вхідні дані в запиті і перевірити точність вихідних даних. Одним з таких інструментів як раз і є Postman. Ось що він вміє:

- Складати і відправляти запити;
- Зберігати запити в папках і колекціях;
- Параметризовувати запити;
- Додавати до виклику API контрольні точки;
- Створювати різні оточення для одних і тих же запитів;
- Запускати колекції за допомогою Collection Runner і використовувати їх як автоматизовані тести

При реалізації API було створено контролери різних видів для обробки запитів протоколу HTTP: Get, Post, Put, Delete(Додаток Б). Основні з них :

- POST auth/registration (реєстрація користувача). Потрібно вказати ім’я, пошту, пароль, підтвердження паролю і роль в системі. Після реєстрації адміністратор системи повинен активувати акаунт. (Додаток Б рисунок 3.5.4)

- POST auth/login (вхід в систему). Потрібно вказати пошту і пароль акаунту. Якщо акаунт неактивований наступні дії будуть неможливі. У випадку успішного входу надсилається токен, з допомогою якого можна продовжити роботу в системі. (Додаток Б рисунок 3.5.3)
- POST products/upload (завантаження кодів ліків з файлу типу .csv). У випадку, якщо буде змога завантажити файл з інформацією, яка вже існує в базі, з'являться дані, які повторюються і не збережуться повторно. (Додаток Б Рисунок 3.5.15)
- GET supervisor/get-all (отримання інформації про помилки). Якщо при скануванні було виявлено дублікат ліків, система зберігає запит і скориставшись supervisor/get-all можна отримати всі дані по ним(Додаток Б Рисунок 3.5.14)

  Приклад :


```
{
    "scanningTime": "2021-06-10T08:21:28.778+00:00",
    "productName": "Aspirin2",
    "manufacture": "Aspirin",
    "serialNumber": "111111112",
    "description": "Test2",
    "productCreated": "2021-06-10T08:00:24.743+00:00",
    "errorType": "duplicate",
    "userEmail": "t.r.kuziv26@gmail.com",
    "username": "Tetyanatest"
}
```


- GET scan/serial-number(сканування ліків). Користувач код ліків, система розшифровує складові коду і перевіряє на наявність таких в базі даних. Також перевіряє чи не були продані ці ліки раніше, чи не протрочений термін придатності і чи не було повернень. Якщо було знайдено помилки, вони додаються в базу даних. При умові, що такого коду немає в базі даних, він записується в окрему таблицю невідомих номерів. (Додаток Б Рисунок 3.5.8)

В даному випадку для кожного типу запитів в контролері визначено свої методи. Так, метод Get () обробляє запити типу GET і повертає колекцію об'єктів з БД. Якщо запит Get містить параметр id (ідентифікатор об'єкта), то він обробляється іншим методом - Get (int id), який повертає об'єкт по переданому id. Запити типу Post обробляються методом Post (User user), який отримує з тіла запиту відправлені дані і додає їх в базу даних. Метод Put (User user) обробляє запити типу Put - отримує дані з запиту і змінює ними об'єкт в базі даних. Метод Delete (int id) обробляє запити типу Delete, тобто запити на видалення - отримує із запиту параметр id і по цим ідентифікаторам видаляє об'єкт з БД.

3.6 Тестування системи

При розробці системи одним з головних етапів є тестування, без нього продукт не може вважатися якісним, тому що дефекти існують у кожному програмному забезпеченні. Для покращення якості системи було реалізовано Unit-тестування, яке ще називається Модульним тестуванням. Це тип тестування програмного забезпечення, при якому тестуються окремі модулі або компоненти програмного забезпечення. Його мета полягає в тому, щоб перевірити, що кожна одиниця програмного коду працює належним чином. Модульні тести ізолюють частину коду і перевіряють його працездатність. Одиницею для виміру може служити окрема функція, метод, процедура, модуль або об'єкт.

Рис. 3.5 Структура тестів

Було написано 35 Unit-тестів для перевірки сервісів ResetPasswordServiceImpl(скидання паролю), RoleServiceImpl, UserServiceImp, VerificationTokenServiceImpl:

- Клас ResetPasswordServiceTests перевіряє ResetPasswordServiceImpl

1. Тест findByTokenTest()
2. Тест findByTokenEntityNotFoundExceptionTest()
3. Тест findTokenByUserTest()
4. Тест findTokenByUserEntityNotFoundExceptionTest()
5. Тест saveTokenTest()
6. Тест updateTokenTest()
7. Тест updateTokenNullEntityReferenceExceptionTest()
8. Тест updateTokenUserNullEntityReferenceExceptionTest()

- Клас RoleSerciceTests перевіряє RoleServiceImpl

1. Тест getRoleByIdTest()
2. Тест getRoleByIdEntityNotFoundExceptionTest()
3. Тест getRoleByNameTest()
4. Тест getRoleByNameEntityNotFoundExceptionTest()
5. Тест getEmptyRoleListTest()
6. Тест getRoleListTest()

- Клас UserServiceTests перевіряє UserServiceImp :

1. Тест exceptionWhenCreateUserWithExistingEmail()
2. Тест createUser()
3. Тест findByEmailEntityNotFoundExceptionTest()
4. Тест FindByEmailTest()
5. Тест getEmptyListOfUsersTest()
6. Тест getAllUsersTest()
7. Тест findUserByLoginAndPasswordTest()
8. Тест findUserByLoginAndWrongPasswordTest()
9. Тест findUserByIdEntityNotFoundExceptionTest()
10. Тест findUserByIdTest()
11. Тест getUsersByStatus()
12. Тест getEmptyUsersByStatus()
13. Тест updateUserStatusNullEntityReferenceExceptionTest()

- Клас VerificationTokenServiceTests перевіряє VerificationTokenServiceImpl

1. Тест findByTokenTest()
2. Тест findByTokenAccessDeniedExceptionTest()
3. Тест findByUserTest()

4. Тест findByUserEntityNotFoundExceptionTest()
5. Тест saveTokenTest()
6. Тест updateTokenTest()
7. Тест updateTokenNullEntityReferenceExceptionTest()
8. Тест updateTokenUserNullEntityReferenceExceptionTest()

Висновки до розділу 3

В даному розділі описано засоби розробки системи, її реалізацію та етапи створення. Було представлено функціональну схему системи, діаграму класів, структуру бази даних і їхніх зв’язків. Описано реалізацію API, створення бази даних, архітектуру модулів системи та модульне тестування.

…….

ВИСНОВКИ

В даній дипломній роботі було розглянуто і проаналізовано такі теми, як : види баз даних, види штрих-кодів. Особливу увагу було приділено штрих-коду DataMatrix, його створенню, декодуванню та складовим частинам.

У дипломній роботі описано хід розробки системи : реалізація API, створення бази даних, архітектуру модулів системи та модульне тестування. Також було описано середовища, мову розробки, використані бібліотеки і фреймворки.

