МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,
СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ
им. проф. М. А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)

А. В. Параничев

ПРОГРАММИРОВАНИЕ КРИТИЧЕСКИХ СЕРВИСОВ

ПРАКТИКУМ

САНКТ-ПЕТЕРБУРГ
2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

ПРЕДМЕТНАЯ ОБЛАСТЬ ПРАКТИКУМА

Понятие «критический сервис» (critical service) не определено однозначно не только в практике российских стандартов [1], но и в международных нормативных документах по информационных технологиям [2]. В этой связи, для установления предметной области «Программирование критических сервисов», следует привести несколько определений [1–2] (табл. 1).

Таблица 1

Установление предметной области «критический сервис»
в действующих нормативно-правовых документах (2021 г.)

№	Термин	Определение	Источник
1	услуга (service)	«Возможность определенного уровня и нижерасположенных уровней, предоставляемая объектам следующего верхнего уровня».	[1, 109]
2	service	1) «Средства предоставления пользовательского контента» ("means of delivering value for the customer"). 2) «Выполнение рабочих процессов» ("performance of activities"). 3) «Интерактивно переключаемый режим управления информацией» ("behavior, triggered by an interaction").	[2, 407]
3	критическая секция (critical section)	«Часть асинхронной процедуры, которая не может выполняться параллельно с определенной частью той же или другой асинхронной процедуры».	[1, 45]
4	critical information	«Информация, описывающая безопасное использование программного обеспечения в контексте секретности создаваемого информационного контента, либо в контексте защиты персональной информации, создаваемой или хранимой в приложении» ("information describing the safe use of the software, the security of the protection of the sensitive personal information created by or stored with the software")	[2, 109]

Сопоставляя определения, приведенные в табл. 1 в контексте задач программирования, можно понятие «критический сервис» (critical service) сформулировать следующим образом: «обособленный набор программных модулей, содержащий информацию, доступ к которой должен быть строго ограничен». В таком контексте существует ряд актуальных задач, решаемых IT-специалистами, в том числе:

- разработка компонентов программного обеспечения, обеспечивающих безопасный доступ к данным пользователя в мессенджере [3–5];
- построение / сопровождение Full-stack веб-приложений, основанных на микросервисной архитектуре (Microservice Architecture, MSA) доступа к данным [6–8].

В данном учебном пособии представлены этапы проектирования обоих видов задач, при этом:

- разработка критических сервисов рассматривается для мессенджеров Telegram и WhatsApp в контексте реализации базовых функций на языке Python [3–5; 9–10] (практические работы № 01–03);
- развертывание микросервисной архитектуры при построении критических сервисов авторизации (Login service), пользовательских настроек (Account Service), поиска товаров (Search Service) и их оплаты (Checkout Service) [6–8; 11–12] (практические работы №04–11).

Проектирование сервисных компонентов (service component) [2] осуществляется в соответствии со стандартом UML 2 [13–14], основываясь на базовых шаблонах проектирования [15–16].

ОБЩИЕ ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТОВ ПО ПРАКТИЧЕСКИМ РАБОТАМ

Практические работы следует выполнять в виде отчетов в формате docx, соответствующие диаграммы создаются в онлайн-редакторе PlantUML [17; 18] и добавляются в отчет. После выполнения каждой из практических работ создается отдельный docx-файл по следующему шаблону:

PCS_P**01_12**_Familiya_Imya.docx

где **01** – соответствует номеру практической работы;

- ***12*** – индивидуальный номер варианта заданий, равный последним двум цифрам зачетной книжки; если номер зачетной книжки состоит из одного знака, то его необходимо дополнить нулем слева (*например, для зачетной книжки № 7, индивидуальный номер варианта заданий равен* ***07***); если номер оканчивается на 00, то необходимо взять предшествующие две цифры подряд, не равные нулю одновременно (*например, для зачетной книжки № 1234500 индивидуальный номер варианта заданий равен* ***50****; а для № 67890000 – составляет* ***90***).
- **Familiya** – фамилия студента латиницей;
- **Imya** – имя студента латиницей.

СОДЕРЖАНИЕ ПРАКТИКУМА

В данном учебном пособии представлены общие указания по выполнению и оформлению результатов практических занятий по разделу «Разработка критических сервисов для развертывания MSA» дисциплины «Программирование критических сервисов».

Практикум содержит 11 практических работ, в ходе выполнения которых следует составить 10 различных диаграмм по стандарту UML 2 (Deployment, Use Case, Profile, State Machine, Sequence, Activity, Package, Object, Class, Communication) и две SADT-диаграммы (IDEF0, IDEF1X), в соответствии с индивидуальным номером варианта заданий. По каждой из 12 UML- и SADT-диаграмм:

- приводятся и иллюстрируются (рис. 1–14) краткие теоретические сведения, необходимые для выполнения индивидуальных заданий;
- формулируется типовая задача и устанавливаются исходные данные (табл. 2–4);
- приводятся пояснения по решению сформулированной задачи и оформлению результатов в отчете.

