ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ І ЗВ’ЯЗКУ

Навчально-науковий інститут інфокомунікацій та програмної інженерії
Кафедра інформаційних технологій

Пояснювальна записка

до випускної кваліфікаційної роботи
здобувача освітнього ступеню «бакалавр»
на тему ІНФОРМАЦІЙНЕ ВИРІШЕННЯ ТА СЕРВІС ДЛЯ РОСЛИННОГО РИНКУ

Виконав: студент 4 курсу, групи ІПЗ-4.2.01ТЕ
спеціальності
121 Інженерія програмного забезпечення
__________Голота Я.О.________________
Керівник _____Чепок А.О.______________
Рецензент ____________________________

Одеса – 2023 р.

Д О В І Д К А

кафедри ІТ про виконану бакалаврську роботу

студента 4 курсу ННІ ІКПІ групи ІПЗ-4.2.01ТЕ

Голоти Ярослава Олеговича

на тему Інформаційне вирішення та сервіс для рослинного ринку

Висновок нормоконтролера ___________________________________

____

Нормоконтролер ____________________ ________ __________

(науковий ступінь, вчене звання, посада) (підпис, дата) (і. б. прізвище)

Висновок відповідального за наявність плагіату __________________

____

Відповідальна особа ____________________ ________ __________

(науковий ступінь, вчене звання, посада) (підпис, дата) (і. б. прізвище)

Попередня експертиза (захист) __бакалаврської роботи___________

(бакалаврської роботи чи магістерської роботи)

студ. ______Голота Я.О.______ проведена "_____"____________ 20__ р.

(прізвище і.б.)

Висновки ____

____

____

____

____

____

____

Члени комісії ______________ ______________________________

(підпис) (науковий ступінь, вчене звання,посада, прізвище і.б.)

______________ ______________________________

(підпис) (науковий ступінь, вчене звання,посада, прізвище і.б.)

______________ ______________________________

(підпис) (науковий ступінь, вчене звання,посада, прізвище і.б.)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНТЕЛЕКТУАЛЬНИХ
ТЕХНОЛОГІЙ І ЗВ'ЯЗКУ

Навчально-науковий інститут інфокомунікацій та програмної інженерії_____
Кафедра ______інформаційних технологій_______________________________
Освітній ступінь _____бакалавр______________________________________
Галузь знань _____12 Інформаційні технології ___________________________
Спеціальність 121 Інженерія програмного забезпечення___________________

ЗАТВЕРДЖУЮ
В. о. завідувача кафедрою ІТ
к.т.н., доц.______С.М. Вороной
“ ____ ” ________ 20__ року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

_____Голота Ярослав Олегович_____

1. Тема роботи **Інформаційне вирішення та сервіс для рослинного ринку**____

керівник роботи Чепок Андрій Олегович, к. ф-м. н., ст. викл. каф. ІПЗ
затверджені наказом закладу вищої освіти від 28.04.2023 р. № 01-07-70.

2. Строк подання студентом роботи 30.05.2023 р.

3. Вихідні дані до роботи:

1) Операційна система Windows 10
2) Мови програмування C#, TypeScript та Elm
3) Середовище програмування Visual Studio та Visual Studio Code
4) Науково-технічні та певні комерційні публікації, пов'язані з тематикою дипломної роботи
5) Протоколи транспорту: HyperText Transport Protocol (HTTP), Advanced Message Queuing Protocol (AMQP), gRPC Remote Procedure Calls (gRPC) та HyperText Transfer Protocol Secure (HTTPS)

4. Зміст розрахунково-пояснювальної записки:

1) Аналіз дійових осіб , що контактують між собою в рамках індустрії
2) Розробка функціональних вимог та дослідження варіантів використання веб-платформи для роботи рослинного ринку
3) Розробка модулів веб-системи
4) Опис функціонування вед-системи
5) Аналіз архітектурного підходу

5. Перелік графічного матеріалу (з зазначенням обов’язкових креслень)
Слайд 1 – Тема диплому
Слайд 2 – Діаграма Use Cases
Слайд 3 – Аналіз використання CQRS
Слайд 4 - Аналіз моделі запитів
Слайд 5 – Аналіз моделі команд
Слайд 6 - Аналіз використання Event Sourcing
Слайд 7 – Подяка присутнім

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
	Вибір теми та отримання завдання від керівника	03-04.01.2023р.	*вик*
	Формування основних складових дипломної роботи. Пошук, збір та опрацювання потрібної інформації.	05-31.01.2023 р.	*вик*
	Складання плану роботи на основі групування та систематизації зібраних матеріалів за тематикою роботи. Консультації з керівником	01-09.02.2023 р.	*вик*
	Складання ТЗ до ПЗ бакалаврської ВКР. Розробка ПЗ бакалаврської ВКР. Консультації з керівником	10-20.02.2023 р.	*вик*
	Написання програмного коду згідно з ТЗ. Отладка ПЗ. Консультації з керівником	21.02-15.04.2023 р.	*вик*
	Написання пояснювальної записки до ВКР (підготовка рукопису / драфт письмової роботи). Консультації з керівником	16-30.04.2023 р.	*вик*
	Оформлення отриманих результатів (підготовка рукопису згідно до ДСТУ-2015 «Документація. Звіти у сфері науки і техніки»). Консультації з керівником	01-10.05.2023 р.	*вик*

	Подання дипломної роботи на рецензію та нормоконтроль	травень 2023 р. (за графіком)	
	Підготовка презентації (докладу) результатів дипломної роботи	травень 2023 р. (за графіком)	

Студент ________ __Я.О. Голота____
(підпис)

Керівник роботи ________ ____А.О. Чепок___
(підпис)

ВІДГУК КЕРІВНИКА

на бакалаврську роботу студента Голоти Я.О.

з теми: «ІТ-рішення та сервіси для ринку живих рослин»

Бакалаврська робота студента Голоти Я.О. присвячена актуальній темі – розробці автоматичного «помічника» для середнього та малого бізнесу, а саме – т . Тематика роботи є перспективною: вона присвячена сучасним тенденціям використання … націлена на розвиток та розповсюдження високих технологій в бізнес-сфері, має довгострокову перспективу та стосується багатьох напрямів виробничої та соціальних сфер суспільства: 1) автоматична система складання договорів купівлі-продажу; 2) автоматична реєстрація та облік реальних торгівельних угод, тощо.

Зміст роботи відповідає обраній темі. Надана випускна робота складається зі вступу, сімох розділів, висновків та додатка. Особливої уваги заслуговує проведений в роботі ґрунтовний аналіз існуючих програмних засобів щодо автоматизації обліку реальних торгівельних угод.

Авторським розділом даної роботи можна вважати розробку студентом Я.О. Голота відповідного спеціалізованого ПЗ у вигляді сукупності програмних модулів щодо *on-fly* обробки певних комерційних даних. У цій частині студент Я.О. Голота провів розробку свого ПЗ згідно до тематичного завдання та провів успішні тестування роботи модулів для демонстрації працездатності свого авторського ПЗ.

Під час виконання даної роботи студент Голота Я.О. проявив ініціативність, креативність, старанність та наполегливість, обізнаність у сучасних ІТ та технологіях програмування, здатність та вміння застосовувати набуті знання для вирішення доволі складних завдань.

Текстова частина бакалаврської роботи викладена послідовно, чітко і технічно грамотно.

Однак дипломна робота містить деякі недоліки. Наприклад, в розділі №3 роботи (3.MODELING OF THE DOMAIN OF INFORMATIONAL SYSTEM) замало уваги автора було надано до викладення математичного підґрунтя щодо генерації «вхідних» даних системи на етапі їх моделювання; в пояснювальній записці бакалаврської роботи зустрічаються граматичні помилки, описки та незначні порушення ДСТУ-2015 в оформленні роботи.

В цілому бакалаврська робота студента Голота Я.О. відповідає вимогам щодо випускних кваліфікаційних робіт бакалаврів і заслуговує оцінки «відмінно», а її автор – Голота Я.О. – заслуговує присвоєння кваліфікації бакалавр з інженерії програмного забезпечення за заявленою спеціальністю 121 Інженерія програмного забезпечення.

Керівник [illegible] Чепок А.О., к. ф.-м. н.

РЕЦЕНЗІЯ

випускної кваліфікаційної роботи здобувача освітнього ступеню «бакалавр»
за освітньо-професійною програмою підготовки Голота Я.О.
на тему: «ІТ-рішення та сервіси для ринку живих рослин»

В XXI сторіччі дуже яскраво проявила себе суспільна потреба максимальної автоматизації діяльності підприємств малого та середнього бізнесу різних напрямів. Дуже важливим є створення автоматичної платформи для бізнесу, де зустрічаються "Клієнт" та "Провайдер послуги", оскільки така е-платформа є тригером підприємництва: від цього напевне можуть виграти всі сторони, кого зачіпає діяльність підприємств – власники, клієнти, суспільство, держава. Тобто, більш значне проникнення ІТ в навколишнє наше життя є прогресивним.

Робота, що рецензується, присвячена розробці програмного забезпечення (ПЗ) для оптимізації менеджменту сервісних підприємств індустрії вирощування живих рослин.

Автором розроблено алгоритм оптимізації інформаційних потоків з потужними елементами евристичного пошуку певних відповідностей у бізнес-БД, що формуються при функціонуванні сервісних підприємств, а також відповідні програмні модулі.

Кваліфікаційна робота виконана відповідно до завдання. Графічні матеріали та пояснювальна записка виконані відповідно до вимог.

До недоліків слід віднести наступне:

1) в пояснювальній записці відсутні переконливі аргументи на користь висловленої авторської позиції щодо переваг авторського ПЗ у порівнянні з існуючими аналогами;

2) в пояснювальній записці при специфікації вимог до ПЗ не були вказані нефункціональні вимоги (вимоги якості до ПЗ), які дуже впливають на вибір шаблонів проектування та інструментів розробки ПЗ;

3) в пояснювальній записці недостатньо відображені ІТ-регулювання ринкових взаємовідносин між Провайдерами послуг та їх «майбутніми» і чинними Клієнтами, через розроблену веб-систему, – це питання відноситься до менеджменту підприємством, і тому потребує доробки;

4) в пояснювальній записці порушена вимога безособового викладання матеріалу.

Вказані недоліки не знижують цінності виконаної роботи. Кваліфікаційна робота відповідає вимогам до кваліфікаційних робіт здобувачів освітнього ступеню «бакалавр» за освітньо-професійною програмою та заслуговує оцінки «відмінно». Студент Голота Я.О. заслуговує присвоєння за заявленою спеціальністю 121 Інженерія програмного забезпечення кваліфікації бакалавр з інженерії програмного забезпечення.

Рецензент к. т. н. , доцент _____________ Вороной С.М.

РЕФЕРАТ

Текстова частина бакалаврської роботи:41 с., 39 рис., 3 табл., 5 додатків, 6 джерел.

АВТОМАТИЗАЦІЯ БІЗНЕС ПРОЦЕСІВ, АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, БАЗУВАННЯ НА ПОДІЯХ, ПОДІЙНО-ОРІЄНТОВАНА АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, СЕГРЕГАЦІЯ КОМАНД І ЗАПИТІВ

Об’єкт дослідження – інформаційний сервіс для рослинного ринку

Мета роботи – реалізація інформаційного сервісу для рослинного ринку

Метод дослідження - емпіричний з використанням комп’ютерних технологій

У бакалаврській роботі аналізовано та реалізовано прототип інформаційного сервісу для рослинного ринку. Використано, проаналізовано і вироблено рішення для поширених проблем із подійно-орієнтовану архітектурою.

ЗМІСТ

ABSTRACT

This bachelor’s thesis contains 41 pages, 39 figures, 3 tables, 5 appendixes, 6 sources.

ANALYSIS OF SOFTWARE REQUIREMENTS, AUTOMATION OF BUSINESS PROCESSES, CQRS, EVENT-DRIVEN SOFTWARE ARCHITECTURE, EVENT SOURCING, SOFTWARE ARCHITECTURE

Area of research – informational service for green market

Goal of the thesis – implementation of the service for green market

Research method – empirical with the usage of software

This thesis analyses and implements a prototype for the informational service for green market. It uses an event-driven architectural approach. The comparison between “database-driven” approach and event-driven approach was produced.

CONTENTS

DEFINITIONS AND TERMS

MVU – Model View Update design pattern.

REST – Representational state transfer and an architectural style for distributed hypermedia systems.

JSON – Javascript object notation.

JWT – JSON web token.

SPA – Single page application.

CQRS – Command Query Responsibility Segregation.

DDD – Domain Driven Design.

INTRODUCTION

Informational Systems are a driving force behind the movement of automatization of business processes, since they allow businesses to streamline and greatly improve the efficiency of delivery of value and as such in increase in income.

The goal of the business is to sell and care for plants. Following from that, following tasks arise:

1. Care for the plants in preparation for their sale.
2. Put plants for sale and organize delivery through the postal service.
3. Provide customers and employees with instructions for plant care.
4. Track the history of orders and payments and present them in a form that would enhance management's decision making.

Following from the goal and tasks of the business the goal of this work is to automate the process of plant selling and care, which includes the following tasks:

1. Analyze business domain and create a logical framework of this application, specified roles of actors, map business aggregates and the use cases that arise in-between them.
2. Describe the business workflows that would solve implement those use cases.
3. Select fitting software components or build fitting ones in cases of their absence.
4. Organize application architecture.
5. Develop the application.

Additionally, due to auditability reasons, less-standard application architecture would be used, so the comparison between it and a more standard approach would be produced. The categories of performance, complexity of the implementation, and talent recruitment would be used for comparison.

1 REQUIREMENTS OF THE INFORMATIONAL SYSTEM

The business process that is being automated by this application has three main roles of actors: consumer, producer and manager. Table 1.1 includes the use cases as well as correspondence of input and output data related to them.

Table 1.1 – System Use Cases

Number	Use Case	Explanation	Input	Output
	Consumer, Producer, Manager			
S1	Access the system.		Login Password	Session
S2	Update your Password	User should only be able to update their own password and no other.	New Password	New Session
	Consumer, Producer			
A1	Search for plants that can be ordered.	Consumers have this task to be able to order plants. Producers have this task for analysis of posted plants.	Plant Families Plant Soils Plant Regions Price Range Plant Name Plant Age	Plants that specify search requirements
A2	Search for instructions for plants.	If some input parameter has not been provided than there should be no filtering performed on that field.	Plant Family Instruction Title Instruction Description	Instructions: Title Description Cover Content
A3	See detailed information for posted plant.	Consumer can do this to be able to perform more informed decision about ordering.	Plant Id	Plant Name Description Price Families Soils Regions Plant Images Age Seller Credentials Caretaker Credentials

Continuation of Table 1.1

Number	Use Case	Explanation	Input	Output
A3	See detailed information for posted plant.	Consumer can do this to be able to perform more informed decision about ordering.	Plant Id	Plant Name Description Price Families Soils Regions Plant Images Age Seller Credentials Caretaker Credentials
	Consumer			
B1	Order plant.		Post Id Delivery Address	Order Id
B2	See previously used addresses on order.	This would speed up delivery process and improve user experience.		Addresses: City Location
B3	Confirm order to be delivered.	This step may be automatically triggered when the postal system notifies package receival	Order Id	
	Producer, Manager			
C1	Find plants that are being prepared for post.		Limit to Cared	Plants: Plant Name Plant Description Is cared flag
C2	Edit plant information.		Plant Id New Plant	New Plant
C3	Create plant.		Name Description Plant Regions Soils Families Pictures Age	Plant Id

Continuation of Table 1.1

Number	Use Case	Explanation	Input	Output
C4	See plant prepared for sale.	Seeing the plant as a client would see it before it is posted would allow producer to create better posts.	Plant Id	Plant Post with no price specified
C5	Post plant for sale.		Plant Id Price	Post Id
C6	Create Instruction.		Family Cover Title Description Content	Instruction Id
C7	Find users.	This allows managers to manage producers and producers to manage	Name Phone Number	Users: Name Phone Number Roles
C8	Invite users.		Login Roles Email Name Phone Number	User created and email with temporary password send.
C9	Update user roles.	Only for roles with lesser priority than current user's.	Login Role	
C10	Remove post.	For producers this is limited to their posts.	Post Id	
C11	Update instruction.		Instruction Id New Instruction	New Instruction
C12	Reject order.		Order Id	
C13	Start Order delivery.		Order Id Tracking Number	Delivery Id

Continuation of Table 1.1

Number	Use Case	Explanation	Input	Output
	Manager			
D1	See popularity for plants based on their family.			Plant Families: Income Stock Number Instructions
D2	See financial info for plant based on their family.		Time Range	Plant Families: Income Sales Number Sold Percent
D3	See the history of changes performed to any item	This is needed for the reasons of transparency and auditing that is legally required from the business.		Changes list: User that performed changes Time Payload

2 INFORMATIONAL SYSTEM ARCHITECTURE

2.1 High-level overview

Requirements of the application that were provided before create a need for architecture that would allow them to be possible. In this case, the three-tier architecture would be used, whose diagram can be seen on fig. 2.1.

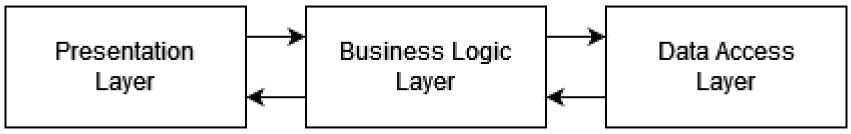


Figure 2.1 – Thee-tier architecture

The main advantage of such architecture over two-tier one is separation of client presentation and the business logic. This allows us to create multiple versions of presentations layer that all use the same business logic component. They may include mobile, web and desktop client applications.

2.2 Business logic layer architecture

Based on the requirements of having the projections of data in form of statistics and aggregations of data from many aggregates, the business logic would be implementing the CQRS principle. The CQRS principle states that the read and the write model of the application should not be one and the same. This allows the business logic to use both highly normalized data sets for write operation and highly de-normalized data sets for read operations. While using this concept the command-query duality arises. Here, the command is any operation that modifies the state of the system, hence using the normalized data set and query is an operation that only requests the state of the system, hence using the de-normalized data sets.