Дана система захисту від реалізації підробок в аптеках дозволяє знизити постачання та продаж фальсифікованих ліків за допомогою унікальних кодів на кожній упаковці ліків. Дана система може використовуватись в цілях державного забезпечення для захисту громадян від підроблених ліків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. https://topjava.ru/blog/spring-framework-vs-spring-boot-differences
2. Clean Code : A Handbook of Agile Software Craftsmanship – Robert C. Martin
3. https://traceway.ru/company/articles/sravnenie-datamatrix-i-qr-code/
4. https://www.mulesoft.com/resources/api/what-is-an-api
5. REST API Design Rulebook – Mark Masse
6. https://hostiq.ua/wiki/ukr/database/
7. https://www.vostok.dp.ua/infa1/shtrih-kod/shtrih_kod_1d_2d/
8. https://landing.ua/blog/sajty-na-java-plyusy-i-minusy.html
9. https://web-creator.ru/articles/postgresql
10. https://bitbucket.org/product/ru/guides/getting-started/overview
11. https://netology.ru/blog/trello
12. https://logrocon.ru/news/unit_testing
13. https://xakep.ru/2014/04/21/62386/
14. https://proselyte.net/tutorials/hibernate-tutorial/architecture/

ДОДАТКИ

Додаток А

Рисунок А.1 - Діаграма класів

Додаток Б

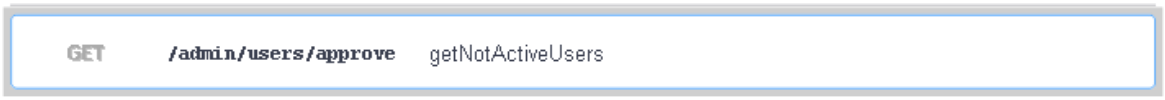

Рис. 3.5.1 GET admin-activate-user-controller (Admin Activate User Controller)

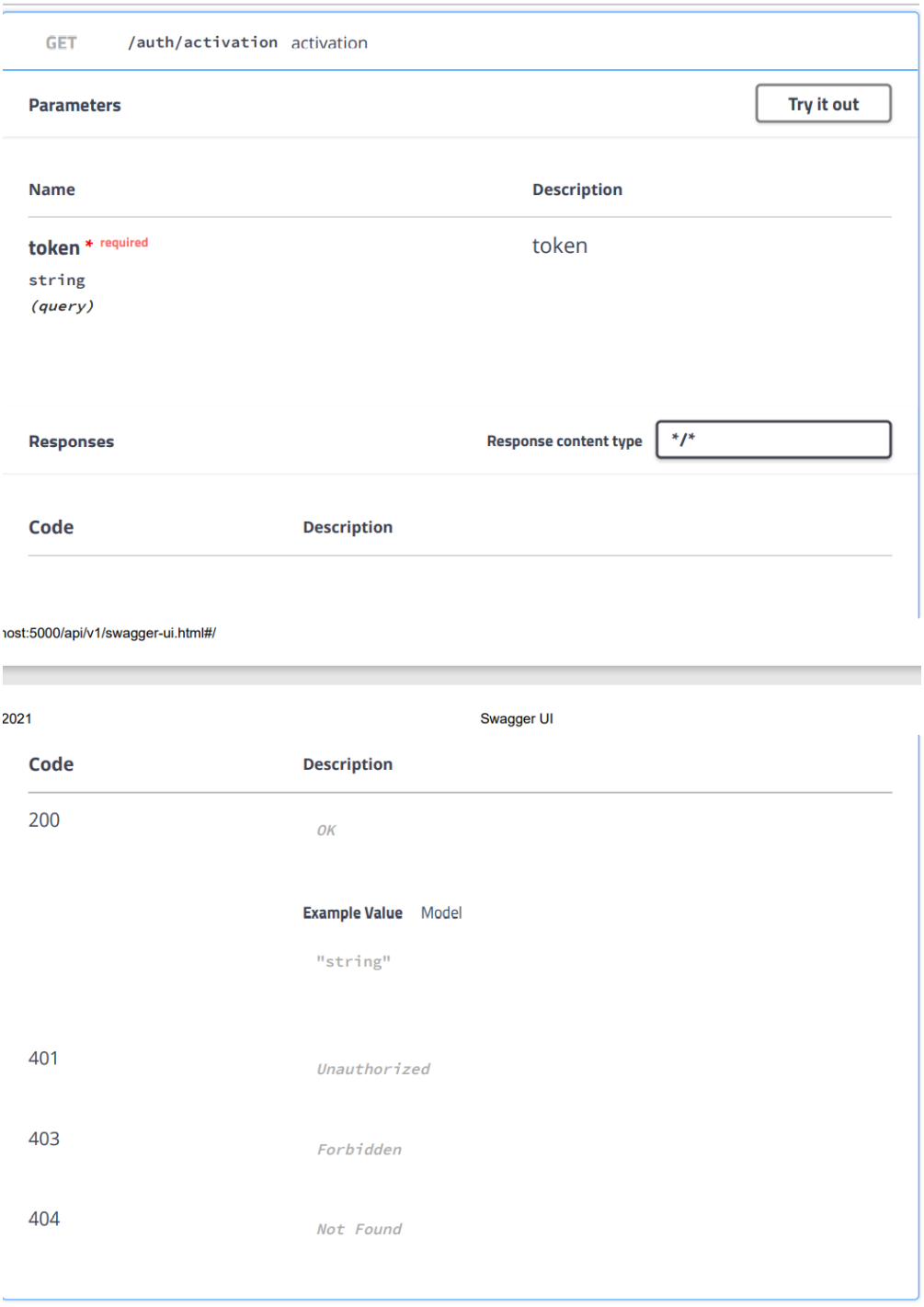

Рис. 3.5.2 GET auth-controller (Auth Controller)