Практическое занятие № 01. РАЗРАБОТКА КРИТИЧЕСКИХ СЕРВИСОВ ДЛЯ РАЗВЕРТЫВАНИЯ MSA. UML-ДИАГРАММА РАЗВЕРТЫВАНИЯ

ПОСТРОЕНИЕ UML-ДИАГРАММЫ РАЗВЕРТЫВАНИЯ

Диаграмма развертывания (Deployment) «определяет конструкции для определения архитектуры выполнения действий в системах (execution architecture of systems), а также назначение программных артефактов (software artifact) по отношению к элементам системы» [13, 653].

При построении UML-диаграммы развертывания используются следующие основные элементы (рис. 1), представленные в [13–14; 17].

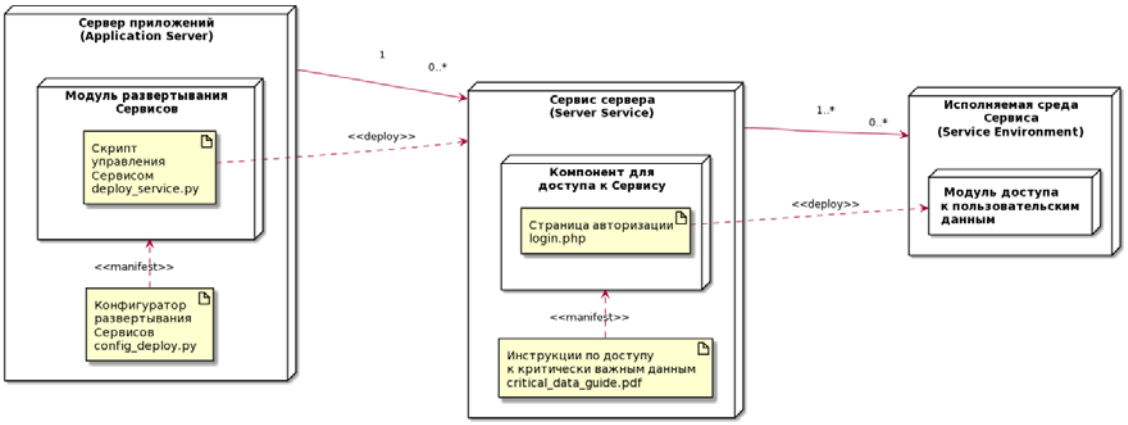

Рис. 1. Основные элементы UML-диаграммы развертывания

На рис. 1 показаны следующие основные блоки и типовые взаимосвязи диаграммы развертывания [13–14]:

- блоки:

o **узел (node)** – «управляющий объект программного обеспечения» ("can host some software"), в качестве которого может выступать устройство аппаратного обеспечения (hardware) или «среда выполнения (execution environment),.. например, операционная система (operating system) или объект контейнеризации (container process)»;

o **артефакт (artifact)** – «декларирование элементов программного обеспечения на физическом носителе (phisycal manifestation of software), как правило, в виде файлов»;

- взаимосвязи:

o **<<deploy>>** (развертывание) – «распределение памяти для артефакта или его сущности под физически развертываемый объект» ("allocation of an artifact or artifact instance to a deployment target") [13, 661].

o **<<manifest>>** (декларация) – «конкретная физическая визуализация одного или более элементов с помощью артефакта» ("concrete physical rendering of one or more model elements by an artifact") [13, 664].

ЗАДАЧА 01

Построить диаграмму развертывания Full-stack приложения, определяя требования к развертыванию микросервисной архитектуры. Среда разработки приложения: PyCharm [19] (язык программирования: Python), среда выполнения кода: Интернет-браузер на платформе Chromium, Firefox или Opera: исходные данные следует взять из табл. 2.

Таблица 2

Исходные данные для установления требований к развертыванию микросервисной архитектуры (MSA)