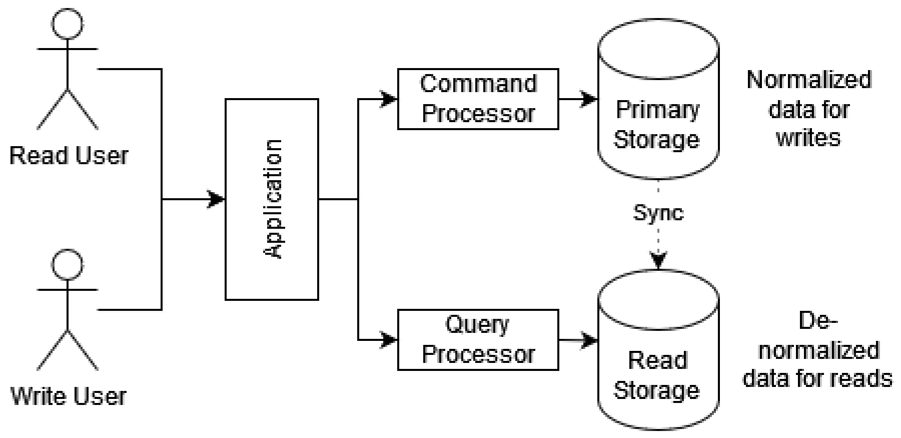


Figure 2.2 – CQRS implementation

Additionally, based on the requirements for traceability and discoverability that were imposed upon the system the approach of Event Sourcing [3] would be taken. The main idea of it consists in considering the events that led to the current state of the system as the source of truth as opposed to a more conventional approach of considering the projection itself as the source of truth. Combining this idea with the CQRS, one may arrive at the architectural depicted on the fig. 2.3

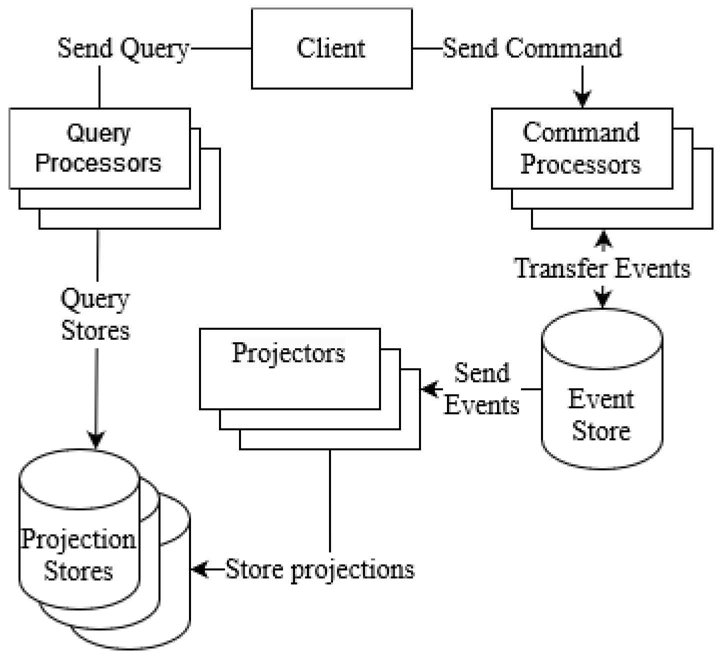


Figure 2.3 – Business layer architecture

Here, the read and write operations are separated by the entry point, which results in us having a separate range of Command Processors and Query Processors. Command Processors are responsible for processing commands with the usage of the events stored with the Event Store. This is important due to concept from Event Sourcing that considers events as the Single Source of Truth. All projections that are needed to perform a command should be handled on-the-fly from the set of events that appeared previously. So, after the command was processed some events may be produced in the response to it. This would be picked up by the projectors, which would project the data into various Projection Stores for future consumption. Due to this type of handling, several projection stores may be used that have their own benefits and drawbacks, so that the storage system is only used in its strong points. As an example of that, one data storage system may be used for its search capabilities and some other data storage system be used for its key lookup capabilities. Correspondingly, the query processor would identify proper projection store to query for the resource requested by the client.

Each of those also needs an internal architectural and compositional structure to use. For that the Clean Architecture [4] would be used as business layer architecture. Its main goal is to separate the actual business logic of the backend application from infrastructural logic. Examples of infrastructure logic include sending emails, querying database and interacting with file system. The diagram of Clean Architecture can be seen on a fig. 2.4.

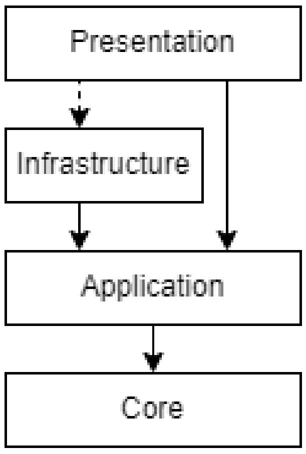


Figure 2.4 – Clean architecture

Here, four structural layers may be seen:

- Core - the base of application and contains system-wide concerns and business entities representation
- Application - business logic

- Infrastructure - external dependencies
- Presentation - a medium for information transfer

Clean architecture has been chosen to allow for separation of business concerns and the actual infrastructure. This is achieved by Presentation layer components providing Infrastructure layer implementations for Application layer dependencies.

2.3 Presentation layer architecture

As a pattern for development of presentation layer MVU [5] pattern would be used. Here, model is an unambiguous and flat representation of all the information that is needed to present the application, view is a function that renders model and convenes user interactions via the messages, and update is a function that uses model and a message and produces new model and optionally commands, side-effects externally processes commands and posts messages.

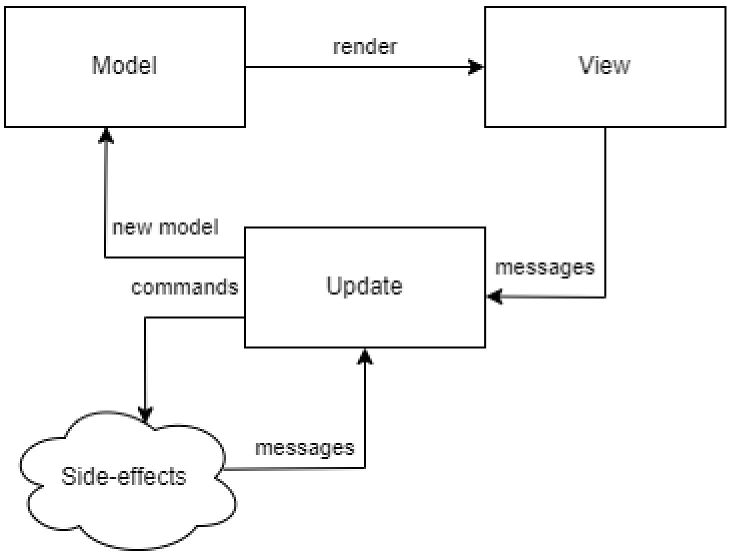


Figure 2.5 – MVU pattern

The MVU has been used to allow for predictable and deterministic user interface design, which would be beneficial for the testing.

Additionally, it is a form of the architectural structure of Event Sourcing that is already in use on the backend, which would decrease the complexity of the system.

3 MODELING OF THE DOMAIN OF INFORMATIONAL SYSTEM

3.1 Modeling data

To model the domain of the informational system the following DDD [2] concepts would be used:

- Entities - identifiable collections of fields that have a schema
- Aggregates - identifiable collections of fields that have a schema, and represent a Unit of Work [6].
- Values objects - are collections of data with some internal consistency logic that are not identifiable.

The full Domain Diagram for read and write models is provided in Appendix B. Otherwise, aggregates, entities and their constraints in tabular form are presented in table 3.1. Both aggregates and entities are identifiable by definition, so they would have an implicit surrogate id in the table.

Table 3.1 – Aggregates and their constraints

Field	Description	Constraints
“User” aggregate		
FirstName	First name of the user	NOT NULL
LastName	Last name of the user	NOT NULL
PhoneNumber	Phone number of the user	NOT NULL
Login	User-friendly identifier of the user	NOT NULL, UNIQUE
Roles	Role that the user has permissions for	NOT NULL, NOT EMPTY
PlantsCared	Number of plants cared for by the user	NOT NULL, NOT NEGATIVE
PlantsSold	Number of plants sold by the user	NOT NULL, NOT NEGATIVE
InstructionsCreated	Number of instructions created by the user	NOT NULL, NOT NEGATIVE
UsedAddresses	Delivery addresses previously used by the user	NOT NULL
“Delivery Address” value object		
City	City of the delivery	NOT NULL
MailNumber	Number of postal location	NOT NULL

Continuation of Table 3.1

Field	Description	Constraints
“PlantStock” aggregate		
CaretakerId	Identitfier of User that is the caretaker	NOT NULL
PlantName	Name of the Plant	NOT NULL
Description	Description of the Plant	NOT NULL
RegionNames	Names of the regions	NOT EMPTY
SoilNames	Names of the soils	NOT EMPTY
FamilyNames	Names of the families	NOT EMPTY
Pictures	Pictures of the plant	NOT NULL
CreatedTime	Time the plant was created in the real world	NOT NULL
“Picture” entity		
Location	Url from which the picture may be downloaded	NOT NULL
“PlantPost” aggregate		
StockId	Identifier of the Stock	NOT NULL
SellerId	Identifier of the User that is the Seller	NOT NULL
Price	Price of the posted item	NOT NULL, NOT NEGATIVE
“PlantOrder” aggregate		
PostId	Identifier of the Post	NOT NULL
BuyerId	Identifier of the User that is the buyer	NOT NULL
DeliveryAddress	Address of the delivery	NOT NULL
OrderTime	Time of order being requested	NOT NULL
DeliveryStartedTime	Time at which the delivery was started	
TrackingNumber	Tracking number for the delivery	
DeliveredTime	Time at which the order was delivered	
“Plant Instruction” aggregate		
FamilyName	Name of the plant family for which the instruction is created	NOT NULL
Text	Content of the instruction	NOT NULL

Continuation of Table 3.1

Field	Description	Constraints
Title	Title of the instruction	NOT NULL
Description	Description of the instruction	NOT NULL
Cover	Cover image	NOT NULL
“PlantsInformation” aggregate		
FamilyNames	Used family names so far	NOT NULL
RegionNames	Used region names so far	NOT NULL
SoilNames	Used soil names so far	NOT NULL
TotalStats	Stats so far	NOT NULL
DailyStats	Day to the stats	NOT NULL
“PlantStats” value object		
FamilyName	Name of family for which stats are collected	NOT NULL
PlantsCount	Number of stock items added	NOT NULL, NOT NEGATIVE
InstructionsCount	Number of instructions created	NOT NULL, NOT NEGATIVE
PostedCount	Number of posts created	NOT NULL, NOT NEGATIVE
SoldCount	Number of orders delivered	NOT NULL, NOT NEGATIVE
Income	Combined income for period	NOT NULL, NOT NEGATIVE

Additional constraints are presented in table 3.2

Table 3.2 – Additional constraints

Aggregate	Constraints
PlantStock	Cannot be updated if was posted. Age of the plant cannot be edited under any condition.
PlantPost	Can only be deleted by any manager or the producer that created it.
PlantOrder	Can only be deleted by a manager or the producer that created the underlying post. Can only be confirmed to be received by the customer that ordered it.

There are two types of relationships between aggregates, entities and value objects:

- One-to-one
- One-to-many

“One-to-one” relationship exists between following items:
- PlantStock and User
- PlantPost and Stock
- PlantPost and User
- PlantOrder and PlantPost
- PlantOrder and DeliveryAddress

“One-to-many” relationship exists between following items:
- User and DeliveryAddress
- PlantStock and Picture
- PlantsInformation and PlantStats

3.2 Modeling Workflows

The workflow describes the existing data and available operation during certain part of the usual user interaction with the system. As such, the workflows combine the use cases and aggregates, and introduce limitations for order of their usage. The first step to modeling the workflows the correspondence between aggregates and the use cases would be created, using the data from table 1.1.

The User contains use cases S1-2, B2, C7-10; the PlantStock contains use cases C1-5; the PlantPost contains use cases A1, A3 and B1; the PlantOrder contains use cases B3, C12-13; the Instruction contains use cases A2, C6 and C11; the PlantsInformation contains use cases D1-2.

Once we have separated out the subdomain, we can map out their interactions, limitations and order of execution. This would be defined in the following figures: fig 3.1, fig 3.2, and fig 3.3.

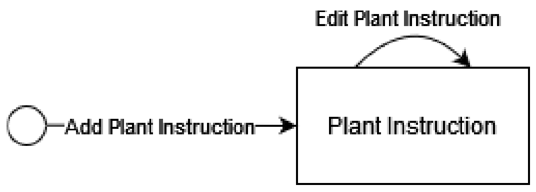


Figure 3.1 – Instruction workflow

Instruction may be added and then edited with no limitations outside of aggregates limitations.

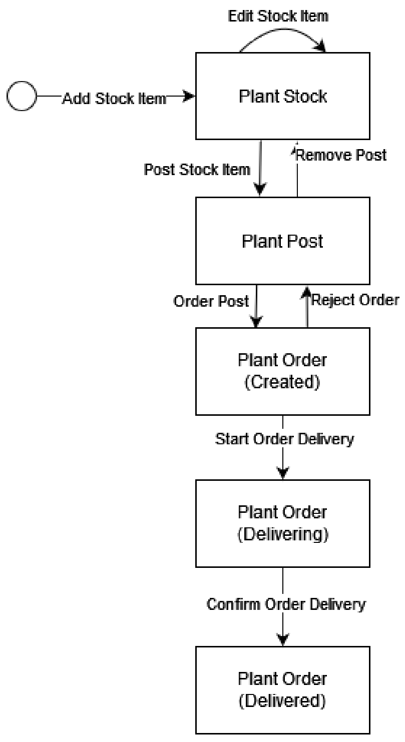


Figure 3.2 – Plant workflow

Plant Stock can be added via Add Stock Item use case, optionally edited via Edit Stock Item use case, and then posted via Post Stock Item use case after which Edit Stock Item Use Case becomes unavailable.

Plant Post may be removed via Remove Post use case by seller, going back to Plant Stock, or ordered via Order Post use case by a buyer.

Plant Order may be rejected, going back to Plant Post, or delivered which happens in two stages – start delivery from seller and configured delivery from buyer, which should appear in that order.

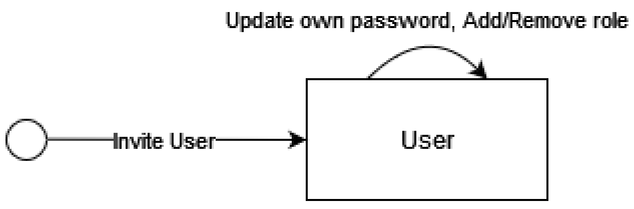


Figure 3.3 – User workflow

User may get invited, change their own password and have a role granted to them or revoked from them by a Producer or Manager.

4 USED TECHNOLOGIES AND SOFTWARE

The informational system is composed of three parts:

- Data Access layer is built with the usage of EventStore for the event storage, MongoDb, and ElasticSearch for Projection Stores.
- Application Layer is built with ASP.NET Core framework [1]
- Presentation Layer is built with Elm, React and Bootstrap 5.

The EventStore was chosen as the Event Store for the following reason:

- User-level access control.
- Throughout documentation.
- Actively supported and developed.
- Support for arbitrary data.

Two Projection Store of MongoDb and ElasticSearch were chosen for their performant key lookup and search queries correspondingly and their support for user-level access control.

The ASP.NET Core framework for backend application has been selected for well-crafted database access packages, advanced support for creation of REST-full APIs and Microsoft support.

The frontend uses Bootstrap 5 for cross-platform support, accessibility and consistency of the user interface, Elm for its support of zero exception runtime and the guarantee of impossibility of undefined state of User Interface and React for its support for Single Page Application development. All of those frameworks are used within Node JS environment that uses Parcel bundler as a build tool for its support for minimization of static files. Build application is being distributed using Nginx web-host through nginx alpine docker image for its support for caching of static files.

5 STRUCTURE OF THE APPLICATION

5.1 Backend architecture

The communication between frontend and backend would be organized through the REST-full API. It may be represented by many forms, but the JSON HTTP API approach would be used, whose diagram is presented in fig. 5.1.

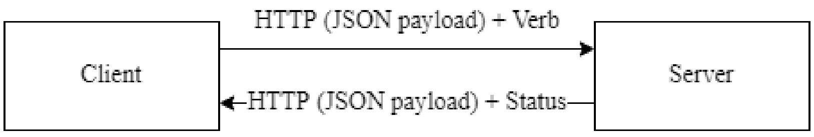


Figure 5.1 - REST API diagram

The authorization would be organized through the usage of JWT, which would use two-way encryption to encode the data so that only the server that has private key is able to read it. Each of the three deployable components would use the Dependency Injection system to implement the abstraction replacement defined in the Clear Architecture. That means that business logic is only aware of abstractions of infrastructural components, such as storage systems, file system, http servers, etc. Such components would be supplied through the infrastructure ports. A diagram of such interaction may be seen on fig. 5.2 and an example of it from the Projection component may be seen on the fig 5.3

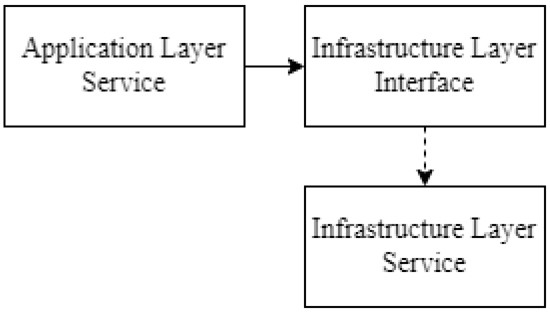

Application Layer Service
Infrastructure Layer Interface
Infrastructure Layer Service

Figure 5.2 – Dependency Injection diagram

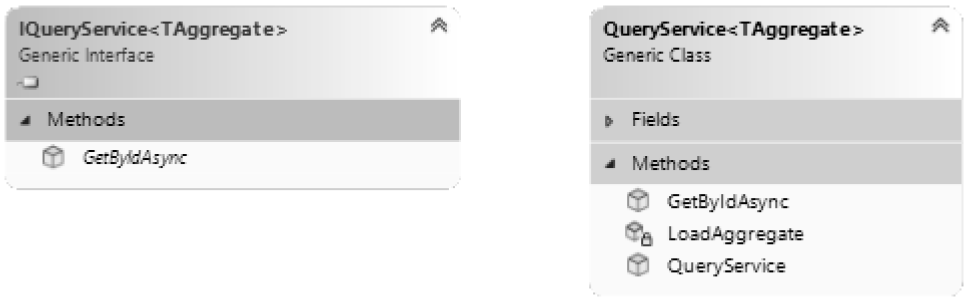


Figure 5.3 – Query service example

The events and commands should be strongly-typed and their schema should be enforced. Abstractions should exist that encapsulates the common parts of events and commands. The same idea should be applied to the aggregates, which should allow the support of high-level event handlers, command handlers, and access controls. The registration of command and event handlers should not require any additional configuration, which might entail them being discovered on the startup. Command handlers should be separated in pure and impure ones. Pure ones should be defined directly on the aggregate, whilst impure ones should be defined externally to allow them to have infrastructural dependencies. However, both command handler types should also be automatically discovered.

The application flow diagrams and source may be found the Appendix D and Appendix E correspondingly.

5.2 Frontend architecture

The frontend application would be structured as one homogeneous application, where all of the users use one and the same application. However, only options that they would be able to execute are visible to them. This should not imply that access validation is limited to the client-side as it should also be enforced on the infrastructural components.