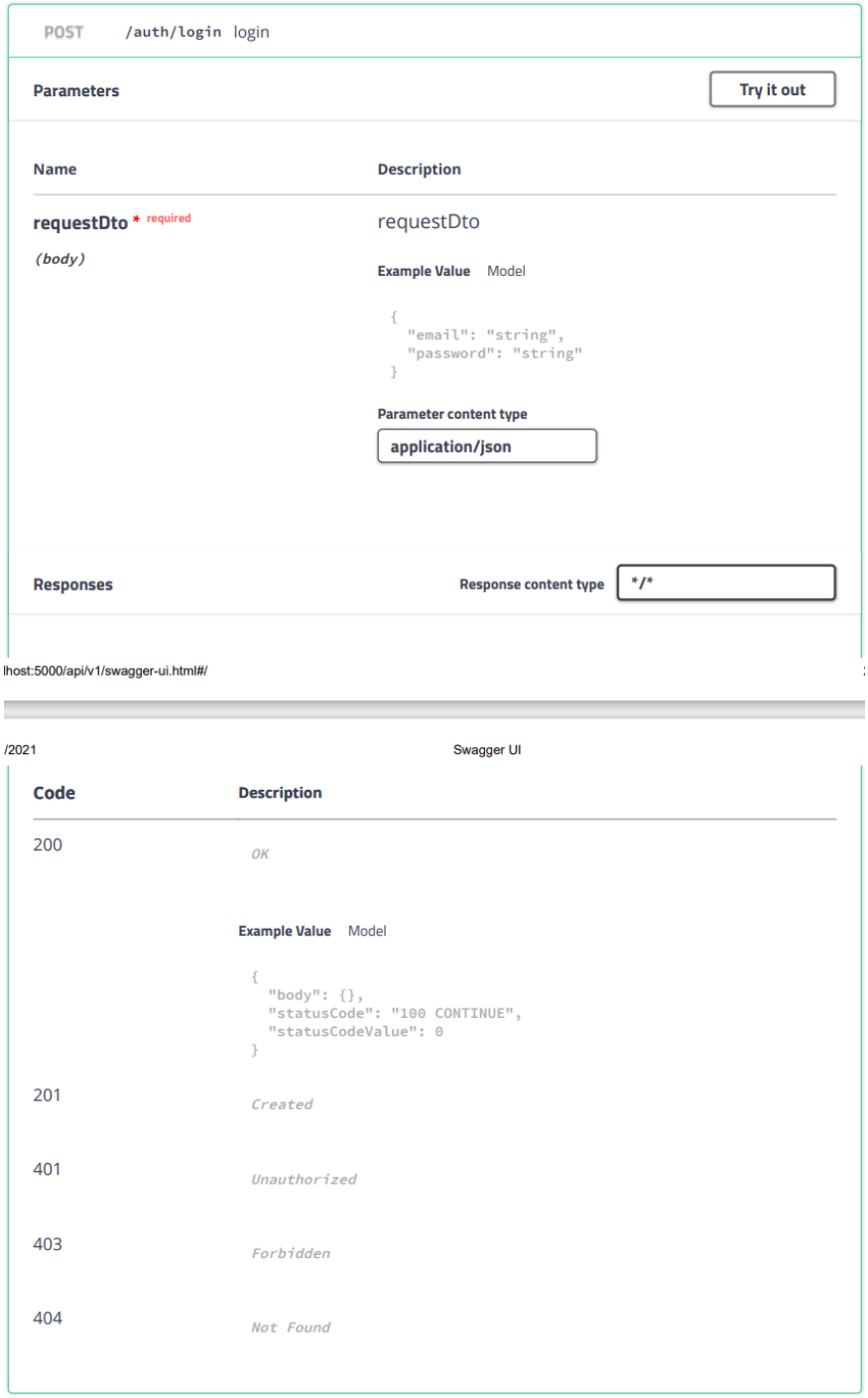

Рис. 3.5.3 POST login

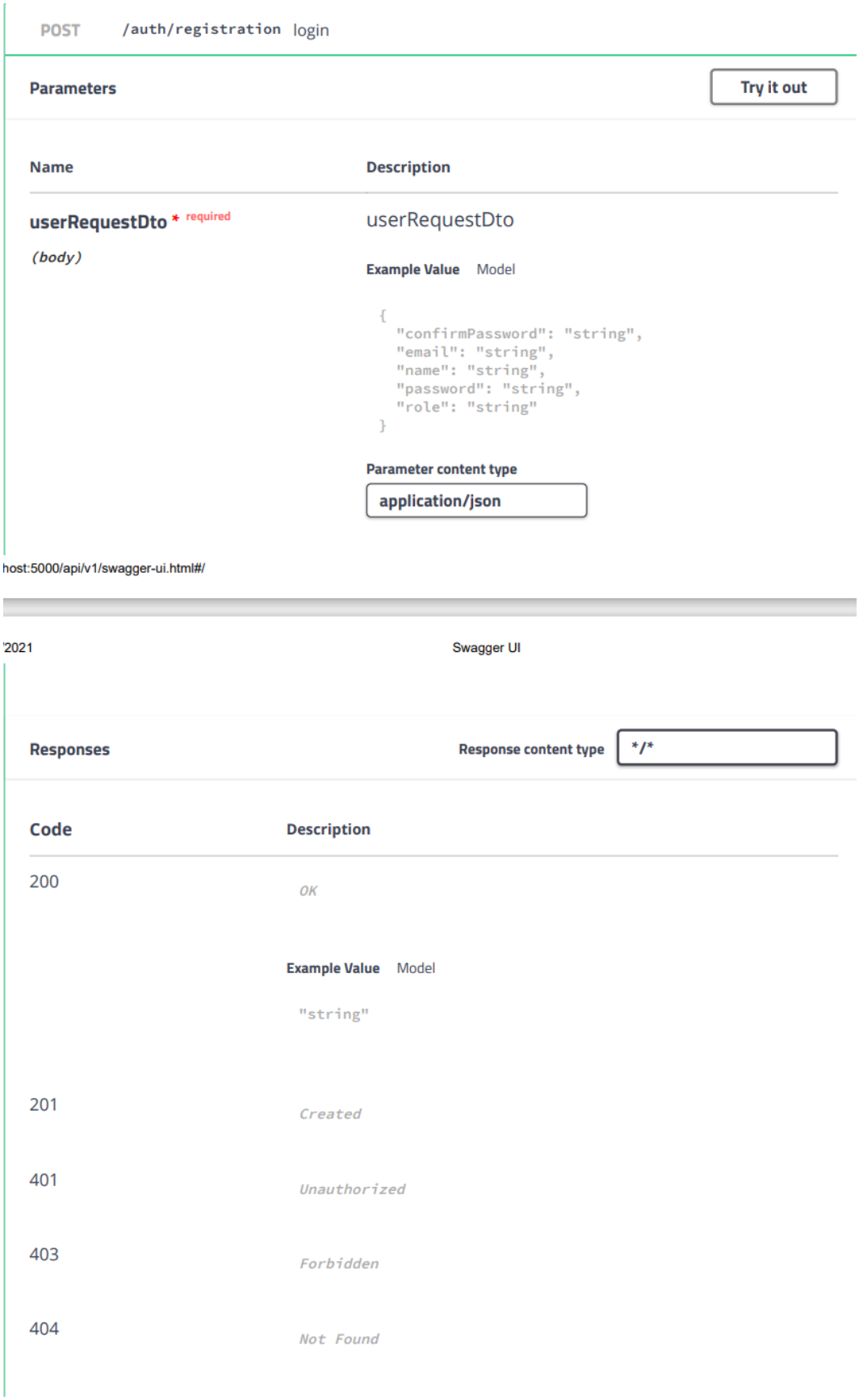

Рис. 3.5.4 POST registration

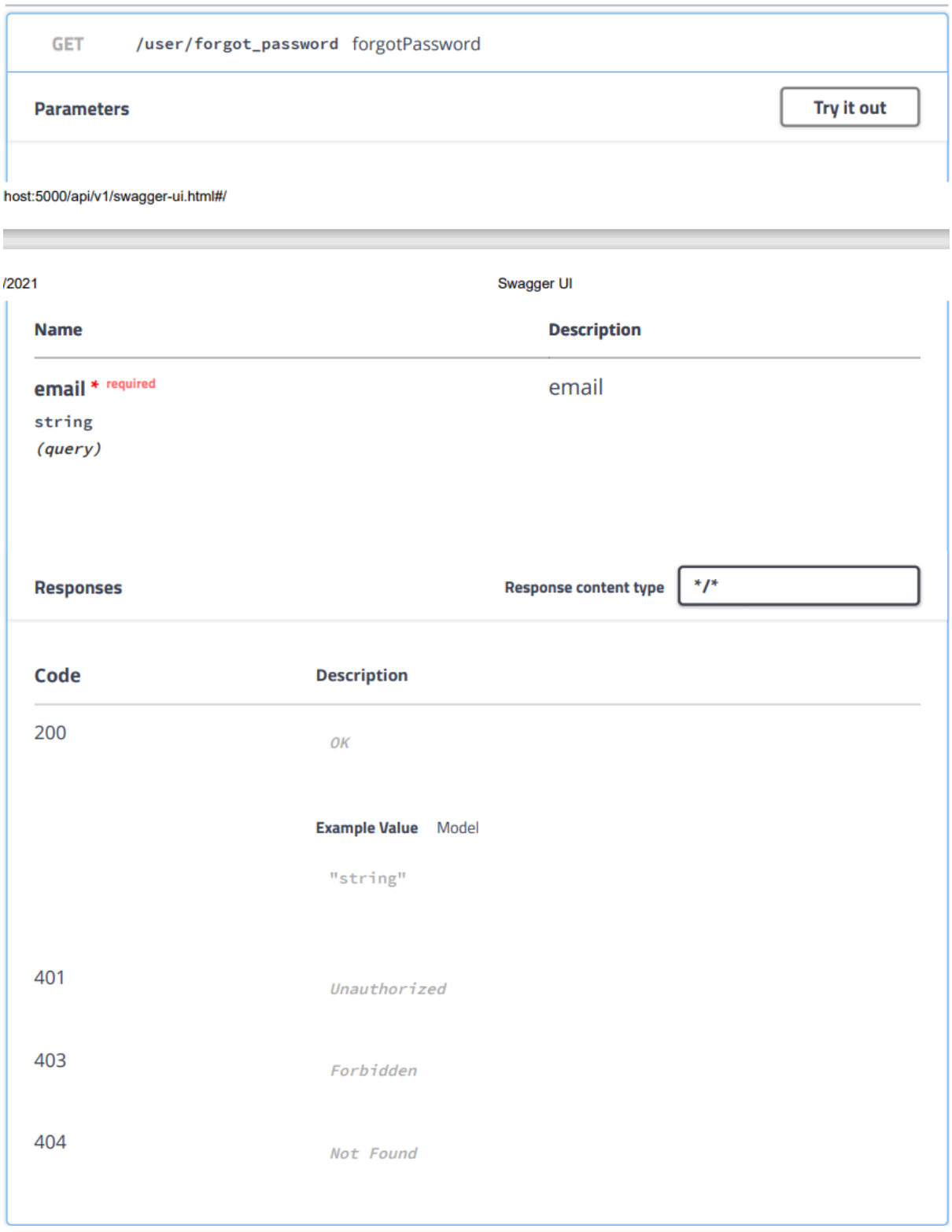