Значение	По цифре *y*				По цифре *z*		
Цифры (*yz*) [1]	Типовые операции [2]	Интерфейсы [3] (API)	Разрешенные типовые операции для заданных ролей [4]	Условие изменения роли User	Мессенджер	Браузер для тестирования [5] в Selenium	Доступ к контенту БД [6]
0	CRUD	SOAP / XML	Guest: **R**; User: **CR**; TopUser: **CRD**; Admin: **CRUD**	TopUser, если сообщений 10 и более	WhatsApp	Chrome	Логин и пароль (более 8 букв / цифр)
1	CQRS	SOAP / AJAX	Guest: **CQ**; User: **CQR**; IgnUser: **C**; Admin:**CQRS**	IgnUser, если 5 одинаковых сообщений подряд	WhatsApp	Chrome	API token
2	CQRS	RESTful / JSON	Guest: **CQ**; User: **CQR**; IgnUser: **C**; Admin: **CQRS**	IgnUser, если 2 раза подряд отвечает сообщением вопроса	WhatsApp	Chrome	SSH key
3	CQRS	RESTful / HTTP	Guest: **CQ**; User: **CQR**; IgnUser: **C**; Admin: **CQRS**	IgnUser, если не пытался ответить на 4 вопроса подряд	WhatsApp	Firefox	API token
4	BREAD	SOAP / AJAX	Guest: **BR**; User: **BRD**; TopUser: **ABRD**; Admin: **BREAD**	TopUser, если отвечает в 2 раза быстрее времени ожидания	WhatsApp	Firefox	SSH key
5	BREAD	RESTful / JSON	Guest: **BR**; User: **CBR**; TopUser: **B**; Admin: **BREAD**	TopUser, если сообщений 4 и более за сутки	Telegram	Firefox	Логин и пароль (от 8 до 10 букв / цифр)

Таблица 2 (Окончание)

Значение	По цифре *y*				По цифре *z*		
Цифры (*yz*) [1]	Типовые операции [2]	Интерфейсы [3] (API)	Разрешенные типовые операции для заданных ролей [4]	Условие изменения роли User	Мессенджер	Браузер для тестирования [5] в Selenium	Доступ к контенту БД [6]
6	BREAD	RESTful / HTTP	Guest: **BR**; User: **CBR**; IgnUser: **B**; Admin: **BREAD**	IgnUser, если 6 одинаковых сообщений подряд	Telegram	Chrome	API token
7	CRUDL	SOAP / AJAX	Guest: **RL**; User: **RDL**; IgnUser: **L**; Admin: **CRUDL**	IgnUser, если 3 раза подряд отвечает сообщением вопроса	Telegram	Chrome	SSH key
8	CRUDL	RESTful / JSON	Guest: **RL**; User: **CRL**; IgnUser: **L**; Admin: **CRUDL**	IgnUser, если не пытался ответить на 5 вопросов подряд	Telegram	Opera	API token
9	CRUDL	RESTful / HTTP	Guest: **RL**; User: **CRL**; TopUser: **CRDL**; Admin: **CRUDL**	TopUser, если отвечает в 3 раза быстрее времени ожидания	Telegram	Opera	SSH key

Примечания:

1. Цифры *yz* соответствуют двузначному номеру индивидуального варианта задания, определение которого представлено на с. 7 данного учебного пособия.

2. Наборы типовых операций [6–8]: CRUD (Create, Read, Update, Delete; создание, чтение, обновление и удаление), CRUDL (Create, Read, Update, Delete, Listing; создание, чтение, обновление, удаление, пролистывание), BREAD (Browse, Read, Edit, Add, Delete; обзор, чтение, редактирование, добавление, удаление), CQRS (Command, Query, Respond, Segregate; команда, запрос, ответ, отделение).

3. Прикладные интерфейсы (API: Application Programming Interface) [6–8]: XML (Extensible Markup Language; язык текстовой разметки), JSON (JavaScript Object Notation; текстовый формат обмена данными, основанный на JavaScript), SOAP (Simple Object Access Protocol; формат обмена сообщениями через XML), REST (Representation State Transfer; стиль передачи состояния через представление), RESTful (соответствующий ограничениям архитектуры REST); AJAX (Asyncronous JavaScript and XML; асинхронный обмен данными между JavaScript Интернет-браузера и XML веб-сервера).

4. Guest – гостевой пользователь, User – зарегистрированный пользователь, Admin – администратор, TopUser / IgnUser – зарегистрированный пользователь с дополнительными правами / ограничениями.

5. Драйвер тестирования Selenium для Интернет-браузера на платформе Chromium, Firefox или Opera (поддерживаемые операционные системы: Windows, MacOS и Linux) [20].

6. Варианты доступа к базе данных (БД): уникальный идентификатор для API-запросов (API token), ключ SSH (SSH key; Secure Shell: безопасная оболочка для удаленного управления данными).

ПРИМЕР РЕШЕНИЯ ЗАДАЧИ 01

Решение задачи определения требований к MSA для варианта № 00 (табл. 2) приводится по следующим исходным данным:

- типовые операции: **CRUD**; доступ к типовым операциям:
  - **Create**: Admin, User, TopUser;
  - **Read**: Guest, User, Admin, TopUser;
  - **Update**: Admin;
  - **Delete**: Admin, TopUser;
- условие изменения роли User: становится TopUser, если сообщений 10 и более;
- интерфейсы: SOAP / XML;
- мессенджер: WhatsApp;
- драйвер тестирования Selenium: для Интернет-браузеров на платформе Chromium;
- доступ к контенту БД: логин и пароль с более 8 буквами или цифрами.

Соответствующая диаграмма развертывания представлена на рис. 2.