The frontend application should be structured as many MVU applications that represent a singular page of the application that acts as a SPA by each pages having a route within an SPA router.

The frontend application should be able to receive notifications from the backend about the processing of user request being completed. The logic required for this should be located within some shared program module that would be reused for each application page. In addition, this shared module should be able to handle user not being authorized. The structural diagram of such a module may be seen on fig. 5.4.

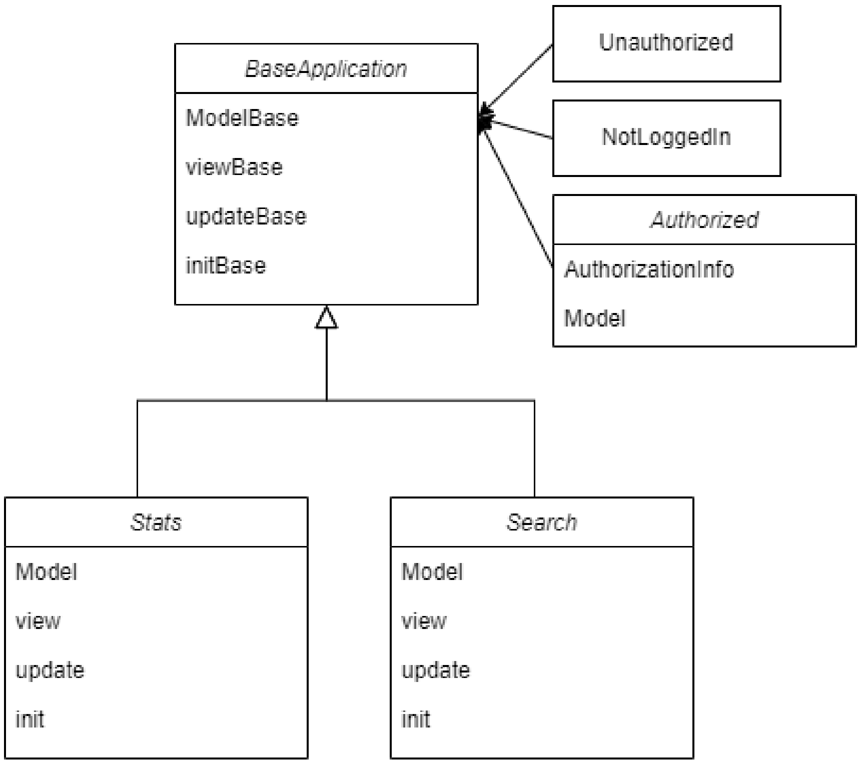


Figure 5.4 – Frontend architecture

The structure of all of the pages may be seen in the Appendix C

6 USER GUIDE WITH ILLUSTRATIONS

This section explores achievement of user tasks through the application user interface.

6.1 Consumer

The initial page of the application is the login page. Its illustration can be seen on fig.6.1. It contains two fields for login and password. There is no way of performing registration, because the system is invite-only. Your credentials should be passed to you through email.

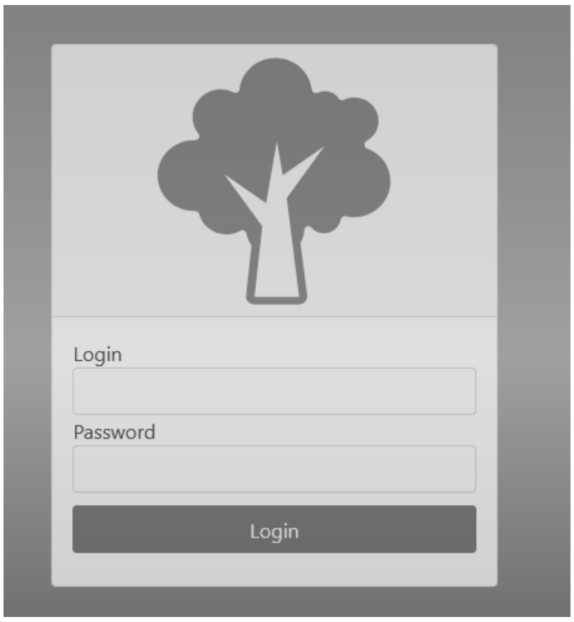
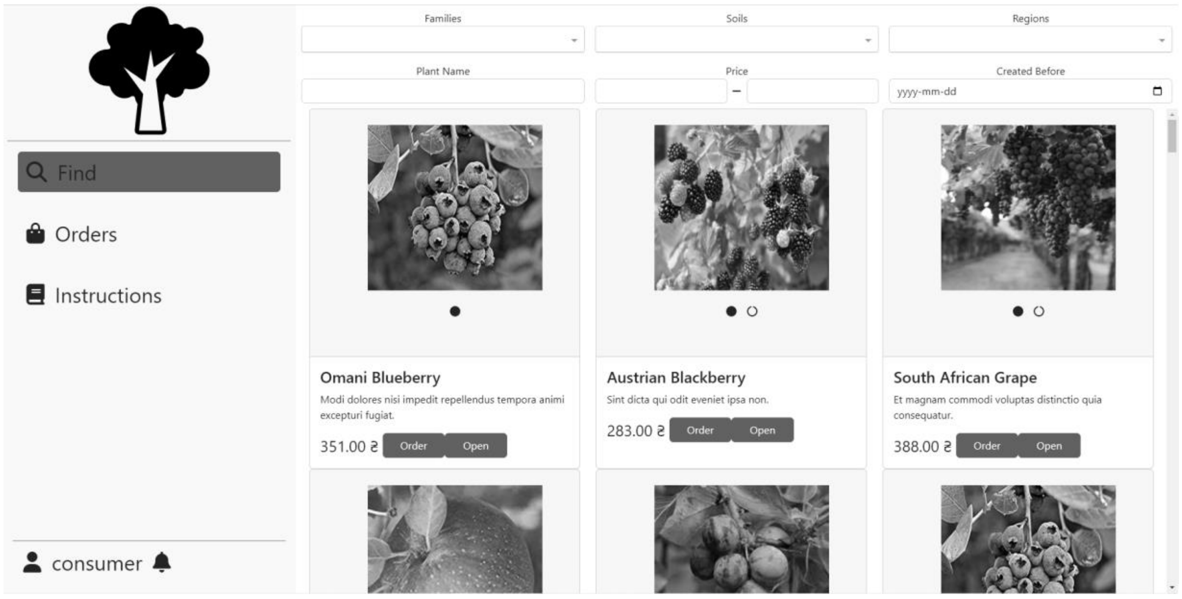


Figure 6.1 – Login Page

The page that you would be forwarded to is the search page. Its illustration can be seen on the fig 6.2. This page contains left-sided navigational bar that is used for the majority of navigation within the application. On the top of the page there are a few inputs for various properties for a plant you are looking for. Upon selecting any of them found list that is displayed below selectors would get updated. From this page you can navigate to order and plant pages by selecting specified buttons of the search result item accordingly.



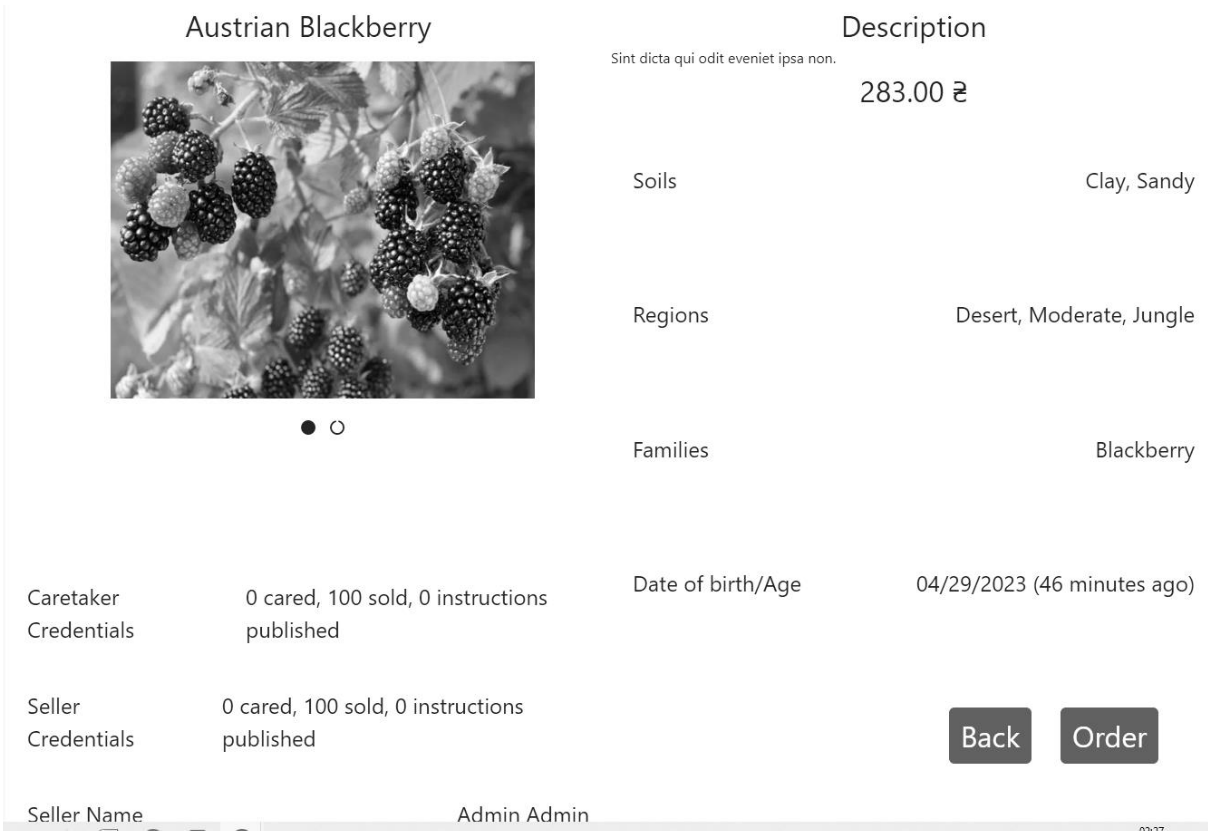
Omani Blueberry
Modi dolores nisi impedit repellendus tempora animi excepturi fugiat.
351.00 ₴ Order Open

Austrian Blackberry
Sint dicta qui odit eveniet ipsa non.
283.00 ₴ Order Open

South African Grape
Et magnam commodi voluptas distinctio quia consequatur.
388.00 ₴ Order Open

Figure 6.2 – Consumer Search page

Page with the detailed plant information can be accessed through search page. Its diagram can be found on fig 6.3. It displays information about plants region, family, age and soil as well as information about its caretaker and seller. From this page you can navigate to ordering page.



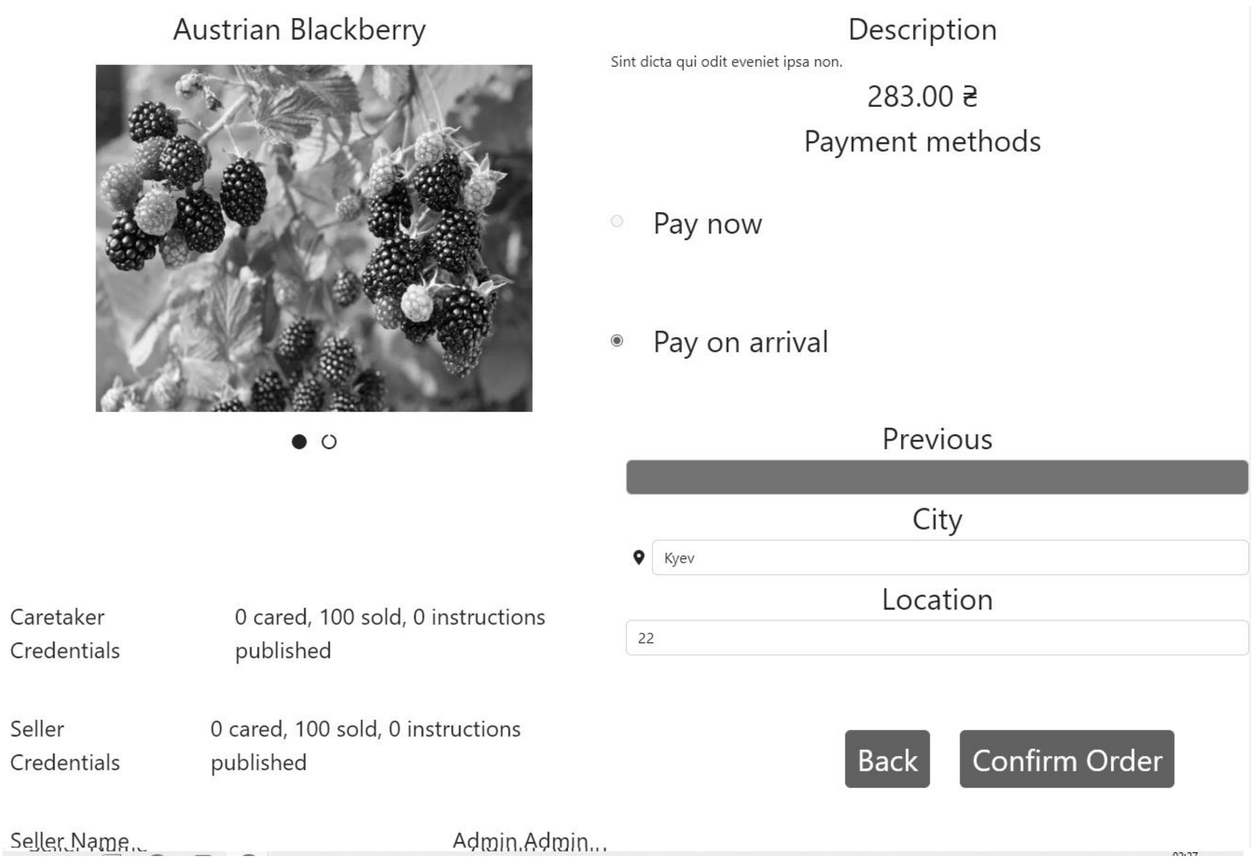
Austrian Blackberry

Description
Sint dicta qui odit eveniet ipsa non.
283.00 ₴

Soils	Clay, Sandy
Regions	Desert, Moderate, Jungle
Families	Blackberry
Date of birth/Age	04/29/2023 (46 minutes ago)
Caretaker Credentials	0 cared, 100 sold, 0 instructions published
Seller Credentials	0 cared, 100 sold, 0 instructions published
Seller Name	Admin Admin

Figure 6.3 – Plant page

Order page displays most important information about plant and allows customer to select payment method as well as delivery address. Its illustration can be found in fig. 6.4. Delivery can be selected out of the list of existing or created on the fly. Upon selecting confirm order an order would be created. The order can found on Orders page that can be accessed through left navigational bar.


Austrian Blackberry

Description

Sint dicta qui odit eveniet ipsa non.

283.00 ₴

Payment methods

Pay now [✓ Pay on arrival]

Previous

City

Kyev

Location

22

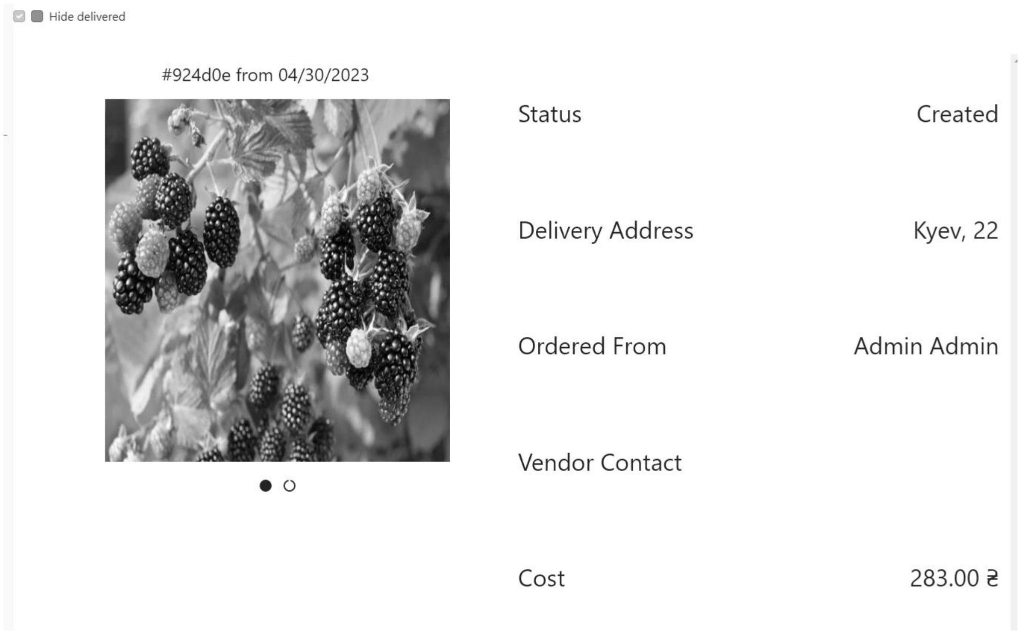
Caretaker Credentials	0 cared, 100 sold, 0 instructions published
Seller Credentials	0 cared, 100 sold, 0 instructions published
Seller Name	Admin Admin

Figure 6.4- Order page

Orders page displays all of the orders that have been made by current customer and allows the customer to confirm the delivery of some order. Its illustration can be seen on fig. 6.5. The status of the plant can have following values:

1) Created – order have not started the delivery
2) Delivering – order have started delivery.
3) Delivered – order have been delivered.

An interaction of confirming delivery can only be performed on delivering status orders. This page allows you to hide delivered orders by checking top-left checkbox.



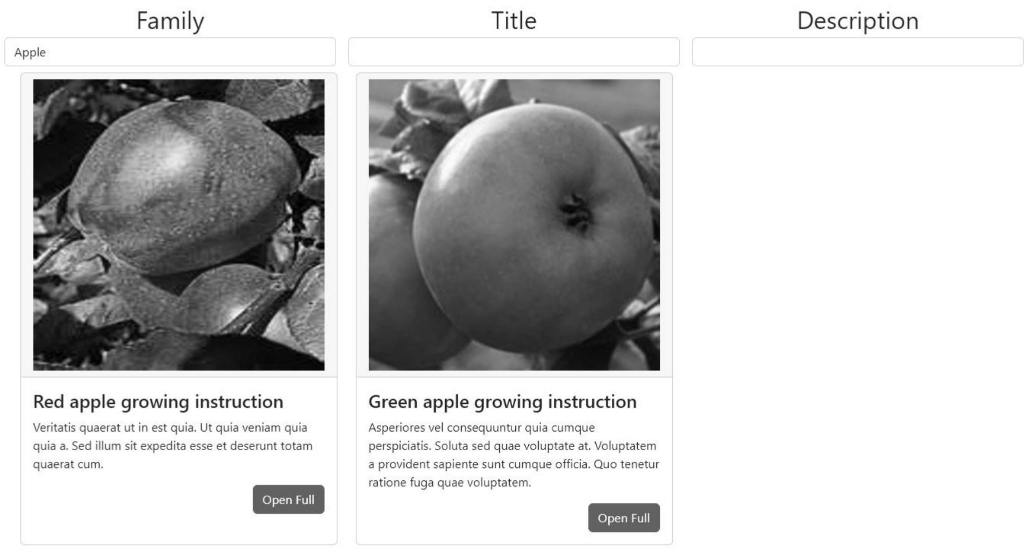
Hide delivered

#924d0e from 04/30/2023

Status	Created
Delivery Address	Kyev, 22
Ordered From	Admin Admin
Vendor Contact	
Cost	283.00 ₴

Figure 6.5 – Consumer Orders page

The instructions page is accessible through the left navigational bar and it displays a search page for instructions that acts the same way as plants search page does. Its illustration can be found on fig. 6.6. This page allows you to change filtering options and then open one for the full view.