Рис. 3.5.5 GET forgot-password-controller Forgot Password Controller

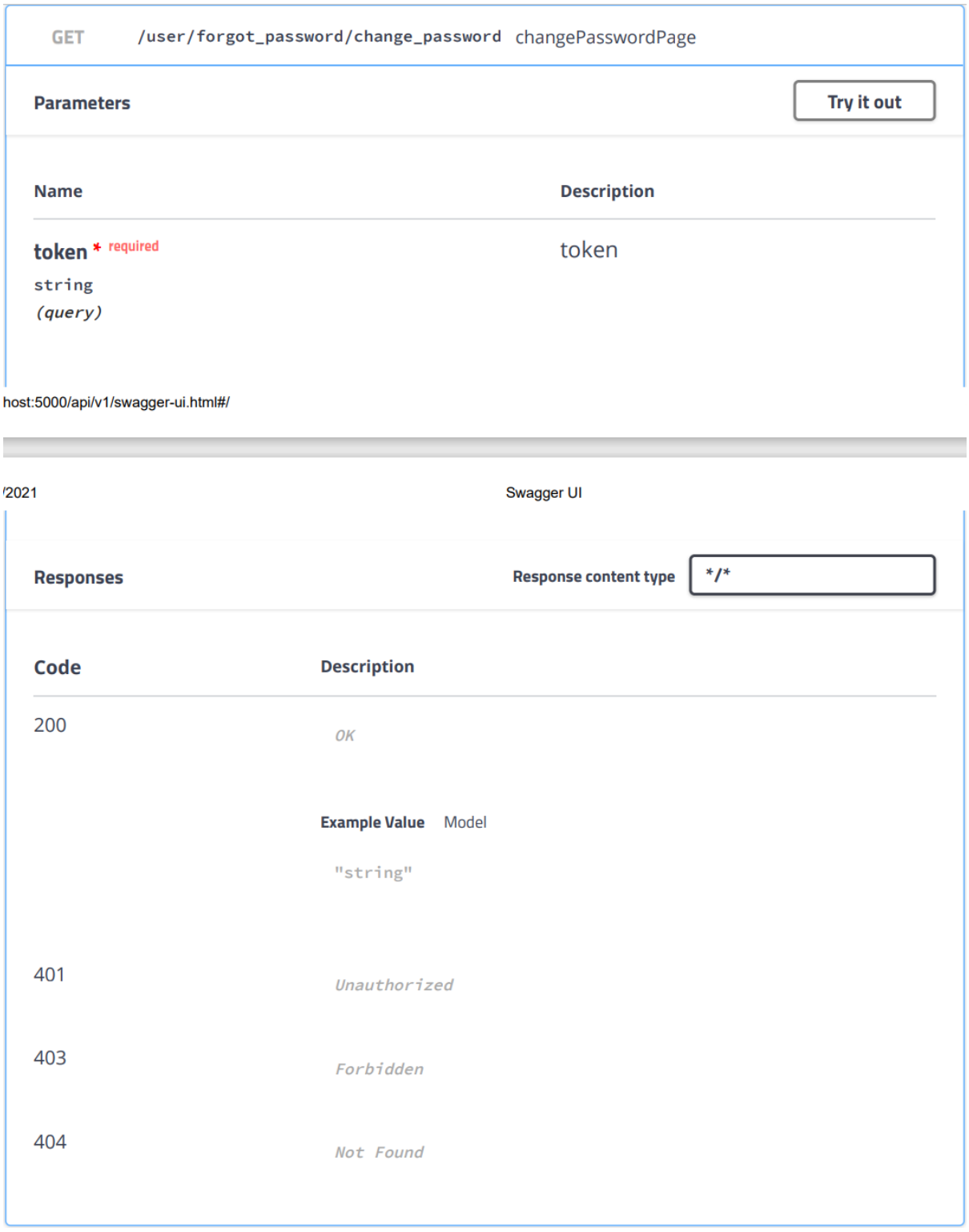

Рис. 3.5.6 GET forgot-password-controller (Change Password)