Ориентируясь на рис. 2, необходимо установить следующие требования при построении Full-stack приложения на основе MSA [6–8; 11–12; 19–25]:

- соответствие драйвера Selenium для задач автоматизированного тестирования: для установленного на компьютере Интернет-браузера Google Chrome (версия 91.0.4472.101 для 64-разрядной Windows 7) выбрана предшествующая версия архива chromedriver_win32.zip (версия 90.0.4430.24 для операционной системы, совместимой с 32-разрядной Windows);
- взаимодействие с веб-сервером, используя Python, может осуществляться с помощью Spyne (выполняет RPC на языке Python; Remote Procedure Call: технология удаленного вызова процедур) [21–22]: входной протокол XML Document, выходной протокол SOAP 1.1 (в случае взаимодействия на основе RESTful / JSON, потребуется также указать варианты выбора фреймворка, такого как: Django Rest Framework, Flask и Pyramid) [23];
- на основе логина и пароля, требования к которым определяются регулярным выражением «**[A-Za-z0-9]{8,}**» (что можно проверить, например, в [24]), должен быть сформирован токен (уникальный идентификатор) API, доступ к которому есть только у администратора (роль Admin); при наличии токена API или ключа SSH в исходных данных, пара «логин–пароль» уже не нужна (при этом, для ключа SSH следует указать защищенную среду доступа, например, PuTTY при выполнении разработки в операционной системе Windows [25]);