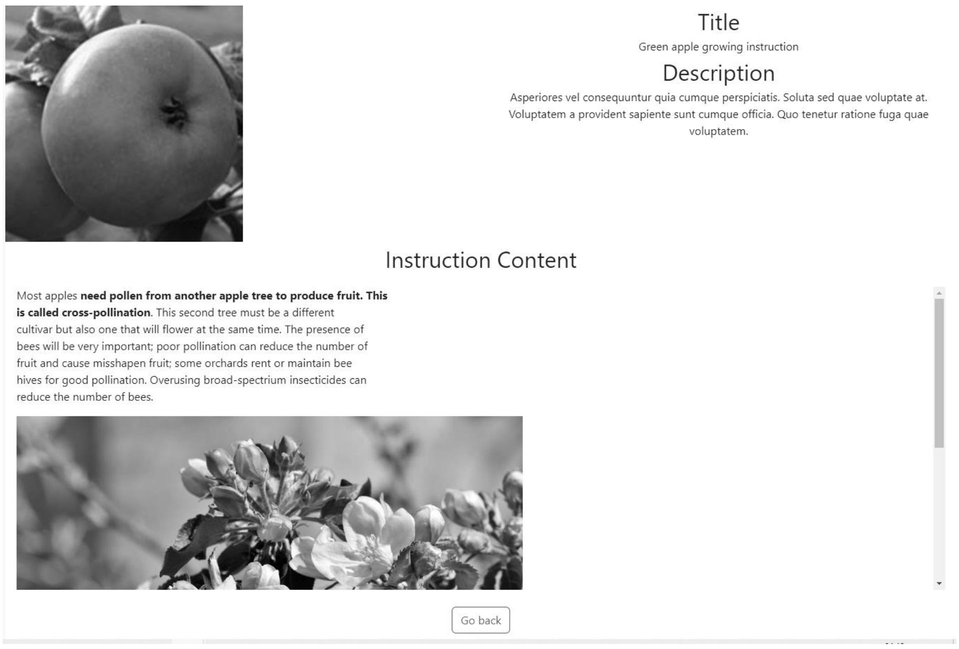
Family: Apple
Title:
Description:

Red apple growing instruction
Veritatis quaerat ut in est quia. Ut quia veniam quia quia a. Sed illum sit expedita esse et deserunt totam quaerat cum.

Green apple growing instruction
Asperiores vel consequuntur quia cumque perspiciatis. Soluta sed quae voluptate at. Voluptatem a provident sapiente sunt cumque officia. Quo tenetur ratione fuga quae voluptatem.

Figure 6.6 – Instructions page

Upon opening instruction for the full view you would see Instruction page that displays all of the relevant information about instruction including its main text that is richly formatted. Its illustration can be seen on fig 6.7. The only interaction is going back to the search page.



Title

Green apple growing instruction

Description

Asperiores vel consequuntur quia cumque perspiciatis. Soluta sed quae voluptate at. Voluptatem a provident sapiente sunt cumque officia. Quo tenetur ratione fuga quae voluptatem.

Instruction Content

Most apples **need pollen from another apple tree to produce fruit. This is called cross-pollination**. This second tree must be a different cultivar but also one that will flower at the same time. The presence of bees will be very important; poor pollination can reduce the number of fruit and cause misshapen fruit; some orchards rent or maintain bee hives for good pollination. Overusing broad-spectrum insecticides can reduce the number of bees.

Figure 6.7 – Instruction page

Profile page can be accessed through left-sided navigational bar and it allows the user to change their password or logout of the system. Its illustration can be seen on fig 6.8.

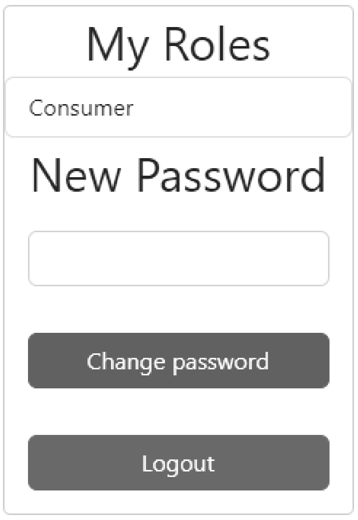
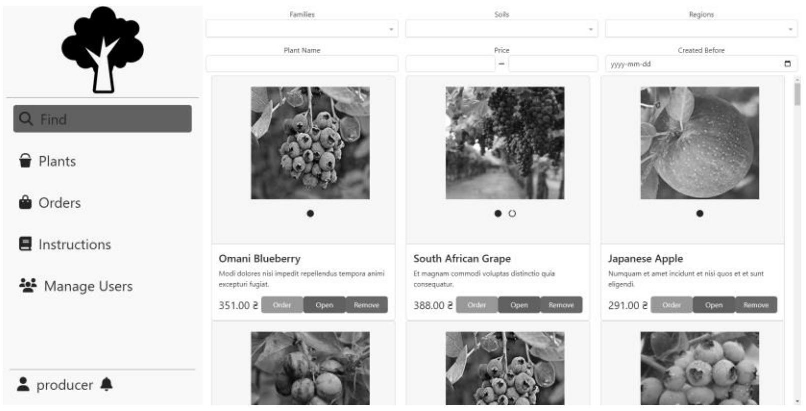


Figure 6.8 – Profile page

6.2 Producer

Producer can access the search page alongside consumer, but the producer would not be able to order the plant. Instead of that producer has interaction to remove the post. This can only be performed for posts that have been created by current producer or by manager. Its illustration can be seen on fig 6.9.



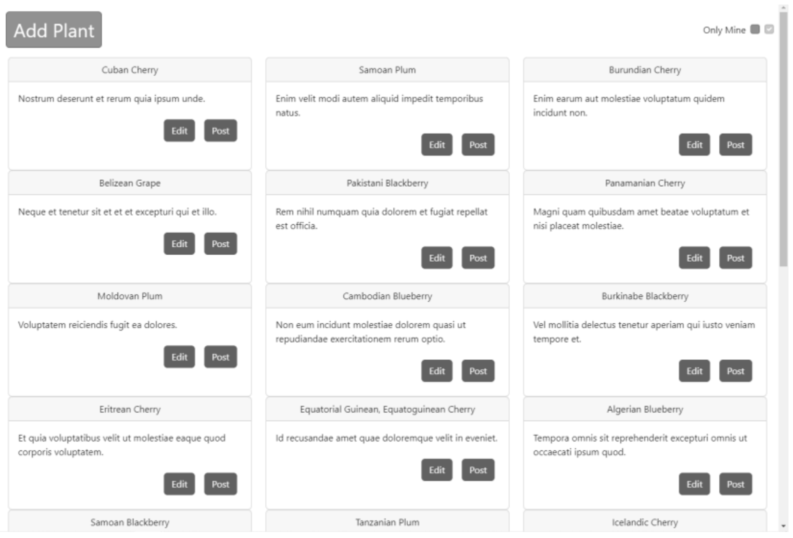
Omani Blueberry
Modi dolores nisi impedit repellendus tempora animi excepturi fugiat.
351.00 ₴

South African Grape
Et magnam commodi voluptas distinctio quia consequatur.
388.00 ₴

Japanese Apple
Numquam et amet incidunt et nisi quos et et sunt eligendi.
291.00 ₴

Figure 6.9 – Producer Search page

Plants page can be accessed through the left-sided navigational bar. It allows the producer to find all of the plants that are being current cared for before they are old enough to be posted for sale. Its illustration can be seen on fig 6.10. It has an option to hide all plants that are being cared for by other producers. It allows producer to add, edit and post a plant that opens corresponding pages.



Cuban Cherry
Nostrum deserunt et rerum quia ipsum unde.

Samoan Plum
Enim velit modi autem aliquid impedit temporibus natus.

Burundian Cherry
Enim earum aut molestiae voluptatum quidem incidunt non.

Belizean Grape
Neque et tenetur sit et et et excepturi qui et illo.

Pakistani Blackberry
Rem nihil numquam quia dolorem et fugiat repellat est officia.

Panamanian Cherry
Magni quam quibusdam amet beatae voluptatum et nisi placeat molestiae.

Moldovan Plum
Voluptatem reiciendis fugit ea dolores.

Cambodian Blueberry
Non eum incidunt molestiae dolorem quasi ut repudiandae exercitationem rerum optio.

Burkinabe Blackberry
Vel mollitia delectus tenetur aperiam qui iusto veniam tempore et.

Eritrean Cherry
Et quia voluptatibus velit ut molestiae eaque quod corporis voluptatem.

Equatorial Guinean, Equatoguinean Cherry
Id recusandae amet quae doloremque velit in eveniet.

Algerian Blueberry
Tempora omnis sit reprehenderit excepturi omnis ut occaecati ipsum quod.

Samoan Blackberry

Tanzanian Plum

Icelandic Cherry

Figure 6.10 – Plants page

Add plant page can be accessed by selecting add plant in plants page. It allows the producer to input all of the information for the plant. Its illustration can be seen on fig 6.11.

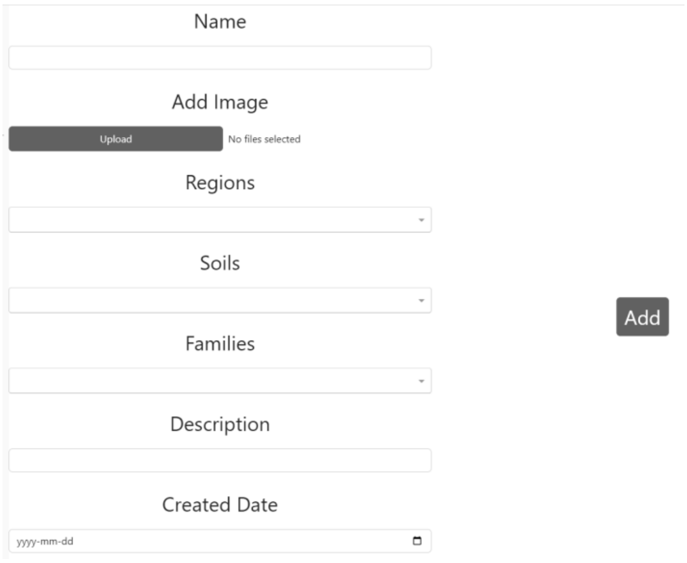
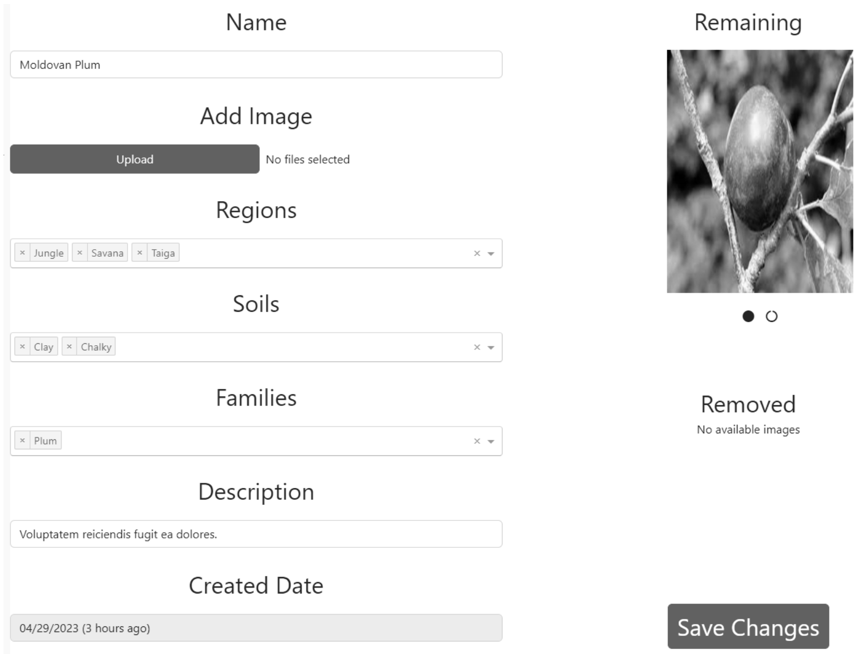


Figure 6.11 – Add plant

Edit plant page is accessible through selecting edit on plant from plants page. Its illustration can be seen on fig 6.12. It allows the producer to change the information about the plant with the limitation of Created Date not being editable. Upon clicking Save Changes the changes would apply.


Name: Moldovan Plum

Regions: Jungle, Savana, Taiga

Soils: Clay, Chalky

Families: Plum

Description: Voluptatem reiciendis fugit ea dolores.

Created Date: 04/29/2023 (3 hours ago)

Figure 6.12 – Edit plant

Add instruction page can be accessed through Instruction page for producers, it allows the producer to create an instruction. Its illustration can be seen on fig 6.13. Upon clicking on edit text a full-screen text editor would be opened. After clicking on Create an instruction would be created.

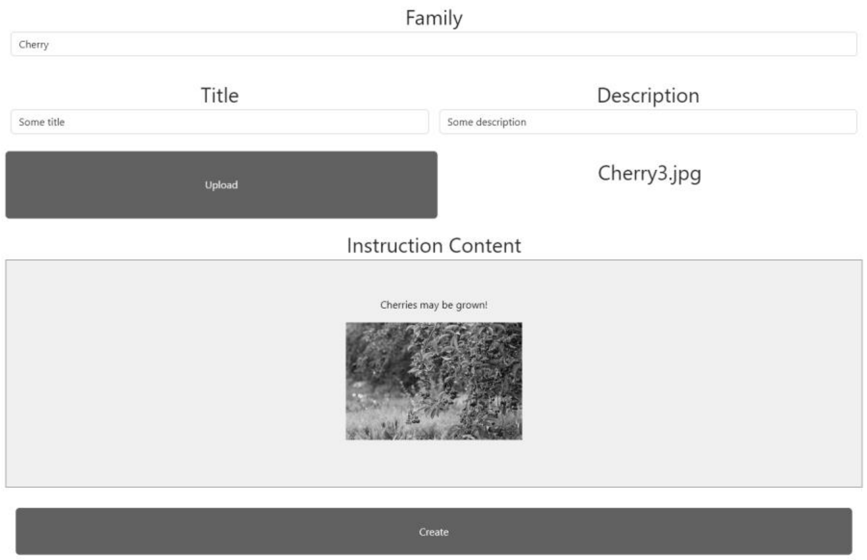
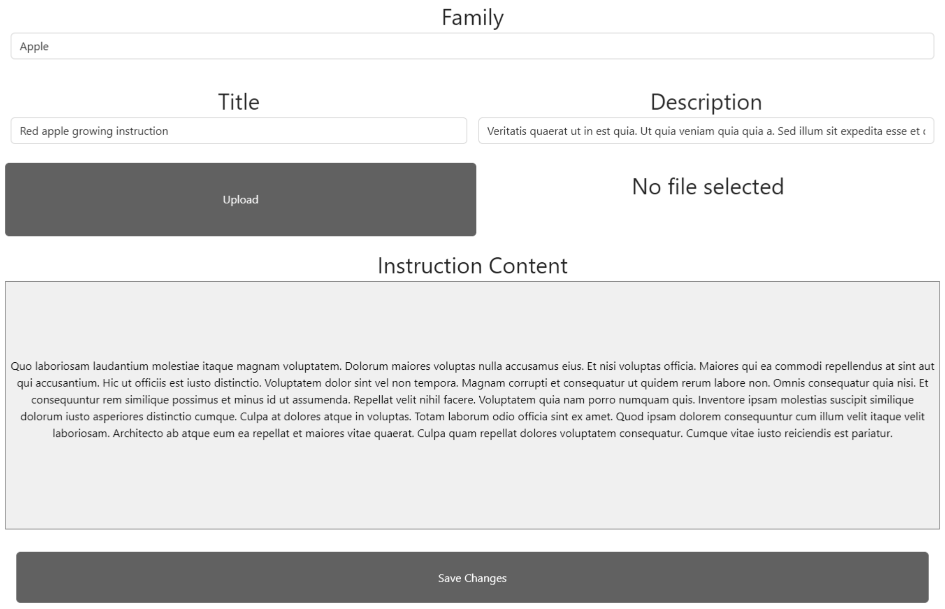


Figure 6.13 – Add instruction

Edit instruction page is accessible through instructions page by clicking on edit on an instruction. Its illustration can be seen on fig 6.14. It allows the producer to change any information about an instruction.

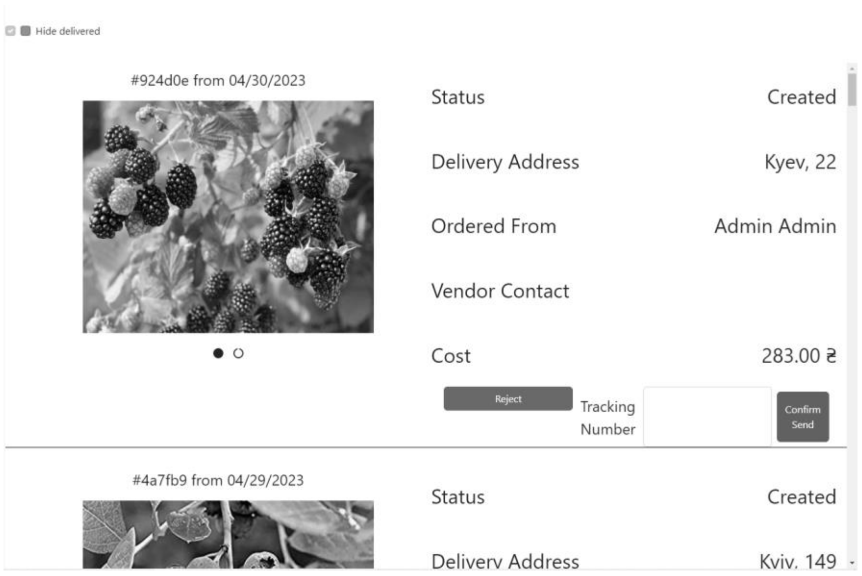


Veritatis quaerat ut in est quia. Ut quia veniam quia quia a. Sed illum sit expedita esse et

Quo laboriosam laudantium molestiae itaque magnam voluptatem. Dolorum maiores voluptas nulla accusamus eius. Et nisi voluptas officia. Maiores qui ea commodi repellendus at sint aut qui accusantium. Hic ut officiis est iusto distinctio. Voluptatem dolor sint vel non tempora. Magnam corrupti et consequatur ut quidem rerum labore non. Omnis consequatur quia nisi. Et consequuntur rem similique possimus et minus id ut assumenda. Repellat velit nihil facere. Voluptatem quia nam porro numquam quis. Inventore ipsam molestias suscipit similique dolorum iusto asperiores distinctio cumque. Culpa at dolores atque in voluptas. Totam laborum odio officia sint ex amet. Quod ipsam dolorem consequuntur cum illum velit itaque velit laboriosam. Architecto ab atque eum ea repellat et maiores vitae quaerat. Culpa quam repellat dolores voluptatem consequatur. Cumque vitae iusto reiciendis est pariatur.