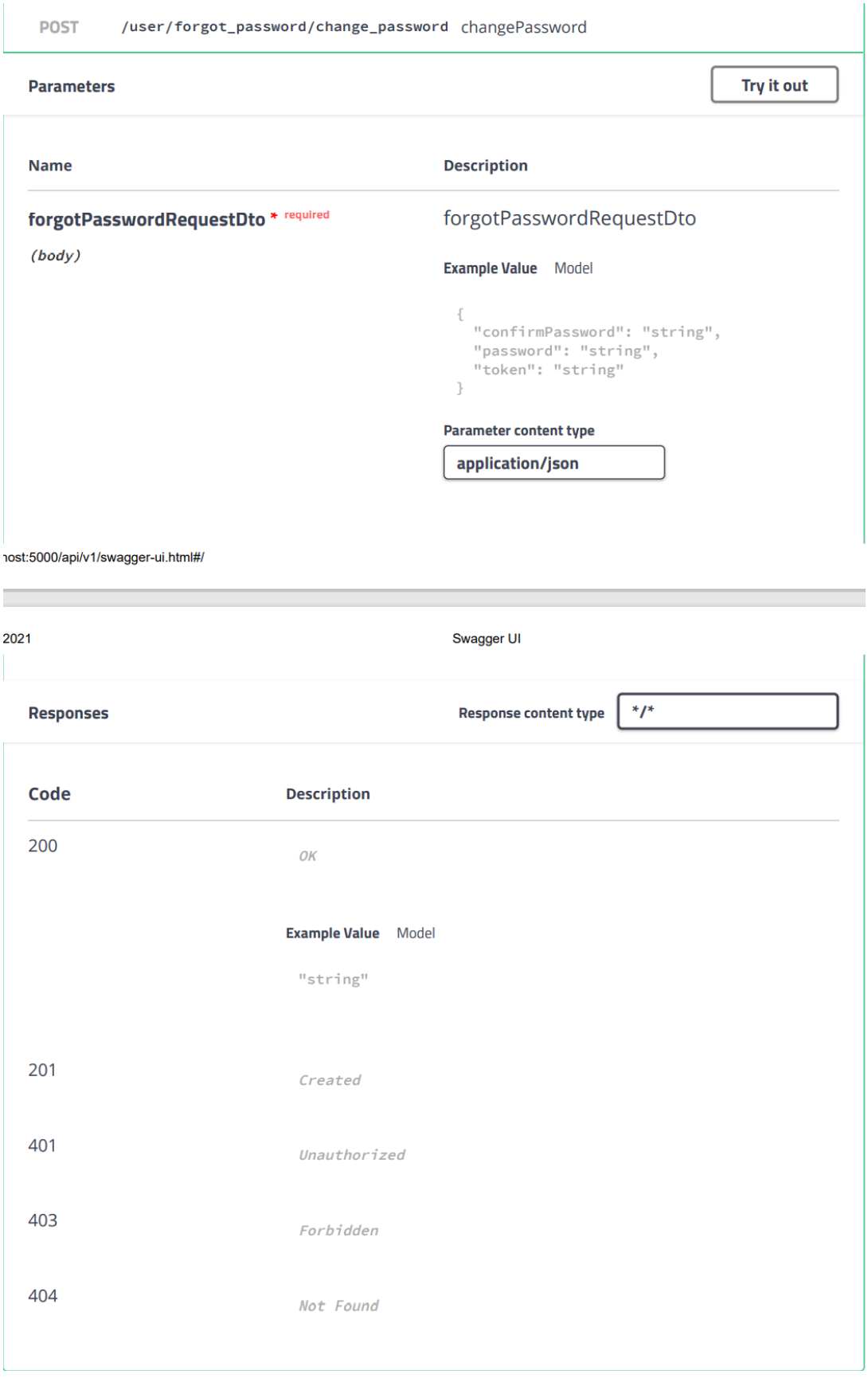

Рис. 3.5.7 POST registration changePassword

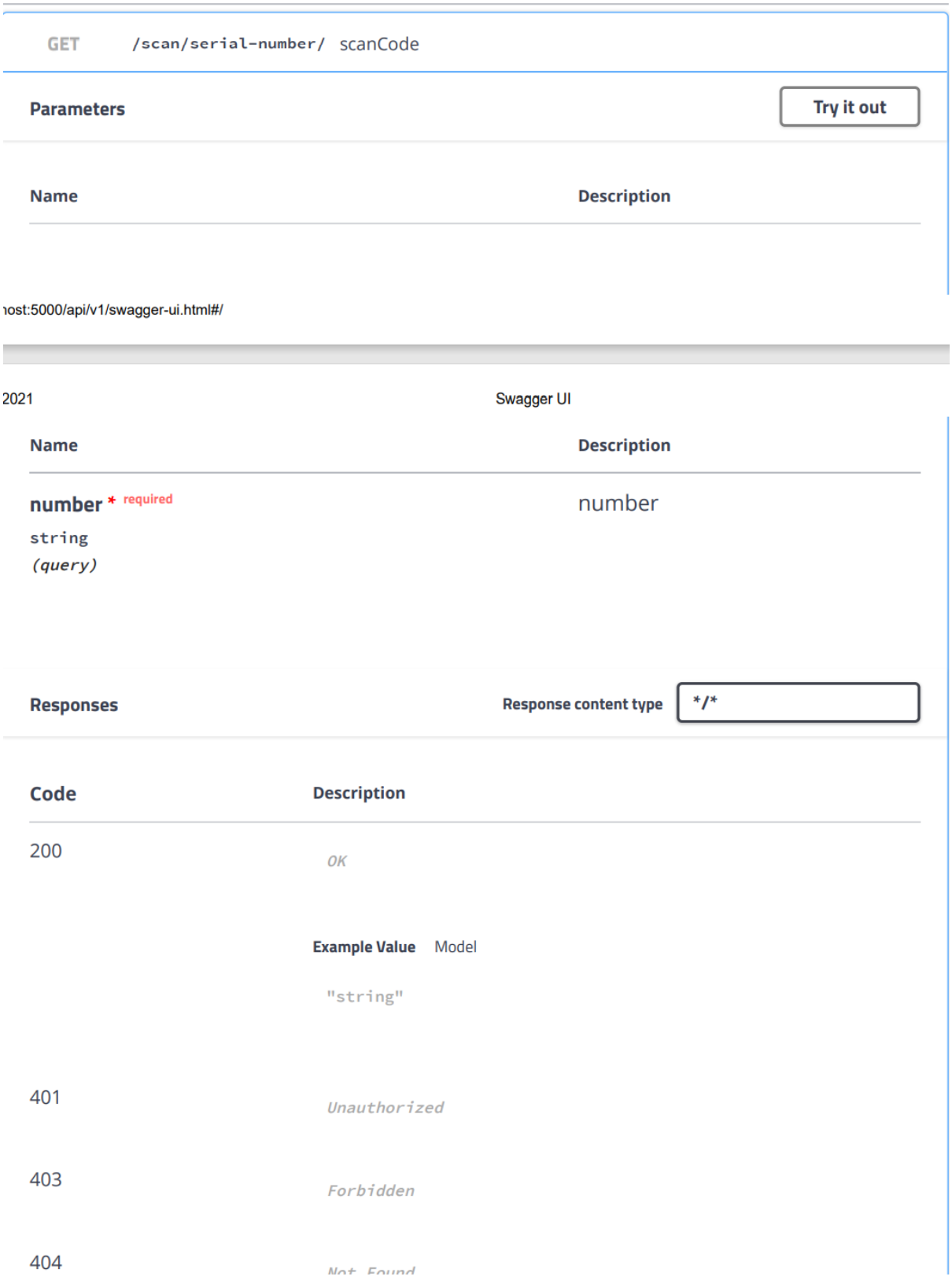

Рис. 3.5.8 GET scan-barcode