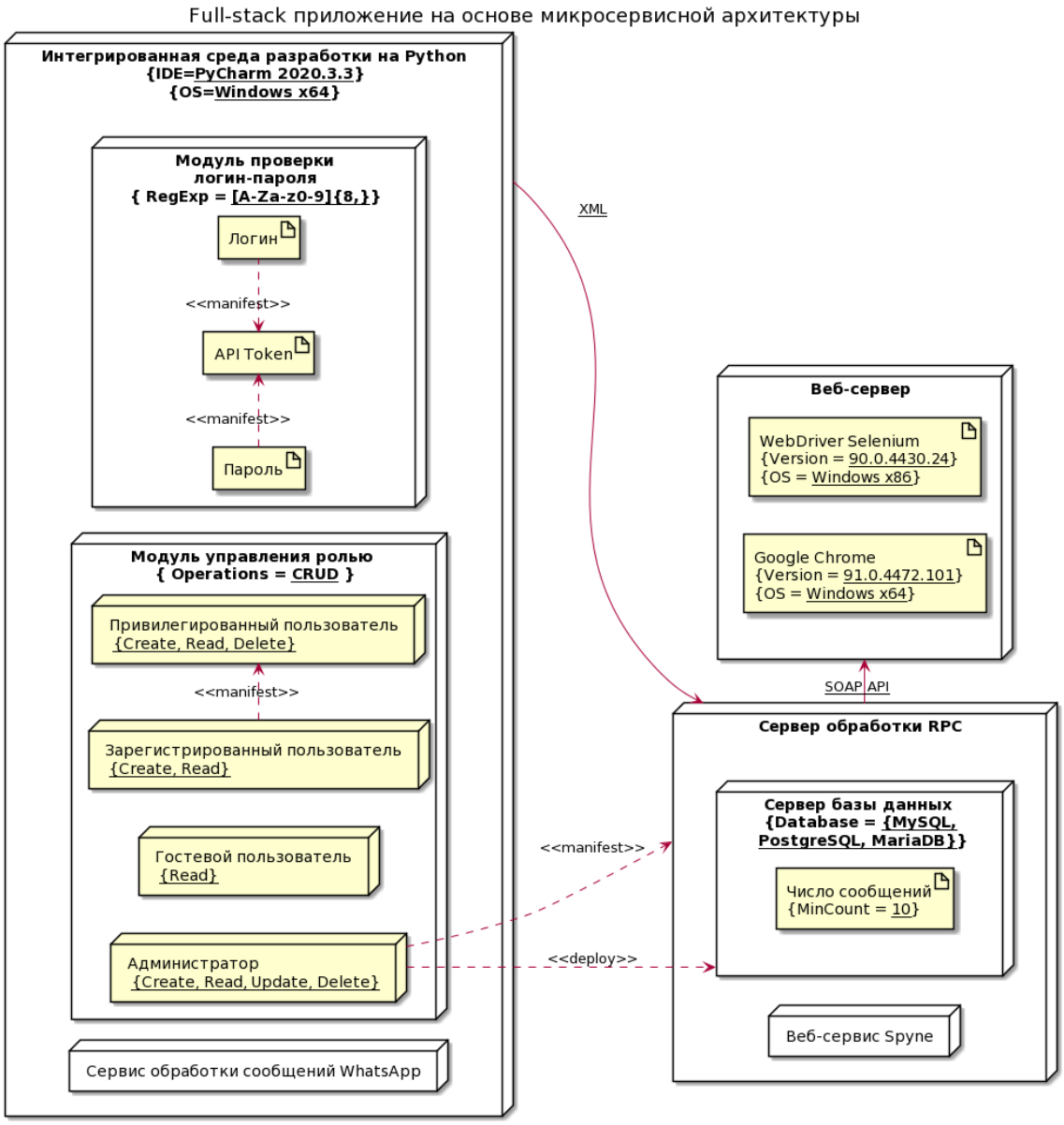

Рис. 2. Диаграмма развертывания для определения требований к MSA

- СУБД должна быть реляционной, поскольку доступ задан с помощью XML (в случае интерфейса RESTful создаваемая БД, как правило нереляционная: MongoDB, Redis, CouchDB, DynamoDB, Cassandra; вместе с тем, СУБД PostgreSQL поддерживает форматы как XML, так и JSON) [7];
- при числе сообщений более 10, пользователь, имеющий роль User, получает привилегии роли TopUser.

Практическое занятие № 02. РАЗРАБОТКА КРИТИЧЕСКИХ СЕРВИСОВ ДЛЯ РАЗВЕРТЫВАНИЯ MSA. UML-ДИАГРАММА ПРЕЦЕДЕНТОВ

ПОСТРОЕНИЕ UML-ДИАГРАММЫ ПРЕЦЕДЕНТОВ

Диаграмма прецедентов (Use Case) является «средством обозначить требования к системе (means to capture the requirements of systems)» [13, 639].

При построении UML-диаграммы прецедентов (вариантов использования) используются следующие основные элементы (рис. 3), представленные в [13–14; 17].

Рис. 3. Основные элементы UML-диаграммы прецедентов

На рис. 3 показаны следующие основные блоки и типовые взаимосвязи диаграммы вариантов использования [13–14]:

- блоки:

o **актор** (**actor**; также называют «**агент**» (**agent**) и «**роль**» (**role**)) – типовой активный объект, поведение которого является внешним по отношению к проектируемой системе;

o **прецедент** (**use case** (usecase); также называют «**вариант использования**») – «набор сценариев, связанных общей целью» ("a set of scenarious tied together by a common user goal") [14];

o **предметная область прецедентов** (subject for a set of Use Cases; также называют «**рамки системы**» (**system boundary**)) определяет границы системы, внутри которых взаимодействие с акторами приводит к рассматриваемым результатам;

- взаимосвязи:

o **<<include>>** (включение) – означает, что есть «общие части поведения двух или более прецедентов» ("common parts of the behavior of two or more Use Cases"): «некоторая общая часть может извлекаться в другой прецедент» ("this common part is then extracted to a separate Use Case"), либо определяется иерархическая композиция вариантов использования прецедентов ("relationship allows hierarchical composition of Use Cases") [13, 641];

o **<<extend>>** (расширение) – определяет, каким образом «добавочное поведение следует дополнить для других прецедентов, в соответствии с заданными условиями» ("additional behavior that should be added, possibly conditionally, to the behavior defined in one or more Use Cases") [13, 640];

o **<<inherit>>** (наследование: обычно указывается в виде стрелки с полым наконечником) – определяет создание роли или прецедента на базе другой роли или прецедента, соответственно.

Составление диаграммы вариантов использования часто составляют в следующей последовательности [14]:

1) выполняют текстовое описание задачи в виде сценария, выделяя типовые действия для основных ролей;

2) упорядочивают сценарий, группируя роли и определяя принадлежность типовых действий к категории <<include>> или <<extend>>;

3) выполняют графическое описание задачи, определяя рамки проектируемой системы.

Пример выполнения этапа 1 при построении диаграммы, представленной на рис. 3, выглядит следующим образом:

- *Пользователь* желает воспользоваться сервисом, в котором представлены его данные.
- *Пользователь* получает доступ к своим данным после авторизации.
- *Разработчик сервисов* изучает инструкции по доступу к критически важным данным.
- *Разработчик сервисов* создает компонент для доступа к сервису, руководствуясь инструкциями по доступу к критически важным данным.
- *Разработчик сервисов* создает страницу авторизации для того, чтобы *Пользователь* получил доступ к своим данным.
- *Разработчик сервисов* также может выполнять действия, определенные для *Пользователя*.
- *Разработчик сервера приложений* конфигурирует развертывание сервисов создаваемых сервисов.
- *Разработчик сервера приложений* создает скрипт управления сервисом на основе настроек по развертыванию сервисов, чтобы *Разработчик сервисов* получил доступ к созданию сервисных компонентов.
- Разработчик сервера приложений также может выполнять действия, определенные для Разработчика сервисов.

Далее следует выполнение этапа 2, что позволяет структурировать текст следующим образом:

- *Пользователь* получает доступ к пользовательским данным.