Figure 6.14 – Edit instruction

Orders page is accessible through left navigational bar. Its illustration can be seen on fig 6.15. It displays all of the orders that have been created so far with their statuses being the same as for consumer. However, for producer the interaction is with Created status orders – a producer can decided to reject it or confirm it as being sent by providing a delivery tracking number.



#924d0e from 04/30/2023

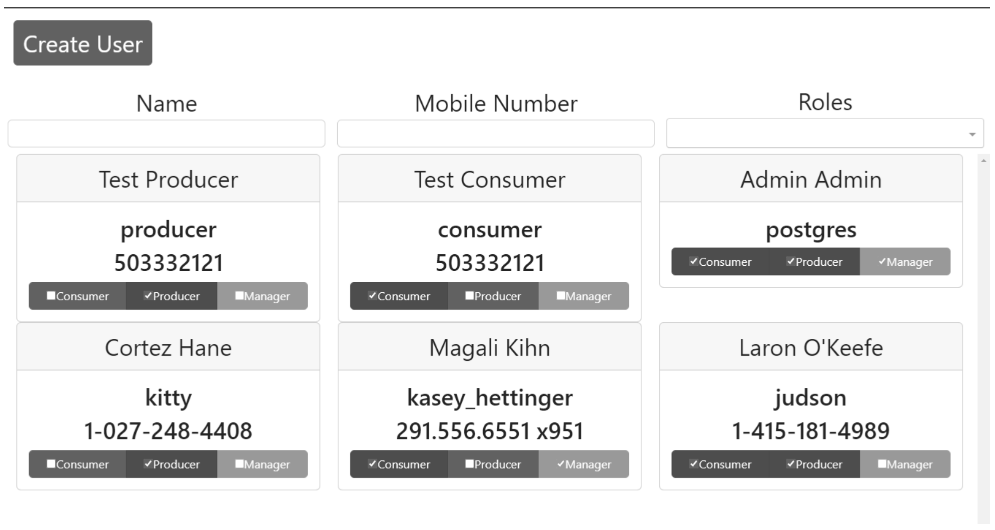
Status	Created
Delivery Address	Kyev, 22
Ordered From	Admin Admin
Vendor Contact	
Cost	283.00 ₴

#4a7fb9 from 04/29/2023

Status	Created
Delivery Address	Kyiv, 149

Figure 6.15 – Producer Orders page

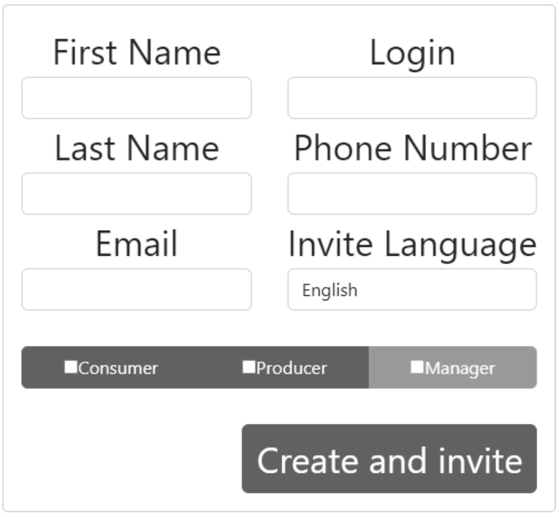
Users page can be accessed through left navigational bar. Its illustration can be seen on fig 6.16. It displays a search by users and it allows a producer to grant producer role to some customer or to revoke customer access as well as an ability to create a user.



Name	Login	Mobile Number	Consumer	Producer	Manager
Test Producer	producer	503332121	■	✓	■
Test Consumer	consumer	503332121	✓	■	■
Admin Admin	postgres		✓	✓	✓
Cortez Hane	kitty	1-027-248-4408	■	✓	■
Magali Kihn	kasey_hettinger	291.556.6551 x951	✓	■	✓
Laron O'Keefe	judson	1-415-181-4989	✓	✓	■

Figure 6.16 – Users page

Add user page displays information required to create a user. Its illustration can be seen on fig 6.17. Upon selecting all of the information and clicking on invite an invite would get send to the selected email.


First Name

Login

Last Name

Phone Number

Email

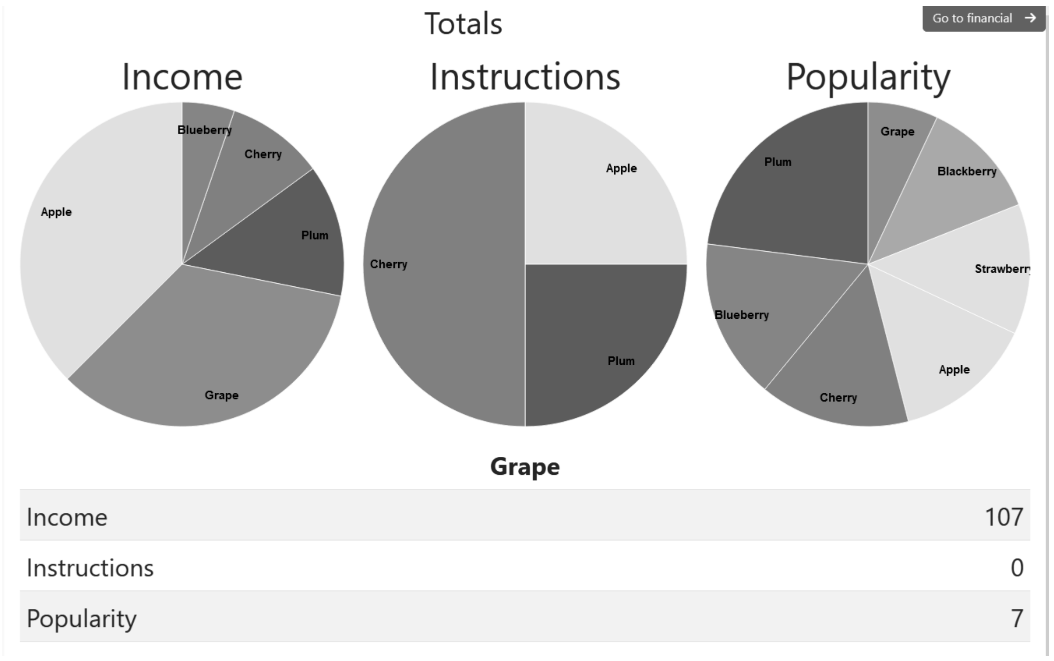
Invite Language

English

Figure 6.17 – Add user page

6.3 Manager

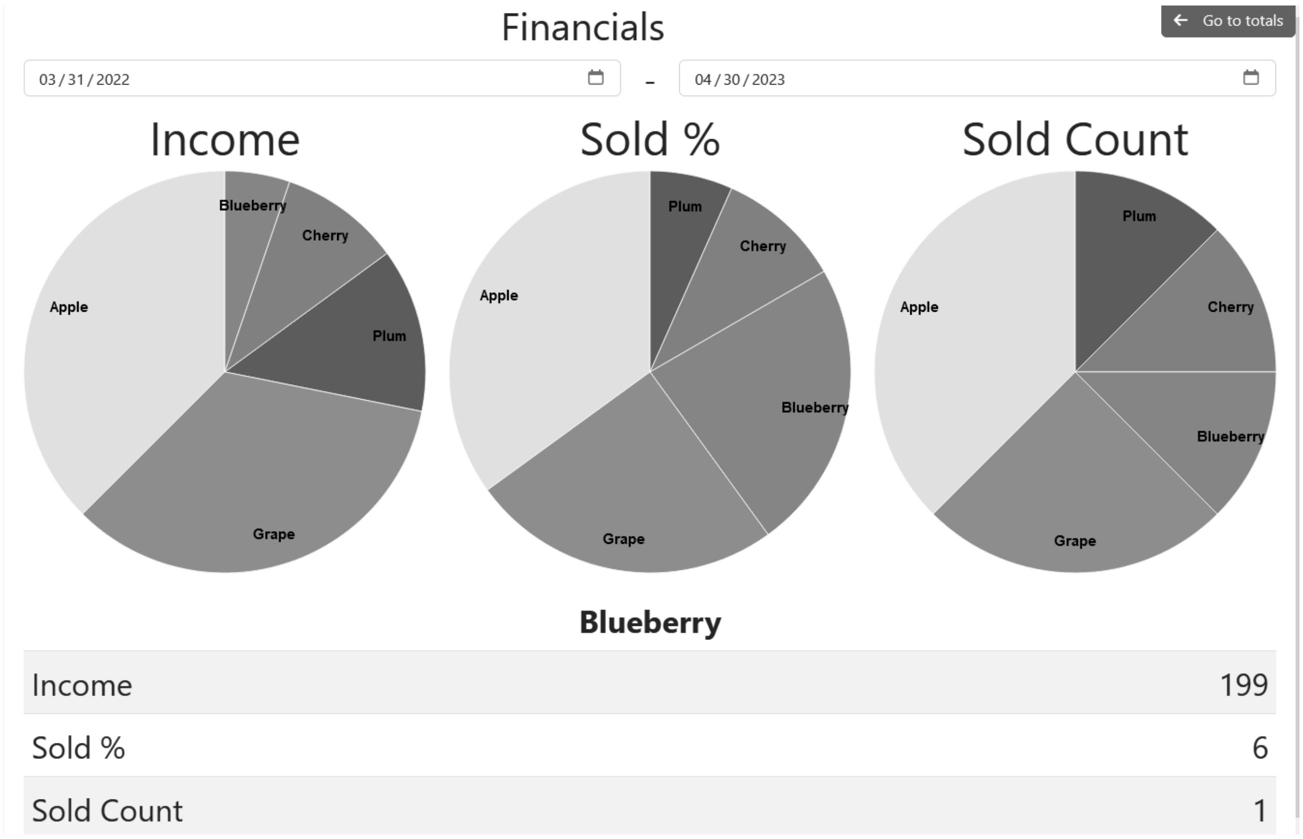
Managers have access to statistics pages that can be accessed through left sided navigational bar. There are two statistics pages: totals statistic page that can be found on fig. 6.18 and financial statistic page that can be found on fig 6.19. Those pages display pie charts for information plant information based on the plant family. Upon selecting a family on pie chart detailed information on it would get displayed in a table below it. Besides that, a manager has access to granting and removing more roles than producer and can remove any post or order.



Grape

Income	107
Instructions	0
Popularity	7

Figure 6.18 – Total statistics page



Income	199
Sold %	6
Sold Count	1

Figure 6.19 – Financial statistics page

In addition to Statistics page, users that are Managers would also see additional “View History” button in many places. Example of such buttons may be seen on fig. 6.20 and 6.21.

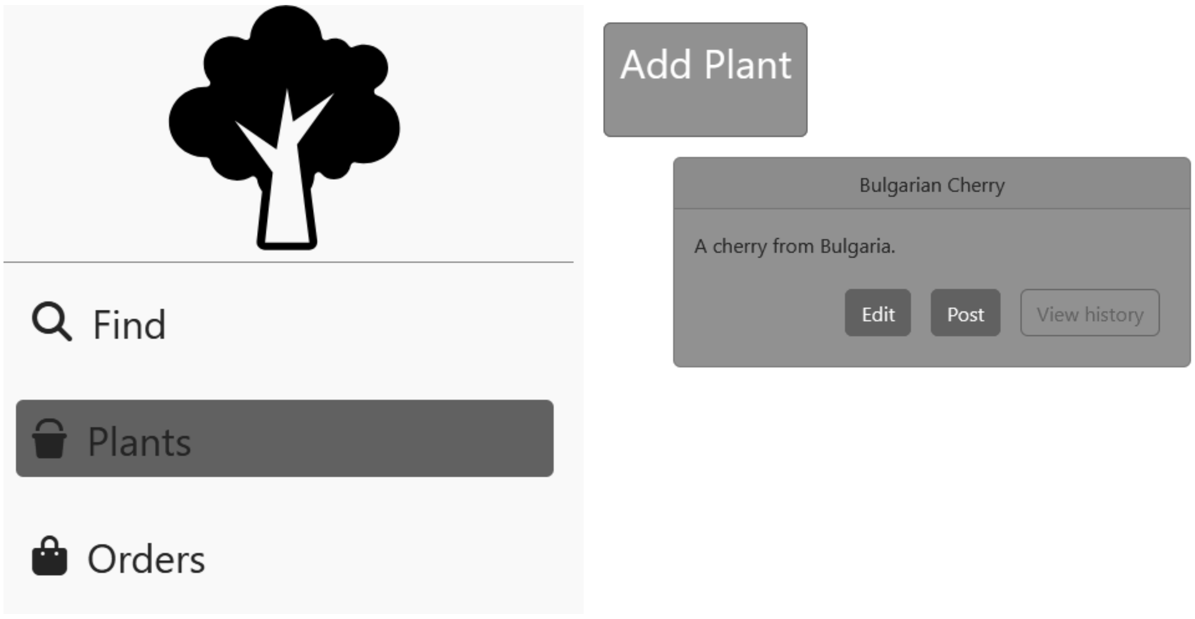


Figure 6.20 - Stock page with history button visible

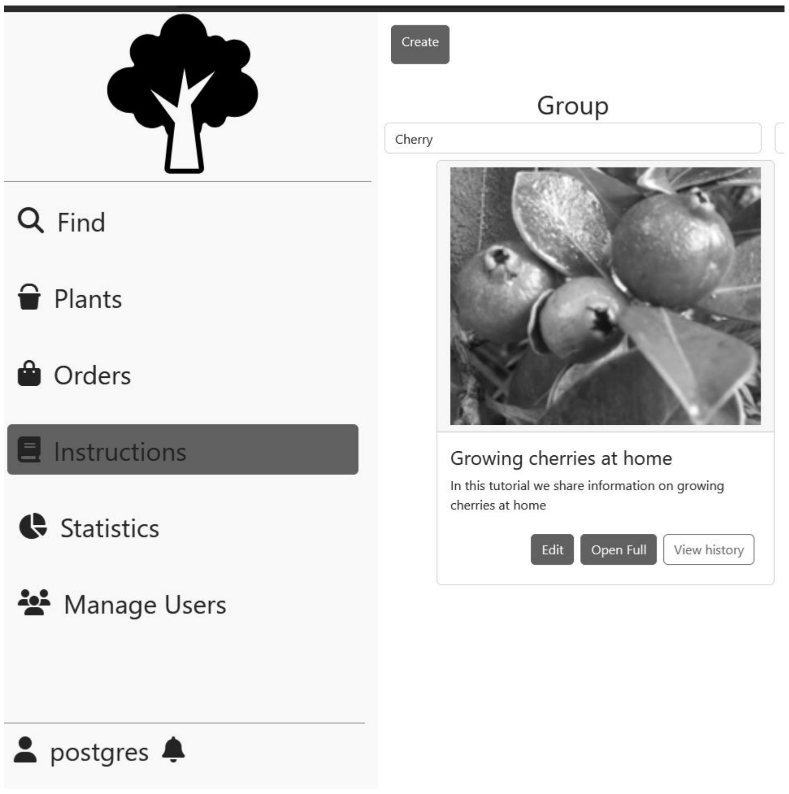
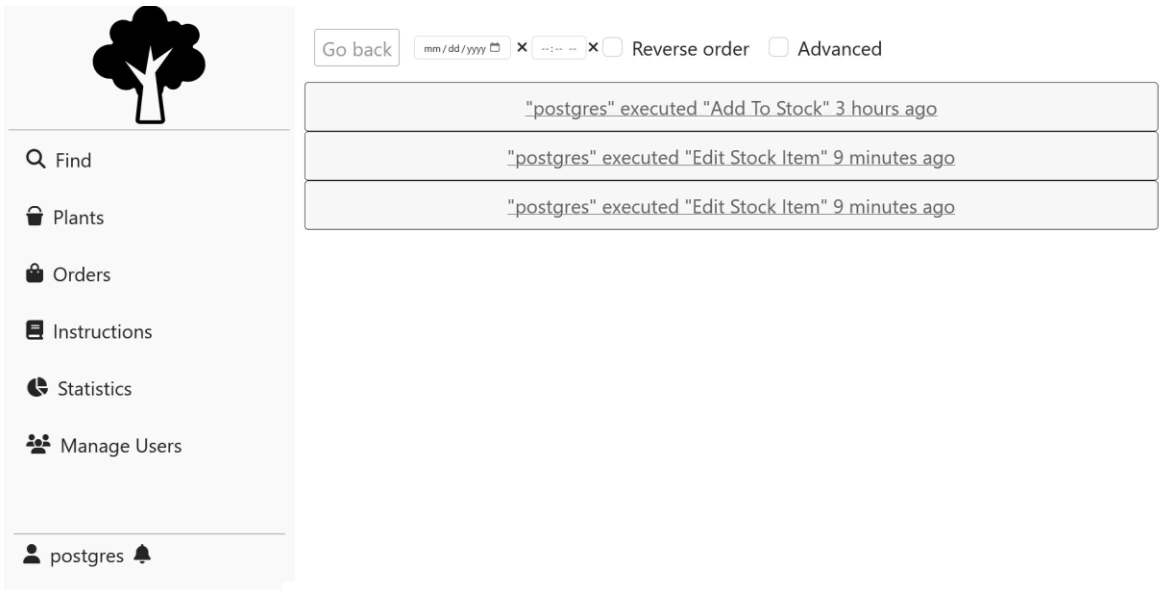


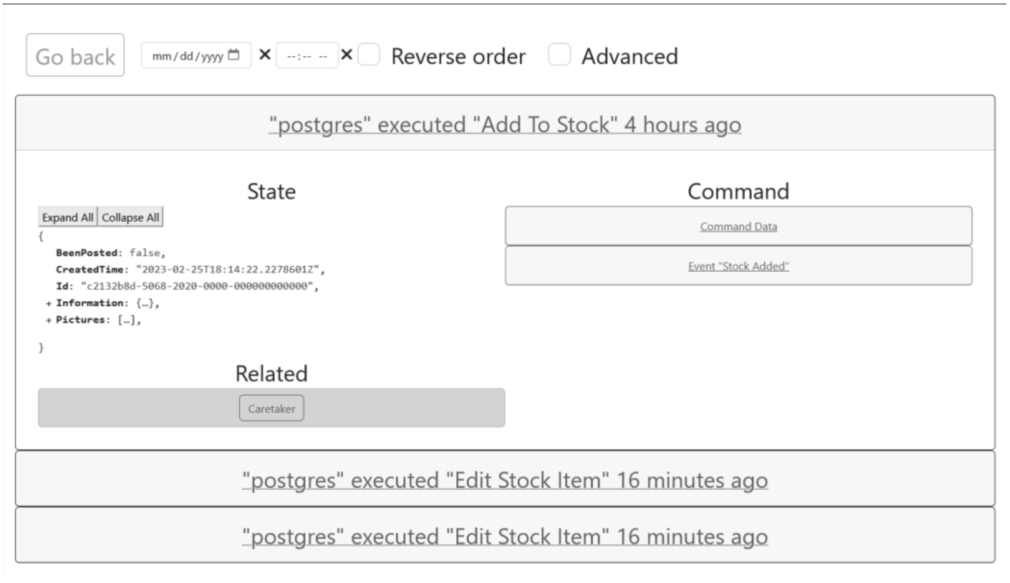
Figure 6.21 – Instructions page with history button visible

Upon clicking on “View History” button, the use would be transported to the history page that may be seen on fig. 6.22. It is showing all of the operations perform with this aggregation in the historical order. The user may reverse the order by checking “Reverse order” checkbox or limit the operations to ones that happened before the specified time. Upon clicking on any of visible commands in would be expanded to the view that may be seen on fig. 6.23.