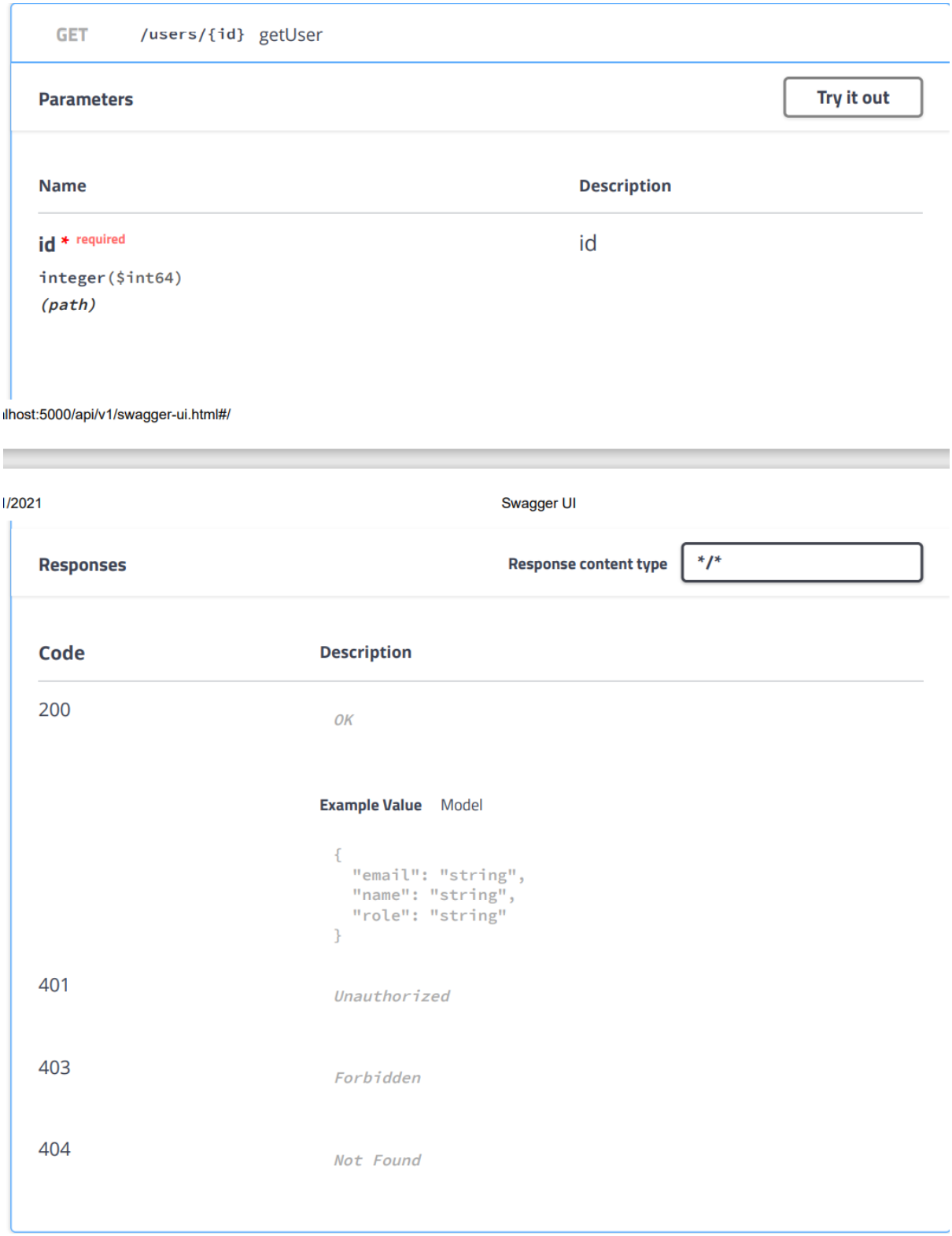

Рис. 3.5.9 GET user-controller

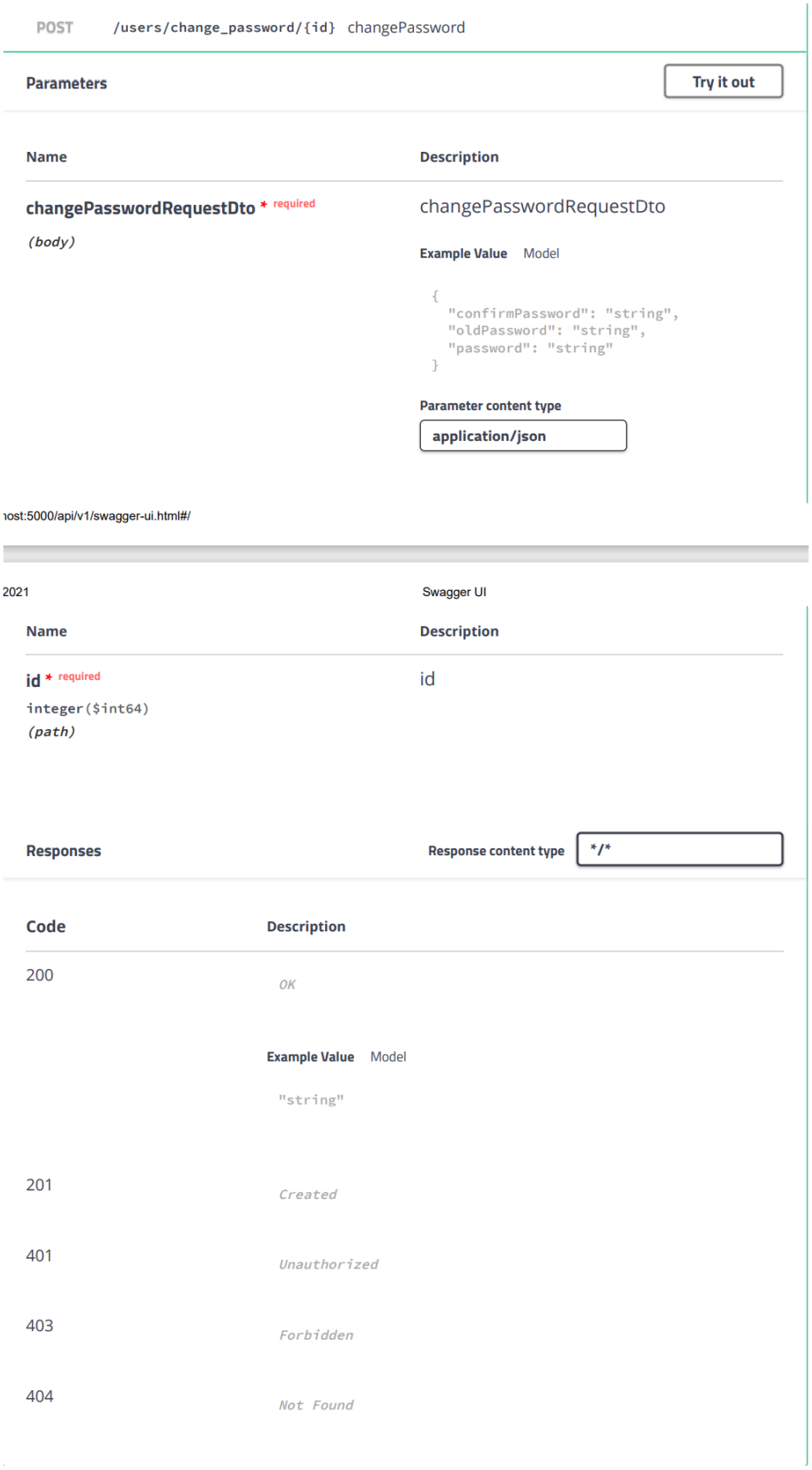

Рис. 3.5.10 POST changepassword

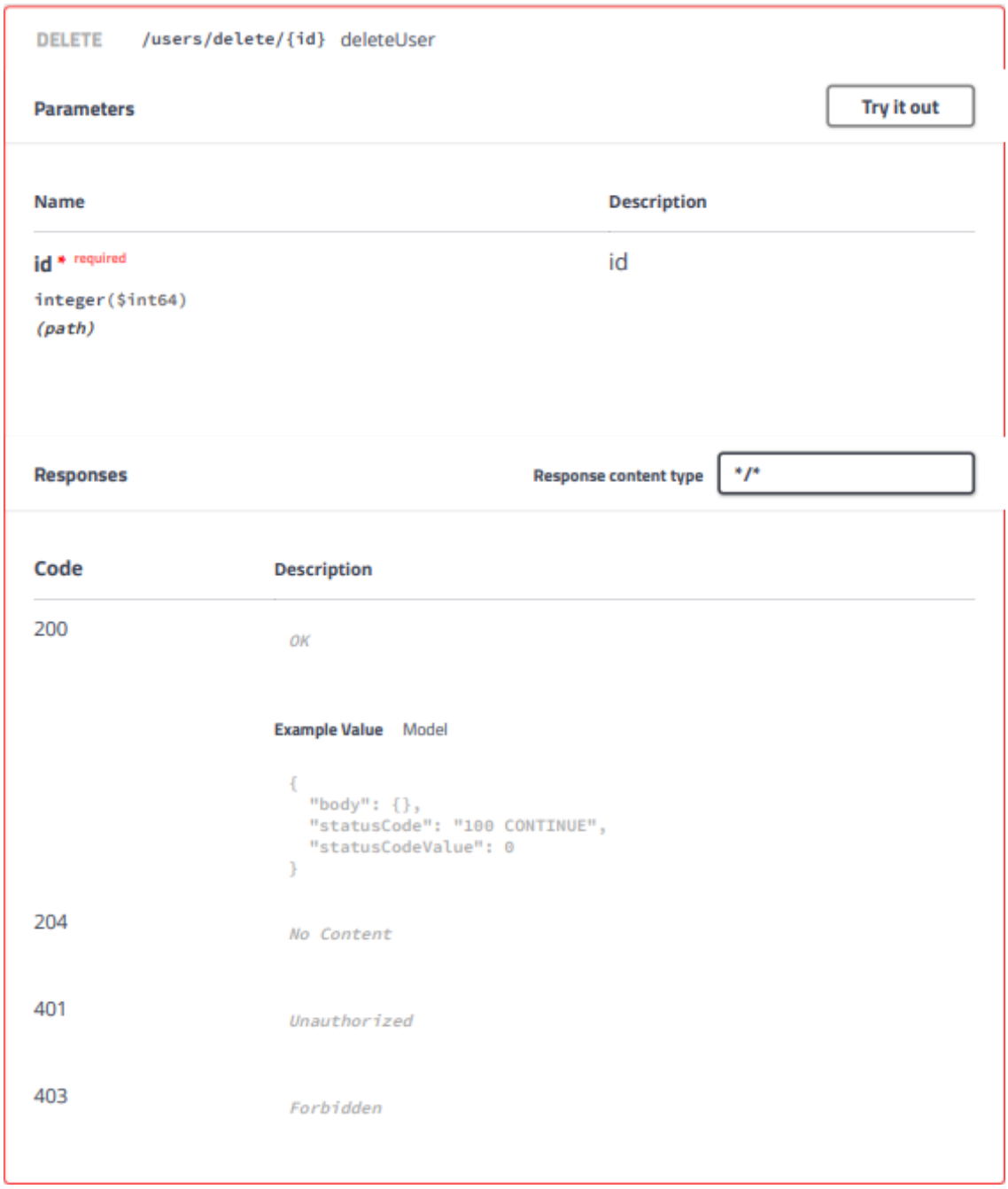