- *Разработчик сервисов* изучает инструкции по доступу к критически важным данным, в том числе (<<include>>), создает компонент для доступа к сервису, что, в том числе (<<include>>), включает создание страницы авторизации, которая позволяет (<<extend>>) *Пользователю* получить доступ к сохраняемым данным. *Разработчик сервисов* наследует (<<inherit>>) возможности *Пользователя*.
- *Разработчик сервера приложений* конфигурирует развертывание сервисов, в том числе (<<include>>), создает скрипт управления сервисом, что позволяет (<<extend>>) *Разработчику сервисов* создать компонент доступа для авторизации. *Разработчик сервера приложений* наследует (<<inherit>>) возможности *Разработчика сервисов*.

После этого выполняется построение UML-диаграммы прецедентов: на рис. 3 представлен пример выполнения этапа 3.

Следует отметить, что в стандарте [13] приведены достаточно общие рекомендации по созданию диаграммы Use Cases, поэтому представленная интерпретация является далеко не единственной [14; 26].

ЗАДАЧА 02

Построить UML-диаграмму прецедентов Full-stack приложения, определяя варианты использования для установления требований к развертыванию MSA. Исходные данные следует взять из табл. 2.

ПОЯСНЕНИЯ ПО РЕШЕНИЮ ЗАДАЧИ 02

Для решения данной задачи необходимо выполнить последовательность действий для указанных ролей и соответствующих типовых операций, как это представлено на рис. 3, построение которого основано на составлении сценария (представленного в предыдущем разделе), и рис. 1. Иными словами, сценарий нужно составить, ориентируясь и упрощая диаграмму развертывания, построение которой выполнено в предыдущей практической работе.

Результатом выполнения практической работы является диаграмма прецедентов, составление которой следует подробно объяснить.

Практическое занятие № 03. РАЗРАБОТКА КРИТИЧЕСКИХ СЕРВИСОВ ДЛЯ РАЗВЕРТЫВАНИЯ MSA. UML-ДИАГРАММА ПРОФИЛЕЙ

ПОСТРОЕНИЕ UML-ДИАГРАММЫ ПРОФИЛЕЙ

Диаграмма профилей (Profile) «определяет то, как программные компоненты в виде расширений с ограничениями относятся к базовой метамодели (высокоуровневой модели), – с целью адаптации такой метамодели к определенной платформе или области определения» (defines limited extensions to a reference metamodel with the purpose of adapting the metamodel to a specific platform or domain) [13, 278].

При построении UML-диаграммы профилей используются следующие основные элементы, представленные в [13; 27] (рис. 4: также учтены тезисы, изложенные в [28]).

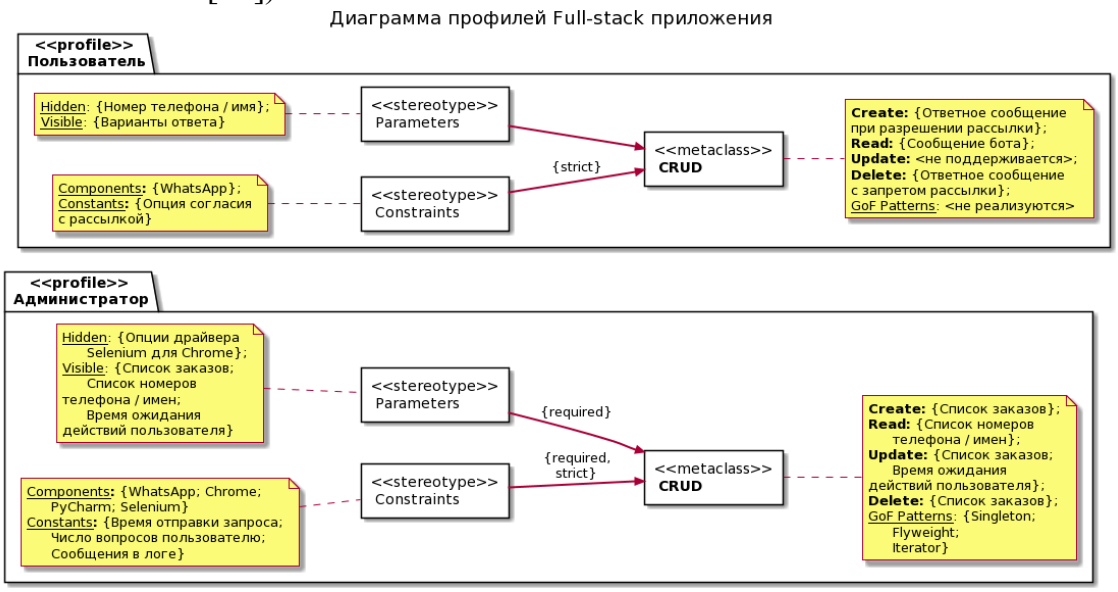

Рис. 4. Основные элементы UML-диаграммы профилей

На рис. 4 показаны следующие основные блоки и типовые взаимосвязи диаграммы профилей:

- блоки:

o **<<profile>>** (**профиль данных**) «может применяться отдельно, при этом должен определить любые требуемые взаимосвязи с метаклассами и/или метамоделями, если они содержат какие-либо стереотипы» ("can be applied individually,.. must specify any required metaclass and/or metamodel references if it contains any Stereotypes") [13, 259];

o **<<metaclass>>** (**метакласс**, высокоуровневый класс; также называют «расширение» (extension)) определяет «способность к «гибкому» добавлению/удалению стереотипов в классы» ("ability to flexibly add (and later remove) stereotypes to classes") [13, 273];

o **<<stereotype>>** (**стереотип**) «определяет, каким образом существующий метакласс можно расширить, обеспечивая доступ для использования

специализированной терминологии для платформы или области применения, либо для нотации (системы обозначений)» ("defines how an existing metaclass may be extended, and enables the use of platform or domain specific terminology or notation ") [13, 280];

- взаимосвязи:

o **{required}** (необходимая взаимосвязь) «указывает на то, что экземпляр расширяющего стереотипа необходимо создать, если экземпляр расширяемого класса существует» ("indicates whether an instance of the extending stereotype must be created when an instance of the extended class is created") [13, 273];

o **{strict}** (строгая взаимосвязь) «устанавливает, что правила фильтрации данных внутри профиля следует строго применять для метаклассов» (specifies that the Profile filtering rules for the metaclasses… shall be strictly applied) [13, 279].

Взаимодействие отдельных блоков профилей данных (<<profile>>) следует рассматривать в контексте диаграммы пакетов данных (Package), представленной в практической работе №8 данного пособия.

ЗАДАЧА 03

Построить диаграмму профилей Full-stack приложения, определяя требования к развертыванию MSA по табл. 2 и 3.

Таблица 3

Исходные данные для определения настроек приложения

Значение / Цифра (*yz*) [1]	Параметр по цифре *y*		Параметр по цифре *z*
	Видимые сущности в стереотипе параметров администратора	Константы в стереотипе ограничений администратора	Шаблоны GoF-стандарта[2] в метаклассе операций администратора
0	Список заказов; Список номеров телефона / имен; Время ожидания действий пользователя	Время отправки и число вопросов пользователю; Сообщения в логе	Singleton; Flyweight; Iterator
1	Расписание дня; Время напоминания; Сообщения в логе о всех действиях	Список номеров телефона / имен; Время ожидания ответа и число напоминаний	Singleton; Proxy; Template Method
2	Список хранимых сообщений; Список номеров телефона / имен; Время ожидания действий пользователя	Максимальное число хранимых сообщений; Время отправки уведомления о числе сообщений; Сообщения в логе	Factory Method; Bridge: Chain of Responsibility

Таблица 3 (Окончание)

Значение / Цифра (*yz*) [1]	Параметр по цифре *y*		Параметр по цифре *z*
	Видимые сущности в стереотипе параметров администратора	Константы в стереотипе ограничений администратора	Шаблоны GoF-стандарта[2] в метаклассе операций администратора
3	Время отправки информации о заказе; Список номеров телефона / имен; Сообщения в логе о всех действиях	Список дней самовывоза; Список интервалов времени доставки; Время ожидания ответа от пользователя	Factory Method; Adapter; Visitor
4	Список номеров телефона / имен; Маска номера телефона / имени; Время ожидания действий пользователя	Время отправки вопроса; Максимальное число телефонов / имен; Сообщения в логе	Prototype; Façade; Memento
5	Опции заказа товара: название и цена; Время ожидания действий пользователя Сообщения в логе о всех действиях	Список номеров телефона / имен; Список имеющихся товаров и цен на них; Время отправки запроса	Prototype; Proxy; State
6	Список дел по дням недели; Время напоминания; Сообщения в логе о всех действиях	Время ожидания сообщения о результате; Список номеров телефона / имен; Максимальное число дел	Builder; Composite; Observer
7	Список хранимых сообщений; Список номеров телефона / имен; Сообщения в логе о всех действиях	Максимальное число хранимых сообщений; Время отправки уведомления о числе сообщений; Время ожидания ответа от пользователя	Builder; Decorator; Mediator
8	Список номеров телефона / имен; Маска номера телефона / имени; Сообщения в логе о всех действиях	Время отправки вопроса и ожидания ответа; Максимальное число телефонов / имен	Abstract Factory; Façade; Strategy
9	Опции заказа товара: название и цена; Время отправки запроса и ожидания ответа	Список номеров телефона / имен; Список имеющихся товаров и цен на них; Сообщения в логе	Abstract Factory; Flyweight; Command

Примечания:

1. Цифры *yz* соответствуют двузначному номеру индивидуального варианта задания, определение которого представлено на с. 7 данного практикума.