"postgres" executed "Add To Stock" 3 hours ago
"postgres" executed "Edit Stock Item" 9 minutes ago
"postgres" executed "Edit Stock Item" 9 minutes ago

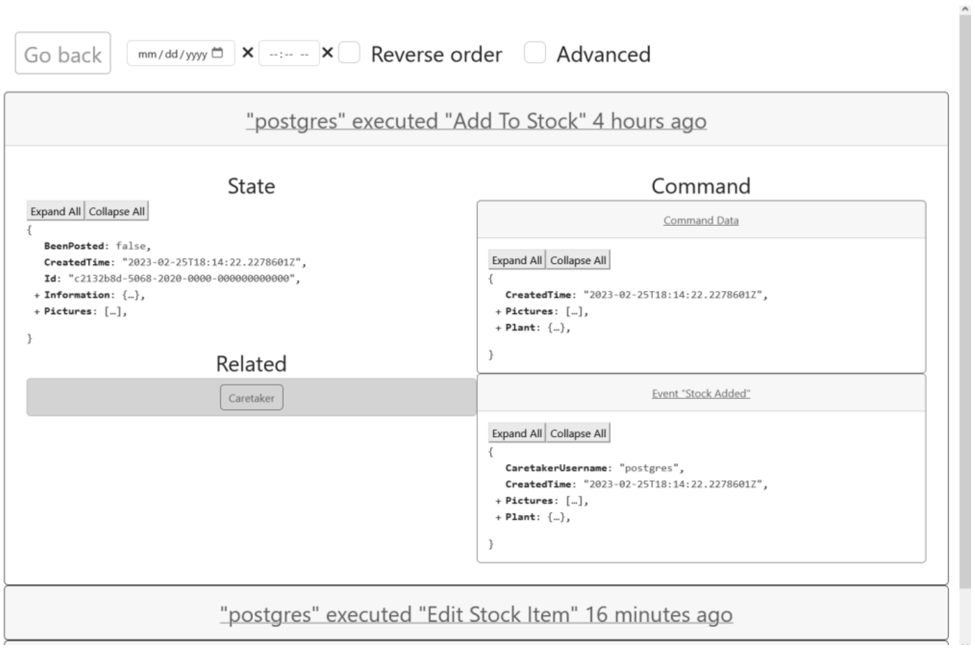
Figure 6.22 – History page


Go back
mm / dd / yyyy
--:-- --
Reverse order
Advanced
"postgres" executed "Add To Stock" 4 hours ago
State
Expand All
Collapse All
Command
Command Data
Event "Stock Added"
Related
Caretaker
"postgres" executed "Edit Stock Item" 16 minutes ago
"postgres" executed "Edit Stock Item" 16 minutes ago

```
{
   BeenPosted: false,
   CreatedTime: "2023-02-25T18:14:22.2278601Z",
   Id: "c2132b8d-5068-2020-0000-000000000000",
 + Information: {…},
 + Pictures: […],

}
```

Figure 6.23 – History page with expanded operation

Once expanded, each operation would contain the state of the aggregate after the operation under the State column, related aggregates list that upon clicking on them would lead to the history of specified user and data related the request that was made by the user labeled as Command Data and results that were produced by the system labeled as Event with some name. Both command and event data may be expanded as is visible on fig. 6.24. That data for state, event and command may be expanded or collapsed by clicking on Expand/Collapse All or clicking on the field of interest.



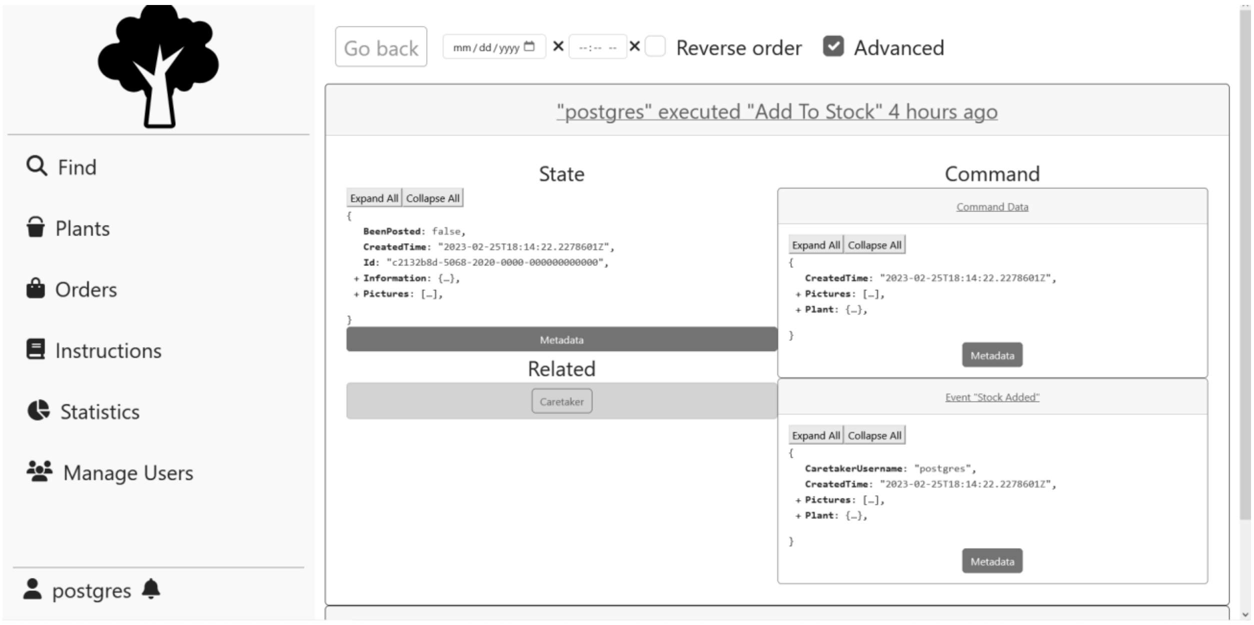
```
{
   BeenPosted: false,
   CreatedTime: "2023-02-25T18:14:22.2278601Z",
   Id: "c2132b8d-5068-2020-0000-000000000000",
 + Information: {…},
 + Pictures: […],

}
```

```
{
   CreatedTime: "2023-02-25T18:14:22.2278601Z",
 + Pictures: […],
 + Plant: {…},

}
```

```
{
   CaretakerUsername: "postgres",
   CreatedTime: "2023-02-25T18:14:22.2278601Z",
 + Pictures: […],
 + Plant: {…},

}
```

Figure 6.24 – Expanded Command and Event views

There is also checkbox that enables advanced mode, which is labeled with “Advanced” tag. Once checked, it would display additional Metadata button for State, Event and Command as is show on the fig. 6.25. Once clicked the button would display some additional information regarding each of those items in an overlaid windows, as may be seen on fig. 6.26.



"postgres" executed "Add To Stock" 4 hours ago

State

```
{
   BeenPosted: false,
   CreatedTime: "2023-02-25T18:14:22.2278601Z",
   Id: "c2132b8d-5068-2020-0000-000000000000",
 + Information: {…},
 + Pictures: […],

}
```

Related

Command

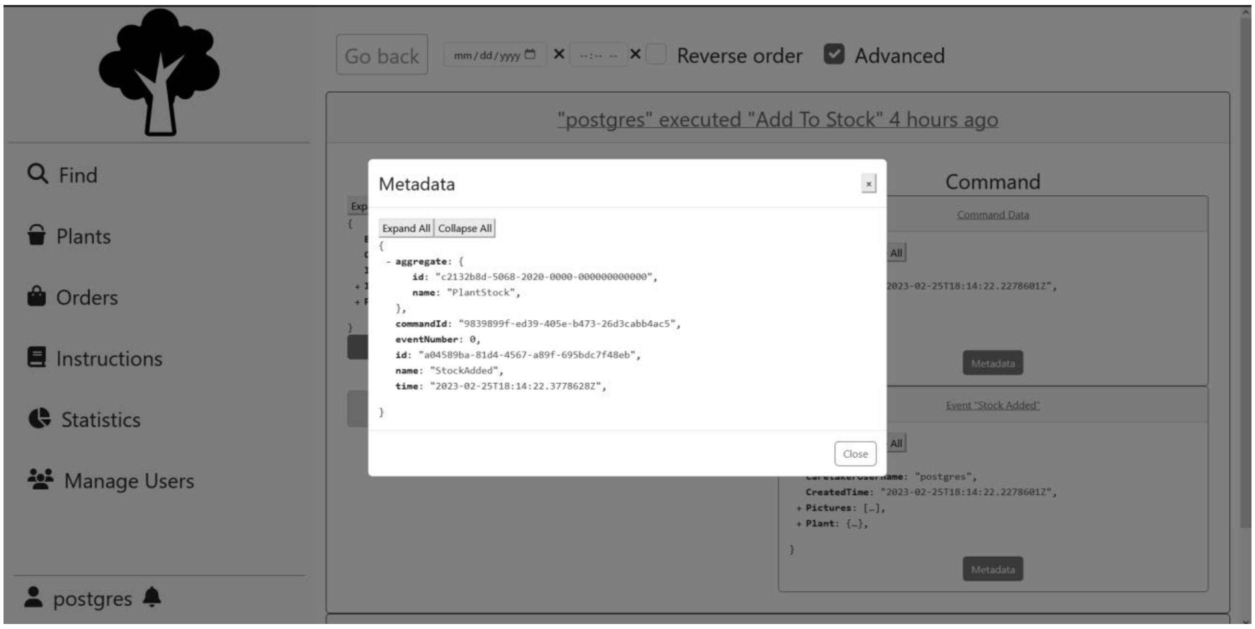
Command Data

```
{
   CreatedTime: "2023-02-25T18:14:22.2278601Z",
 + Pictures: […],
 + Plant: {…},

}
```

Event "Stock Added"

```
{
   CaretakerUsername: "postgres",
   CreatedTime: "2023-02-25T18:14:22.2278601Z",
 + Pictures: […],
 + Plant: {…},

}
```

Figure 6.25 – Advanced mode of history page



"postgres" executed "Add To Stock" 4 hours ago

Metadata

```
{
 - aggregate: {
      id: "c2132b8d-5068-2020-0000-000000000000",
      name: "PlantStock",
   },
   commandId: "9839899f-ed39-405e-b473-26d3cabb4ac5",
   eventNumber: 0,
   id: "a04589ba-81d4-4567-a89f-695bdc7f48eb",
   name: "StockAdded",
   time: "2023-02-25T18:14:22.3778628Z",

}
```

Figure 6.26 – Metadata overlay on the history page

CONCLUSIONS

As the result of implementation of this work, the initial goal of automating the business process was achieved, this included creating software requirements, determining business entities and determining the architecture.

Potential venues of expansion may include:

- Implementation of bulk operations for ordering and posting
- Rating and comment system
- Addition of client-producer messaging service
- Additional of new payment methods

The technological and architectural decisions were able to fulfill business requirements, despite them not being conventional.

Comparing the chosen architecture with the conventional one on the previously discussed categories:

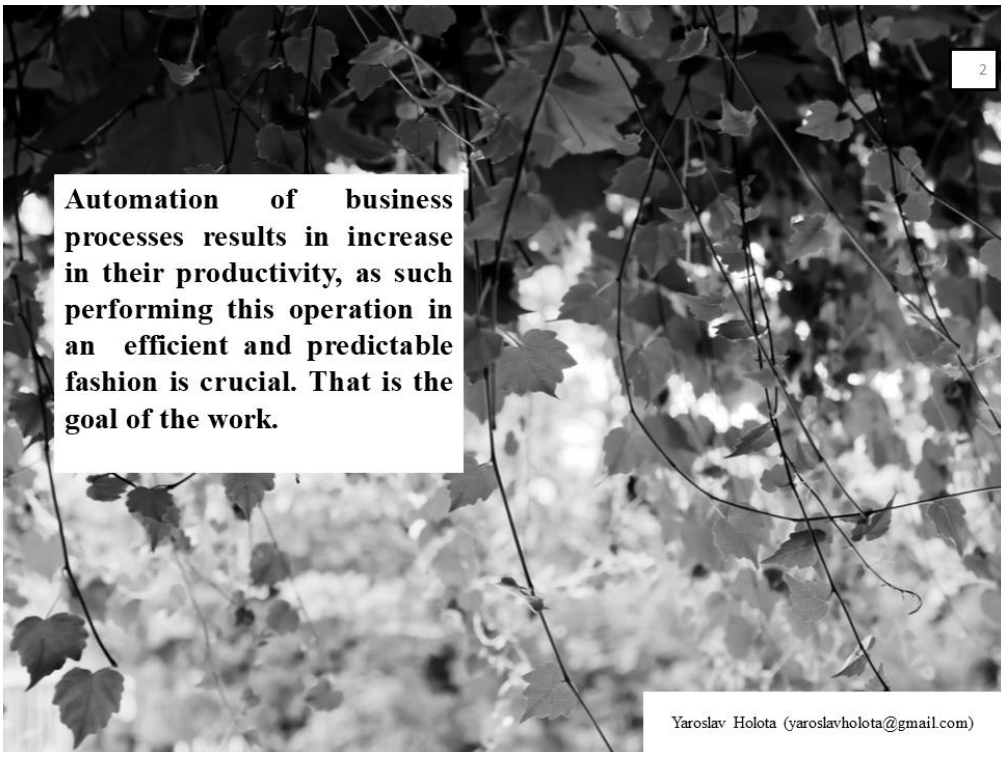
- The category of performance has an inconclusive result, which depends on the context – in single user single deployable instance scenario the conventional approach would have a better response time and would perform less operations overall. However, the event-driven approach has a better scaling ability, which increases multi-processing capabilities of the system.
- The complexity of implementation of the event-driven approach is much higher as it requires much more infrastructure, and there is a larger disconnect between infrastructure and application layers.
- Talent recruitment is a pain-point for the event-driven approach, due to developers already having experience with such technologies.

Overall, the technological and architectural choices were a mixed success.

REFERENCES

1. Lock A. ASP.NET Core in Action – Manning, 2018. – 278 p.
2. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software, 2003 – 560 p.
3. Garofolo E. Practical Microservices: Build Event-Driven Architectures with Event Sourcing and CQRS – 292 p.
4. Common web application architectures: Microsoft Docs - https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures.
5. Elm Architecture Documentation - https://guide.elm-lang.org/architecture.
6. Fowler M., Patterns of Enterprise Application Architecture – 184 p.

APPENDIX A
Presentational Material

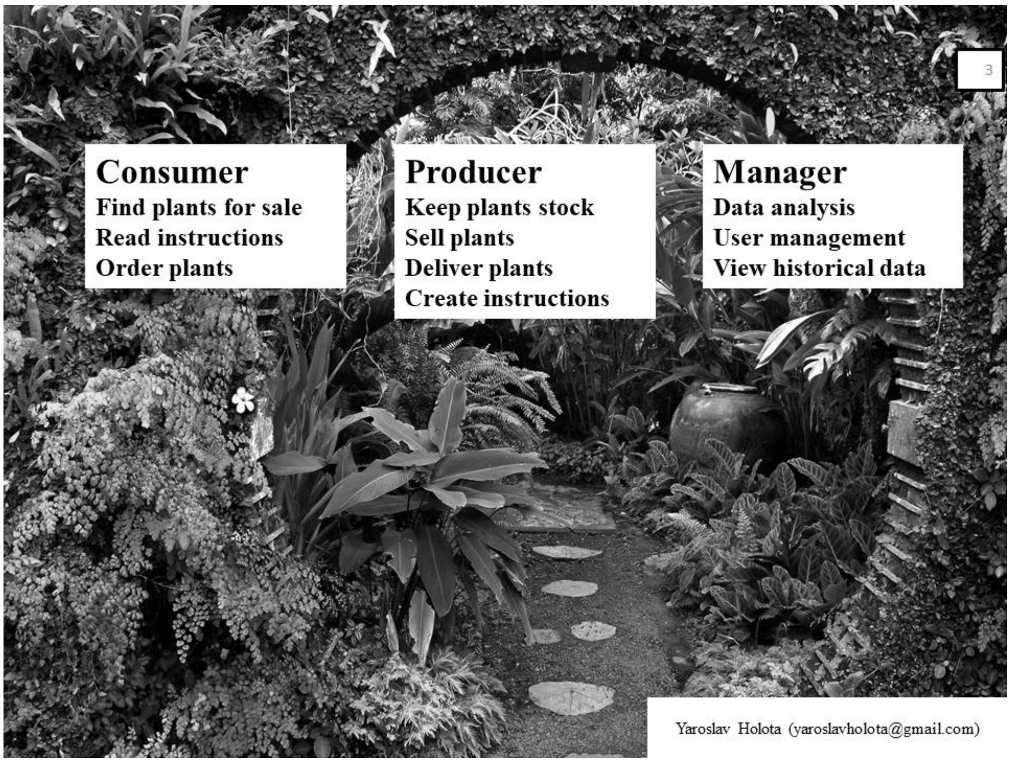


2

Automation of business processes results in increase in their productivity, as such performing this operation in an efficient and predictable fashion is crucial. That is the goal of the work.