Рис. 3.5.11 DELETE deleteUser

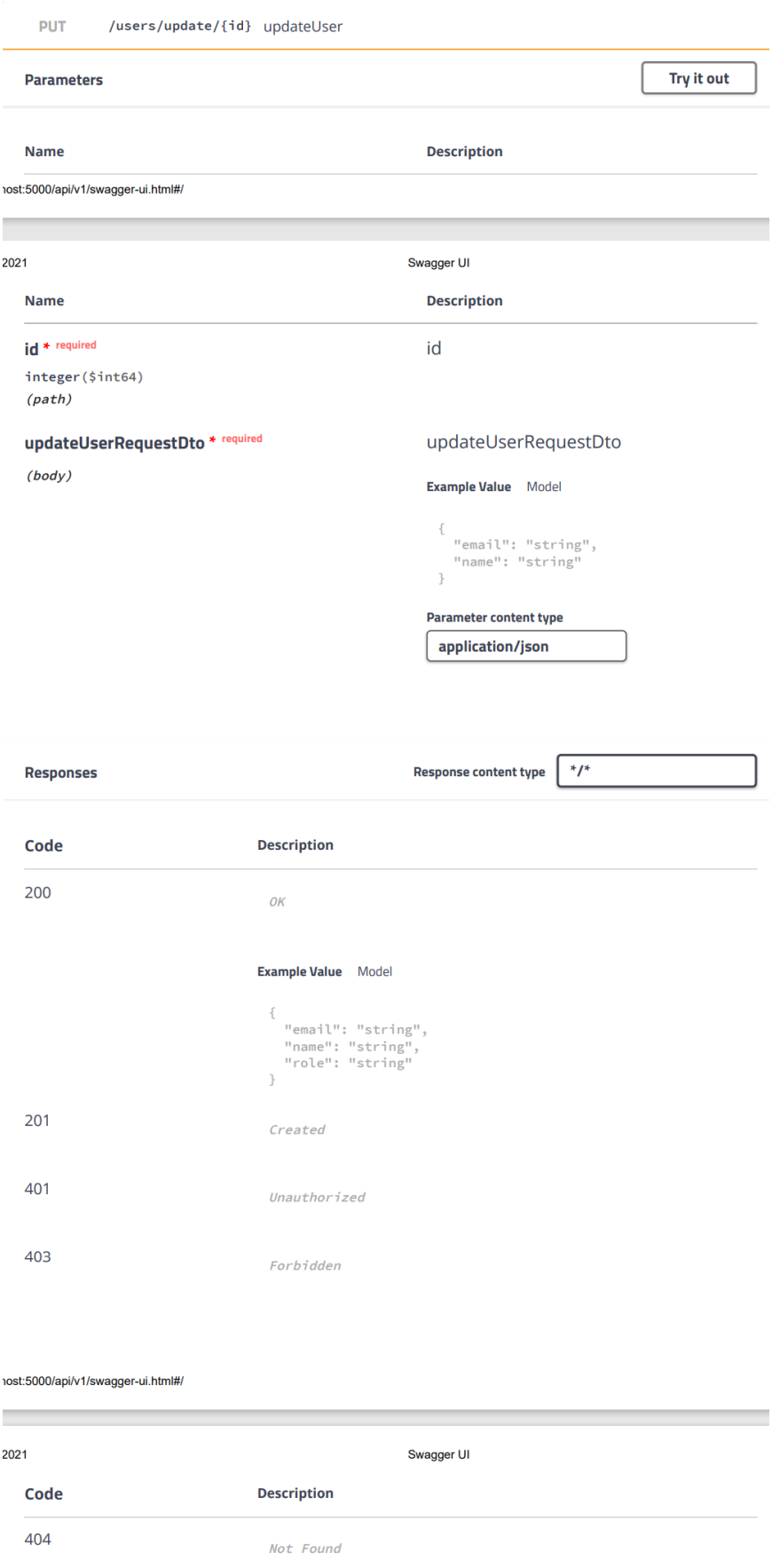

Рис. 3.5.12 PUT updateUser

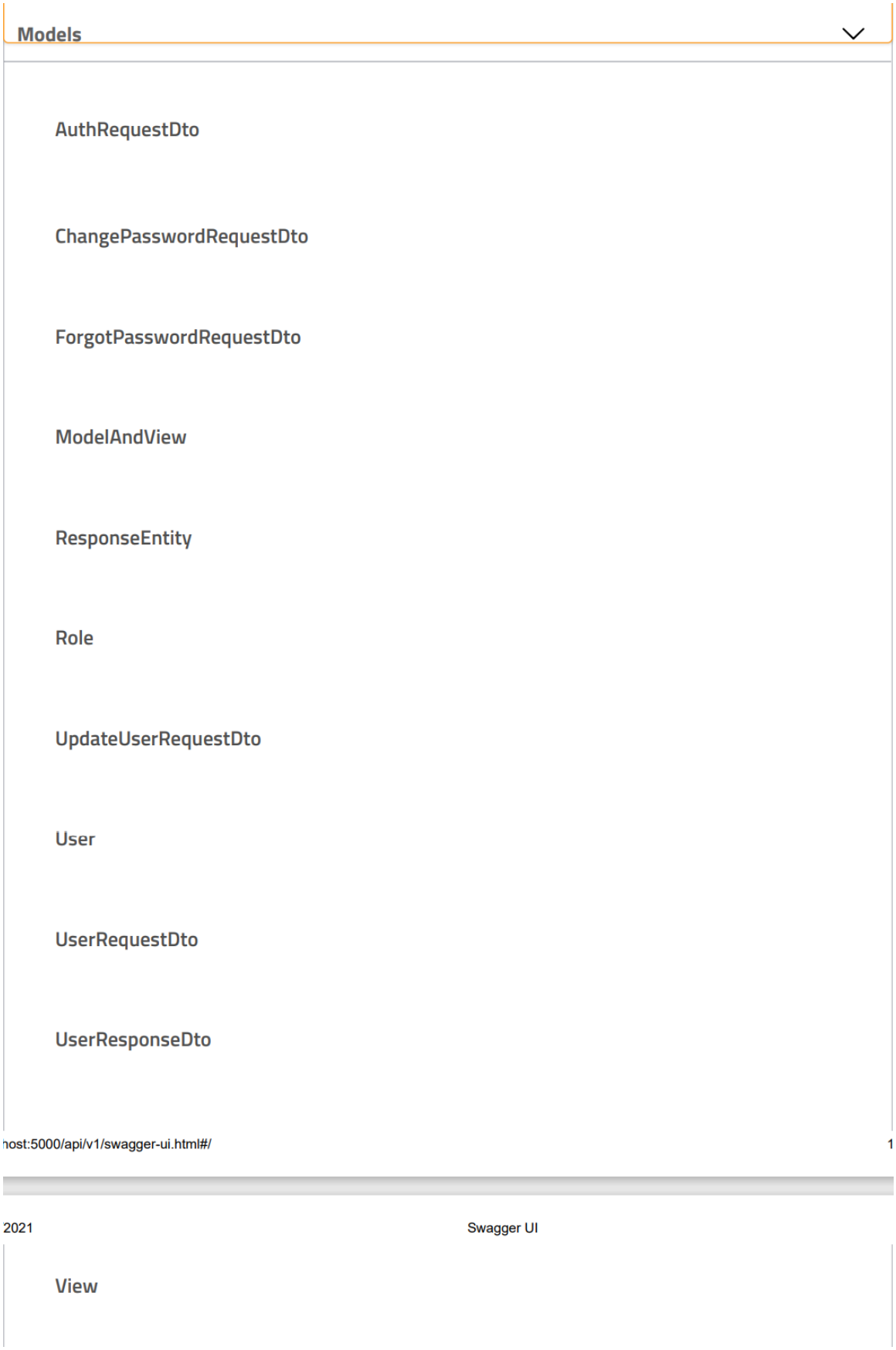

Рис. 3.5.13 Моделі

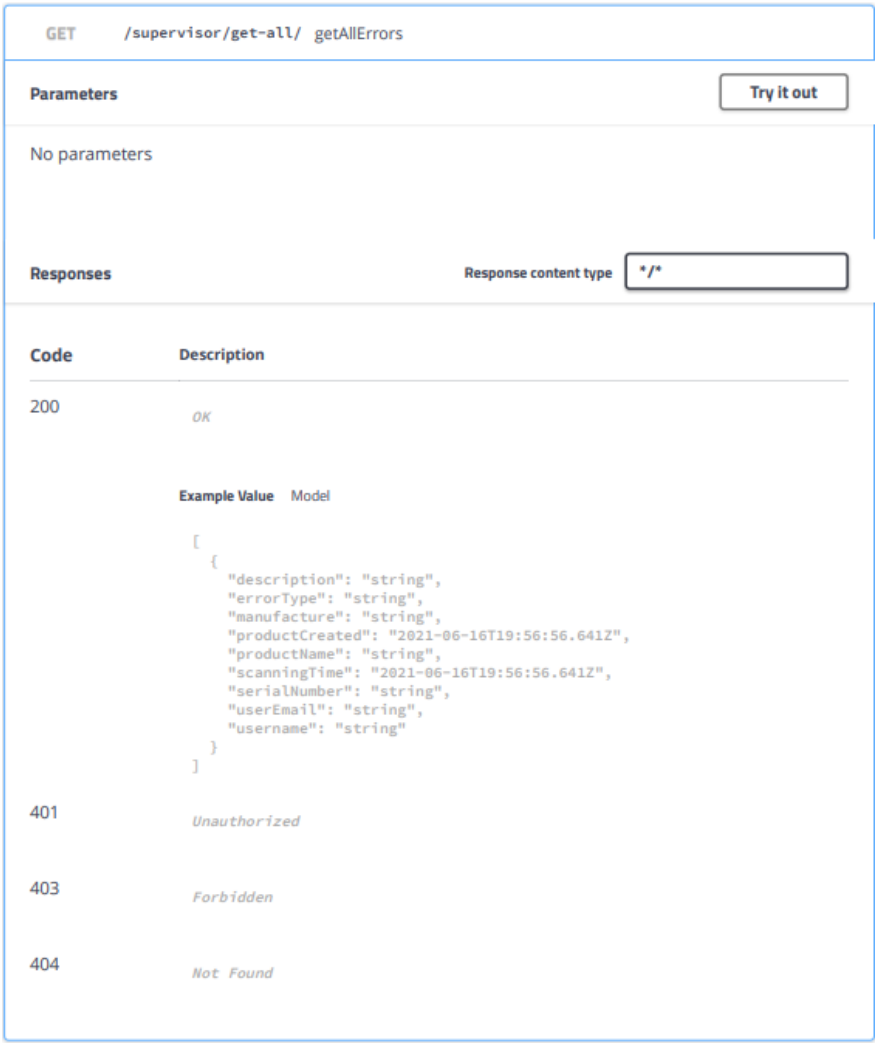

Рис. 3.5.14 supervisor-controller

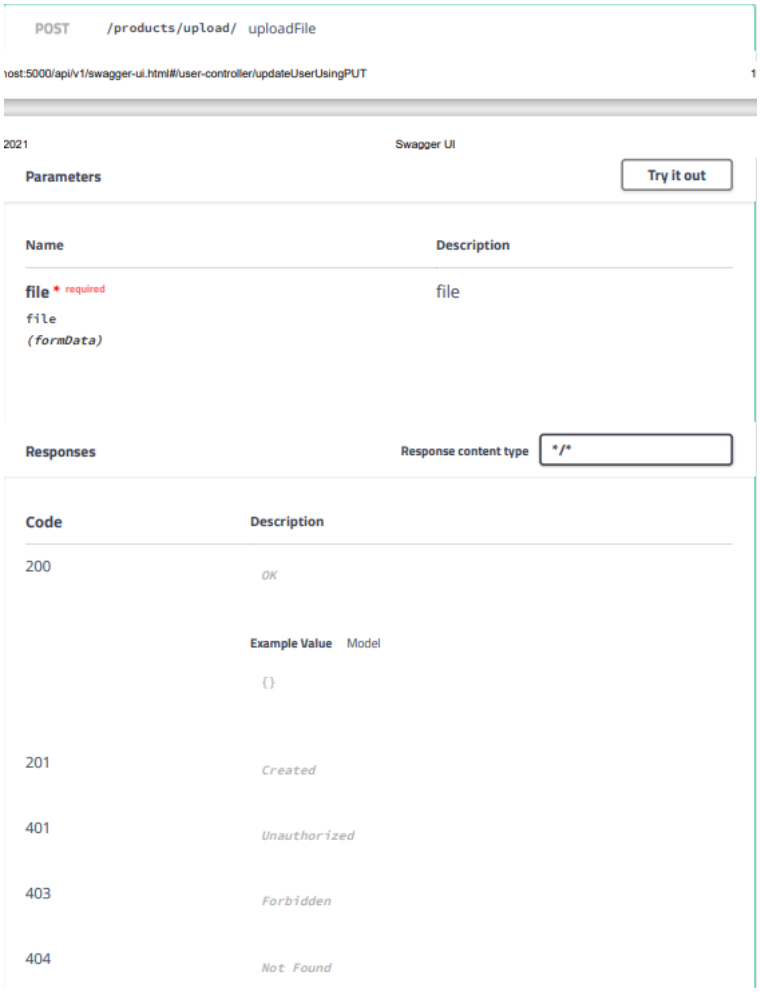

Рис. 3.5.15 upload-products-controller