Yaroslav Holota (yaroslavholota@gmail.com)

Slide A.1 - Introduction



3

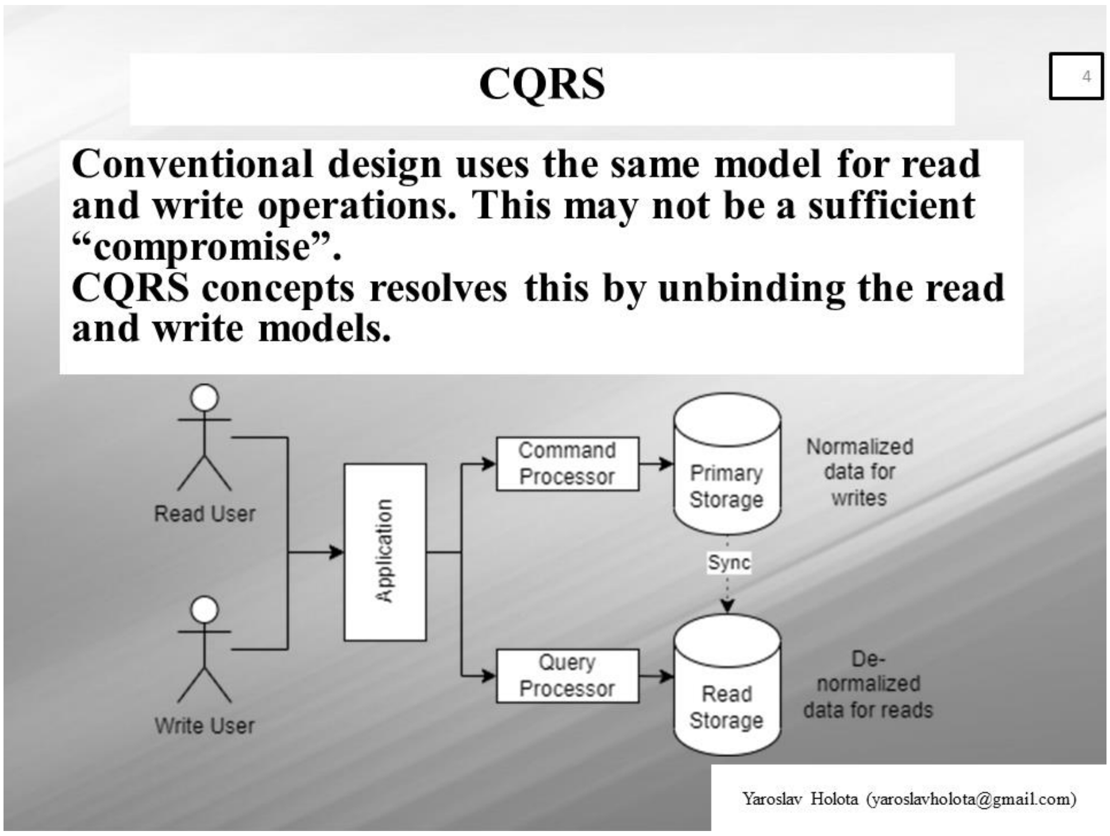
Consumer
Find plants for sale
Read instructions
Order plants

Producer
Keep plants stock
Sell plants
Deliver plants
Create instructions

Manager
Data analysis
User management
View historical data

Yaroslav Holota (yaroslavholota@gmail.com)

Slide A.2 – System Use Cases



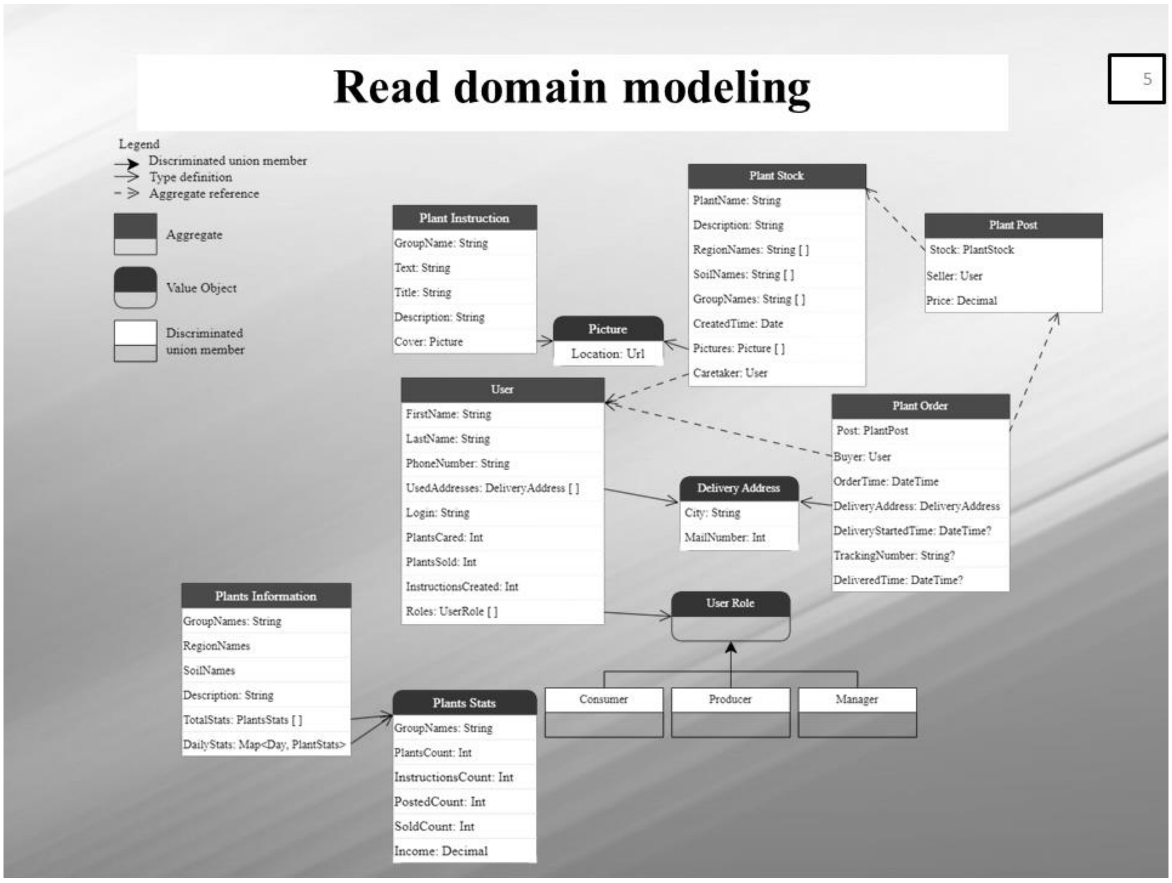
CQRS

4

Conventional design uses the same model for read and write operations. This may not be a sufficient "compromise".
CQRS concepts resolves this by unbinding the read and write models.

Yaroslav Holota (yaroslavholota@gmail.com)

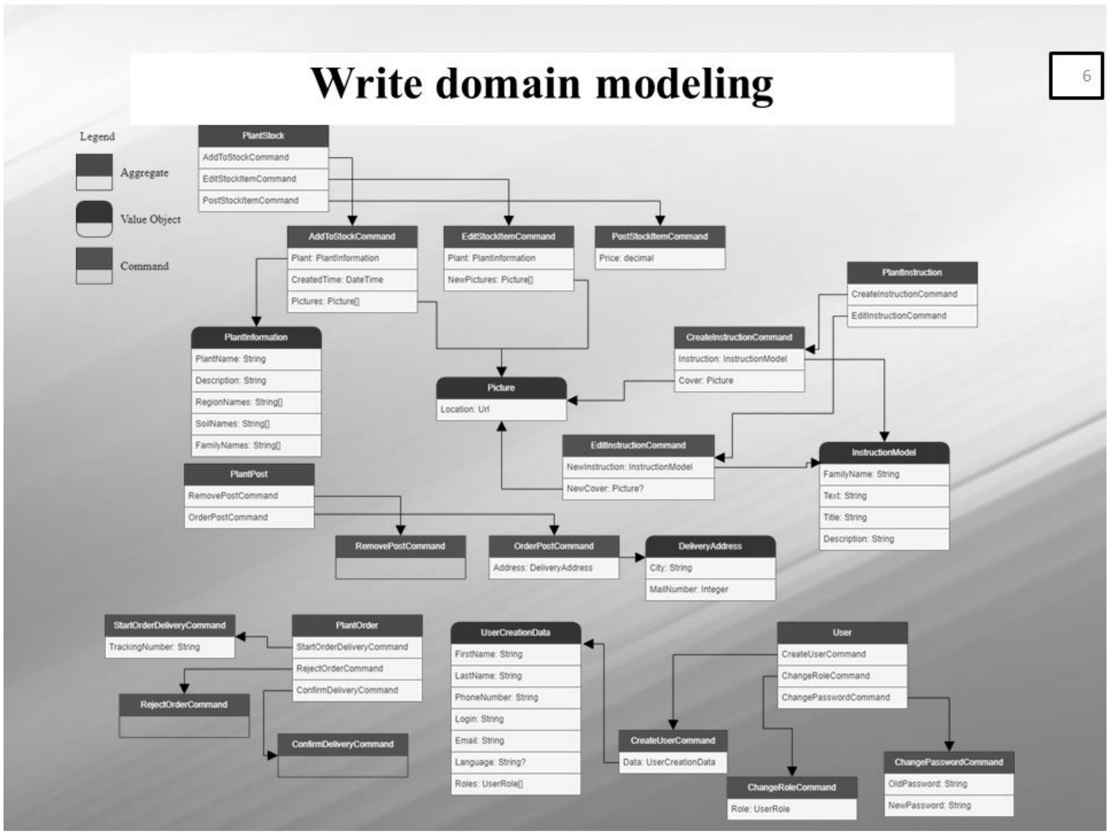
Slide A.3 – CQRS



Read domain modeling

5

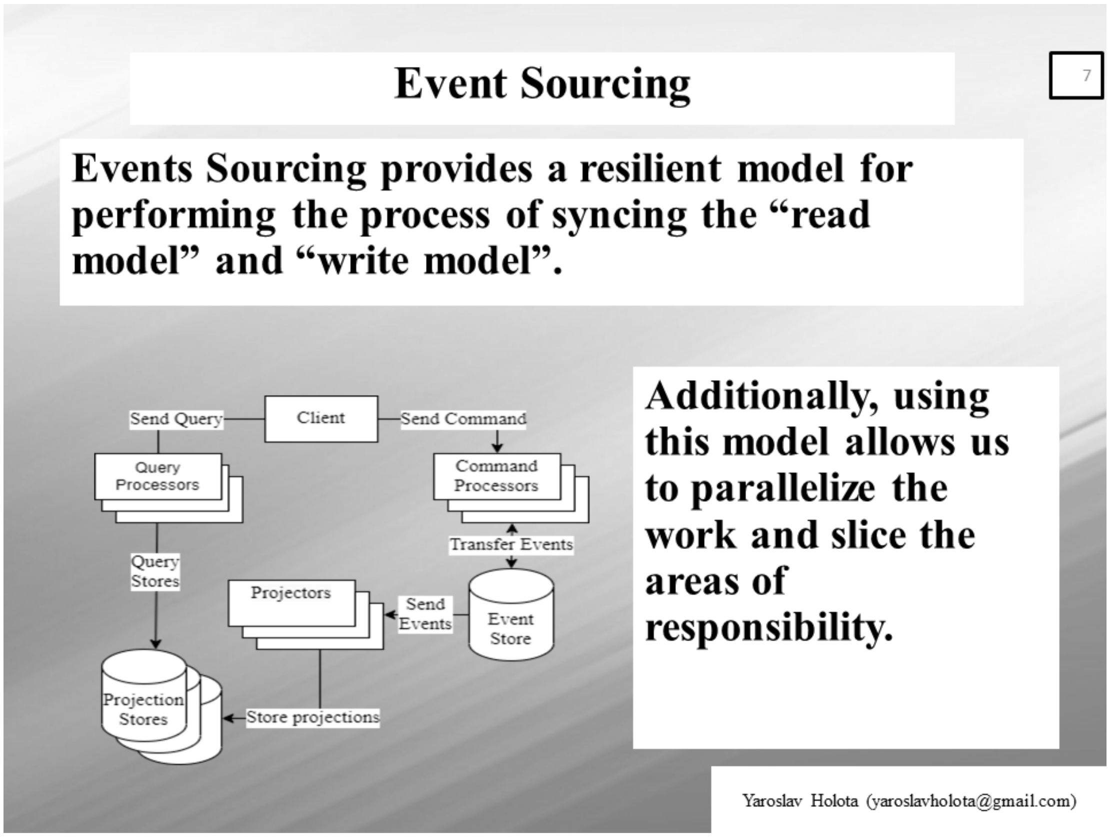
Slide A.4 – Read Model



Write domain modeling

6

Slide A.5 – Write model



Event Sourcing

7

Events Sourcing provides a resilient model for performing the process of syncing the "read model" and "write model".

Additionally, using this model allows us to parallelize the work and slice the areas of responsibility.

Yaroslav Holota (yaroslavholota@gmail.com)

Slide A.6 – Event Sourcing

APPENDIX B
Domain Diagram

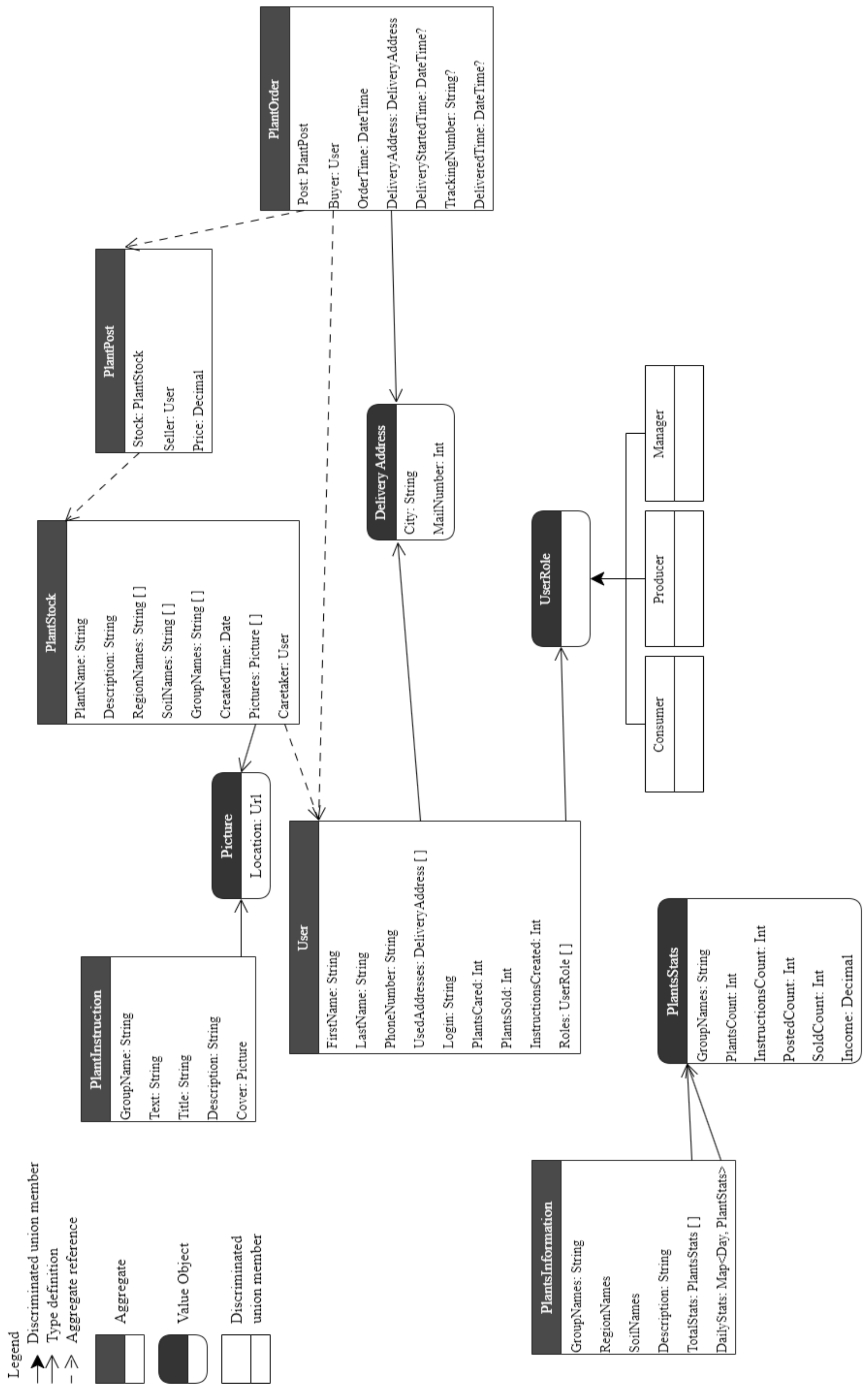


Figure B.1 – Read Domain Diagram

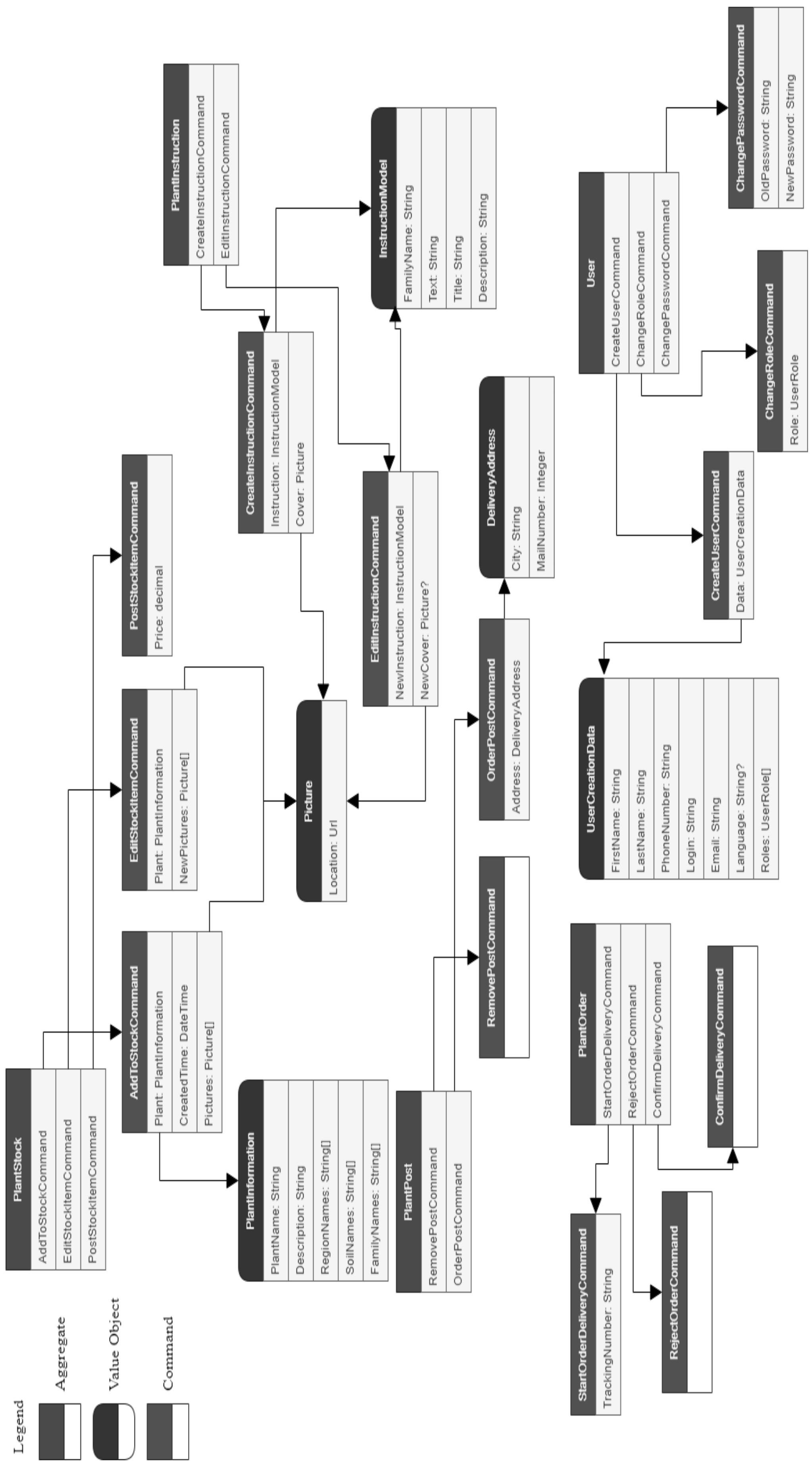


Figure B.2 – Write Domain Diagram

APPENDIX C
Page Navigation

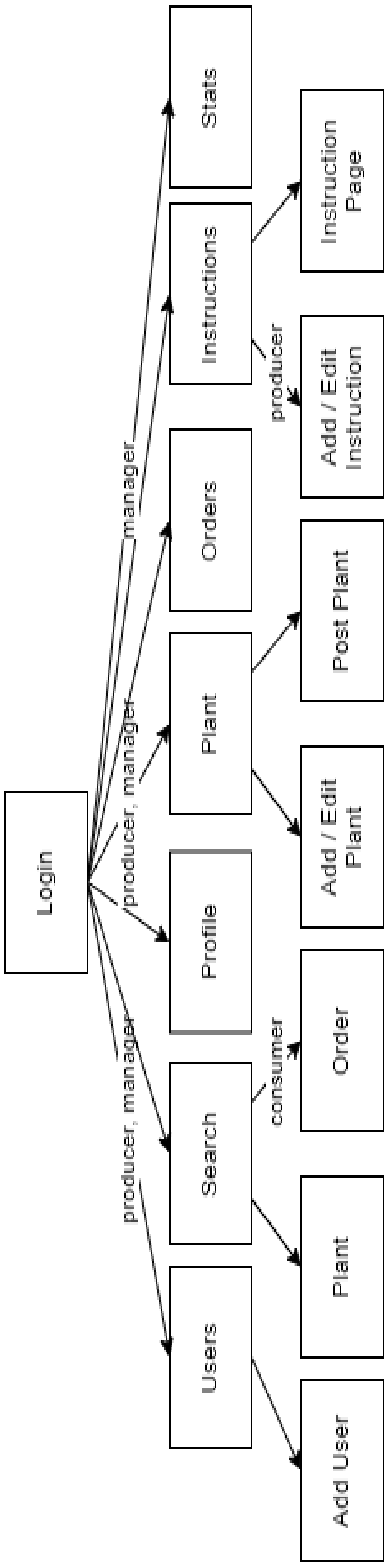


Figure C.1 – Page navigation diagram

APPENDIX D
Application Flow Diagrams

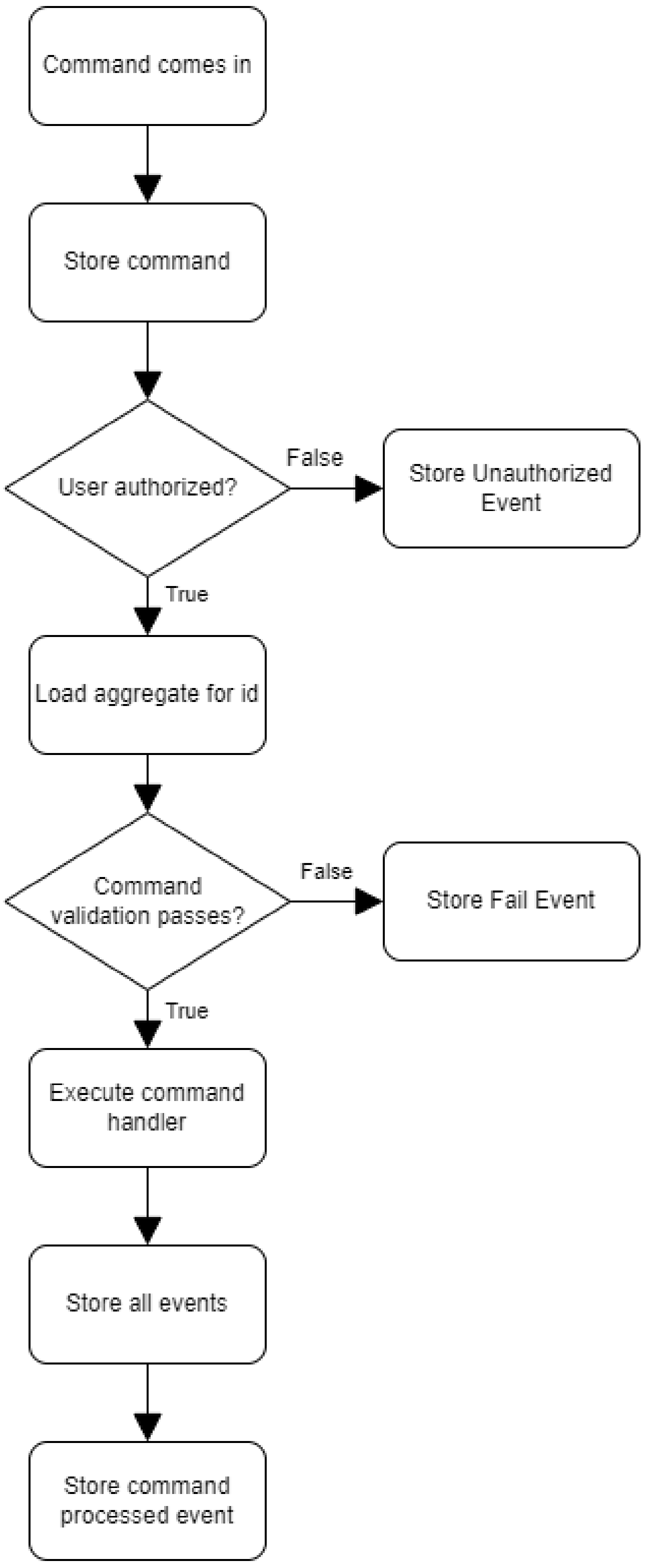


Figure D.1 – Command Sender flow diagram

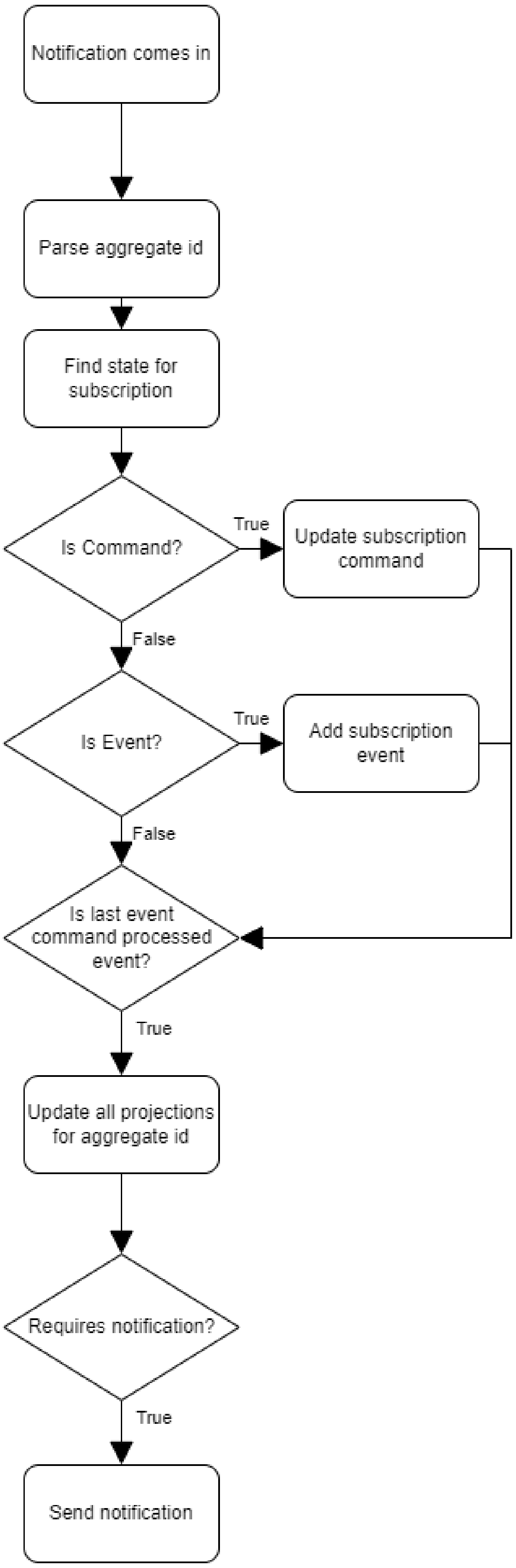


Figure D.2 – Event Subscriber flow diagram

APPENDIX E
Application Source Code

PlantInstructions:

```
namespace Plants.Aggregates;

// Commands

public record CreateInstructionCommand(CommandMetadata Metadata,
InstructionModel Instruction, byte[] CoverImage) :
Command(Metadata);
public record InstructionCreatedEvent(EventMetadata Metadata,
InstructionModel Instruction, string CoverUrl, string
WriterUsername, Guid InstructionId) : Event(Metadata);

public record EditInstructionCommand(CommandMetadata Metadata,
InstructionModel Instruction, byte[] CoverImage) :
Command(Metadata);
public record InstructionEditedEvent(EventMetadata Metadata,
InstructionModel Instruction, string CoverUrl) :
Event(Metadata);

// Queries

public record SearchInstructions(PlantInstructionParams
Parameters, QueryOptions Options) :
IRequest<IEnumerable<FindInstructionsViewResultItem>>;
public record GetInstruction(Guid InstructionId) :
IRequest<GetInstructionViewResultItem?>;

public record FindInstructionsViewResultItem(Guid Id, string
Title, string Description, string CoverUrl);
public record PlantInstructionParams(string FamilyName, string
Title, string Description) : ISearchParams;

// Types

public record GetInstructionViewResultItem(Guid Id, string
Title, string Description,
    string InstructionText, string CoverUrl, string
PlantFamilyName);

public record InstructionModel(
    string FamilyName, string Text, string Title,
    string Description);
```

PlantOrder:

```
namespace Plants.Aggregates;
```

```
// Commands

public record StartOrderDeliveryCommand(CommandMetadata
Metadata, string TrackingNumber) : Command(Metadata);
public record OrderDeliveryStartedEvent(EventMetadata Metadata,
string TrackingNumber) : Event(Metadata);

public record RejectOrderCommand(CommandMetadata Metadata) :
Command(Metadata);
public record RejectedOrderEvent(EventMetadata Metadata) :
Event(Metadata);

public record ConfirmDeliveryCommand(CommandMetadata Metadata) :
Command(Metadata);
public record DeliveryConfirmedEvent(EventMetadata Metadata,
string SellerUsername, string[] FamilyNames, decimal Price) :
Event(Metadata);

// Queries

public record SearchOrders(PlantOrderParams Parameters,
QueryOptions Options) :
IRequest<IEnumerable<OrdersViewResultItem>>;

// Types
public record PlantOrderParams(bool OnlyMine) : ISearchParams;

public record OrdersViewResultItem(
    int Status, Guid PostId, string City,
    long MailNumber, string SellerName, string SellerContact,
    decimal Price, string? DeliveryTrackingNumber, Picture[]
Images,
    DateTime Ordered, DateTime? DeliveryStarted, DateTime?
Shipped)
{
    public string OrderedDate => Ordered.ToShortDateString();
    public string? DeliveryStartedDate =>
DeliveryStarted?.ToShortDateString();
    public string? ShippedDate => Shipped?.ToShortDateString();
}
```

PlantPost:

```
using Humanizer;

namespace Plants.Aggregates;

// Commands

public record RemovePostCommand(CommandMetadata Metadata) :
Command(Metadata);
public record PostRemovedEvent(EventMetadata Metadata) :
Event(Metadata);
```

```
public record OrderPostCommand(CommandMetadata Metadata,
DeliveryAddress Address) : Command(Metadata);
public record PostOrderedEvent(EventMetadata Metadata,
DeliveryAddress Address, string BuyerUsername) :
Event(Metadata);

// Queries

public record SearchPosts(PlantPostParams Parameters,
QueryOptions Options) :
IRequest<IEnumerable<PostSearchViewResultItem>>;
public record GetPost(Guid PostId) :
IRequest<PostViewResultItem?>;

// Types

public record DeliveryAddress(string City, long MailNumber);

public record PostViewResultItem(Guid Id, string PlantName,
string Description, decimal Price,
    string[] SoilNames, string[] RegionNames, string[]
FamilyNames, DateTime Created,
    string SellerName, string SellerPhone, long SellerCared,
long SellerSold, long SellerInstructions,
    long CareTakerCared, long CareTakerSold, long
CareTakerInstructions, Picture[] Images
)
{
    public string CreatedHumanDate => Created.Humanize();
    public string CreatedDate => Created.ToShortDateString();
}

public record PlantPostParams(
    string? PlantName,
    decimal? LowerPrice,
    decimal? TopPrice,
    DateTime? LastDate,
    string[]? FamilyNames,
    string[]? RegionNames,
    string[]? SoilNames) : ISearchParams;

public record PostSearchViewResultItem(Guid Id, string
PlantName, string Description, Picture[] Images, double Price);
```

PlantInformation:

```
namespace Plants.Aggregates;

// Queries

public record GetTotalStats :
IRequest<IEnumerable<TotalStatsViewResult>>;
public record GetFinancialStats(DateTime? From, DateTime? To) :
IRequest<IEnumerable<FinancialStatsViewResult>>;
```

```
public record GetUsedPlantSpecifications :
IRequest<PlantSpecifications>;

// Types

public record TotalStatsViewResult(string FamilyName, decimal
Income, long Instructions, long Popularity);
public record FinancialStatsViewResult(decimal Income, string
FamilyName, long SoldCount, long PercentSold);
public record PlantSpecifications(HashSet<string> Families,
HashSet<string> Regions, HashSet<string> Soils);
```

PlantStock:

```
using Humanizer;

namespace Plants.Aggregates;

// Commands

public record AddToStockCommand(CommandMetadata Metadata,
PlantInformation Plant, DateTime CreatedTime, byte[][] Pictures)
: Command(Metadata);
public record StockAddedEvent(EventMetadata Metadata,
PlantInformation Plant, DateTime CreatedTime, Picture[]
Pictures, string CaretakerUsername) : Event(Metadata);

public record EditStockItemCommand(CommandMetadata Metadata,
PlantInformation Plant, byte[][] NewPictures, Guid[]
RemovedPictureIds) : Command(Metadata);
public record StockEdditedEvent(EventMetadata Metadata,
PlantInformation Plant, Picture[] NewPictures, Guid[]
RemovedPictureIds) : Event(Metadata);

public record PostStockItemCommand(CommandMetadata Metadata,
decimal Price) : Command(Metadata);
public record StockItemPostedEvent(EventMetadata Metadata,
string SellerUsername, decimal Price, string[] FamilyNames) :
Event(Metadata);

// Queries

public record GetStockItems(PlantStockParams Params,
QueryOptions Options) :
IRequest<IEnumerable<StockViewResultItem>>;
public record GetStockItem(Guid StockId) :
IRequest<PlantViewResultItem?>;
public record GetPrepared(Guid StockId) :
IRequest<PreparedPostResultItem?>;

// Types

public record PlantStockParams(bool IsMine) : ISearchParams;
public record StockViewResultItem(Guid Id, string PlantName,
string Description, bool IsMine);

public record PlantInformation(
    string PlantName, string Description, string[] RegionNames,
    string[] SoilNames, string[] FamilyNames
    );

public record PlantViewResultItem(string PlantName, string
Description, string[] FamilyNames,
    string[] SoilNames, Picture[] Images, string[] RegionNames,
DateTime Created)
{
    public string CreatedHumanDate => Created.Humanize();
    public string CreatedDate => Created.ToShortDateString();
}


public record PreparedPostResultItem(
    Guid Id, string PlantName, string Description, string[]
SoilNames,
    string[] RegionNames, string[] FamilyNames, DateTime
Created,
    string SellerName, string SellerPhone, long SellerCared,
long SellerSold, long SellerInstructions,
    long CareTakerCared, long CareTakerSold, long
CareTakerInstructions, Picture[] Images)
{
    public string CreatedHumanDate => Created.Humanize();
    public string CreatedDate => Created.ToShortDateString();
}
```

User:

```
namespace Plants.Aggregates;

// Commands

public record CreateUserCommand(CommandMetadata Metadata,
UserCreationDto Data) : Command(Metadata);
public record UserCreatedEvent(EventMetadata Metadata,
UserCreationDto Data) : Event(Metadata);

public record ChangeRoleCommand(CommandMetadata Metadata,
UserRole Role) : Command(Metadata);
public record RoleChangedEvent(EventMetadata Metadata, UserRole
Role) : Event(Metadata);

public record ChangeOwnPasswordCommand(CommandMetadata Metadata,
string OldPassword, string NewPassword) : Command(Metadata);
public record ChangePasswordCommand(CommandMetadata Metadata,
string Login, string OldPassword, string NewPassword) :
Command(Metadata);
```

```
public record PasswordChangedEvent(EventMetadata Metadata) :
Event(Metadata);

// Queries

public record SearchUsers(UserSearchParams Parameters,
QueryOptions Options) :
IRequest<IEnumerable<FindUsersResultItem>>;
public record GetOwnUsedAddresses : IRequest<AddressViewResult>;

// Types

public record UserCreationDto(string FirstName, string LastName,
string PhoneNumber, string Login, string Email, string Language,
UserRole[] Roles);

public record AddressViewResult(List<DeliveryAddress>
Addresses);
public record UserSearchParams(string Name, string Phone,
UserRole[] Roles) : ISearchParams;
public record FindUsersResultItem(Guid Id, string FullName,
string Mobile, string Login, UserRole[] RoleCodes);
```