2. Следует объяснить применение шаблонов проектирования (в параметре *y* задано по одному паттерну каждого из трех видов: порождающий (Creational pattern), структурирующий (Structural pattern) и поведенческий (Behavioral pattern)) для создания «гибкого» решения при заданных параметрах и ограничениях (параметр по цифре *x*), ориентируясь на источники [15–16; 29–30].

ПОЯСНЕНИЯ ПО РЕШЕНИЮ ЗАДАЧИ 03

Пример диаграммы профилей для варианта № 00 приведен на рис. 4, из которого видно, что (табл. 3):

- видимые сущности в стереотипе *параметров* администратора:
  - список заказов;
  - список номеров телефона / имен;
  - время ожидания действий пользователя;
- константы в стереотипе *ограничений* администратора:
  - время отправки и число вопросов пользователю;
  - сообщения в логе;
- шаблоны GoF-стандарта в метаклассе *операций* администратора:
  - порождающий паттерн: Singleton;
  - структурирующий паттерн: Flyweight;
  - поведенческий паттерн: Iterator.

Типовые *параметры*, *ограничения* и *операции* представлены на рис. 4, учитывая данные предыдущей практической работы:

- профиль администратора (Admin) определен следующими элементами:
  - скрытые требуемые параметры: опции драйвера Selenium для Chrome;
  - видимые требуемые параметры: список заказов, список номеров телефона / имен, время ожидания действий пользователя;
  - требуемые программные компоненты, определяющие ограничения: WhatsApp, Chrome, PyCharm, Selenium;
  - требуемые константы, определяющие ограничения: время отправки запроса, число вопросов пользователю, сообщения в логе;
  - предметная область операций Create и Delete: список заказов;
  - предметная область операций Update: список заказов и время ожидания действий пользователя;
  - предметная область операции Read: список номеров телефона / имен;
  - применение шаблона Singleton: обеспечить создание единственного WhatsApp-бота для данного аккаунта;
  - применение шаблона Flyweight: определить структуру «один-к-одному» управляемых данных;
  - применение шаблона Iterator: выполнять перечисление заказов через унифицированный объект;

- профиль пользователя (TopUser, User, Guest) определен следующими элементами (при этом не предусмотрены следующие опции метакласса CRUD: предметная область операции Update и применение шаблонов проектирования):
  - скрытые параметры: номер телефона / имя;
  - видимые параметры: варианты ответа;
  - программные компоненты, определяющие ограничения: WhatsApp;
  - константы, определяющие ограничения: опция согласия с рассылкой;
  - действия операции Create: ответное сообщение при разрешении рассылки;
  - действия операции Read: сообщение бота;
  - действия операции Delete: ответное сообщение с запретом рассылки.

Практическое занятие № 04. РАЗРАБОТКА КРИТИЧЕСКИХ СЕРВИСОВ ДЛЯ РАЗВЕРТЫВАНИЯ MSA. UML-ДИАГРАММА СОСТОЯНИЙ

ПОСТРОЕНИЕ UML-ДИАГРАММЫ СОСТОЯНИЙ

Диаграмма состояний (State Machine) определяет «концепты (элементы представления знаний)… для моделирования дискретных режимов работы, управляемых событиями, основываясь на формальном подходе конечных автоматов» ("concepts… for modeling discrete event-driven Behaviors using a finite state-machine formalism") [13, 305].

При построении UML-диаграммы состояний используются следующие основные элементы (рис. 5), представленные в [13–14; 17].

Диаграмма состояний включает следующие типовые элементы [13–16]:

- блоки-состояния:
  - **state** (состояние, объект-состояние (Object for States)) – объект, в котором определено поведение частей системы (в виде конечного автомата, основанного на поведении; Behavior State Machine), либо зафиксирована последовательность выполняемых событий в системе или подсистеме (в виде конечного автомата, основанного на протоколе; Protocol State Machine);
  - **pseudostate** (псевдосостояние) – состояние, предназначенное для выполнения или протоколирования определенного действия в системе или подсистеме:

1) **initial state** (состояние входа в объект-состояние);

2) **final state** (состояние выхода из системы / подсистемы);

3) **shallow history** (состояние, которое возвращает последние действия в объекте-состоянии);

4) **deep history** (состояние, которое возвращает все зафиксированные действия в объекте-состоянии);

5) entry point entry: состояние, ассоциированное с выполнением определенного действия:

5.1) **entry point** (вход в объект-состояние);

5.2) **exit point** (выход из объекта-состояния);

5.3) **junction** (асинхронное объединение нескольких переходов в другое состояние);

5.4) **choice** (выбор действия);

5.5) аварийное завершение (terminate),

5.6) **fork** (разветвитель на несколько параллельно выполняемых действий);

5.7) **join** (синхронное объединение нескольких переходов в другое состояние);

- взаимосвязи внутри состояния:
  - **entry /** – действие при инициализации состояния;
  - **do /** – набор действий во время нормальной работы состояния;
  - **exit** / – действие при завершении состояния.

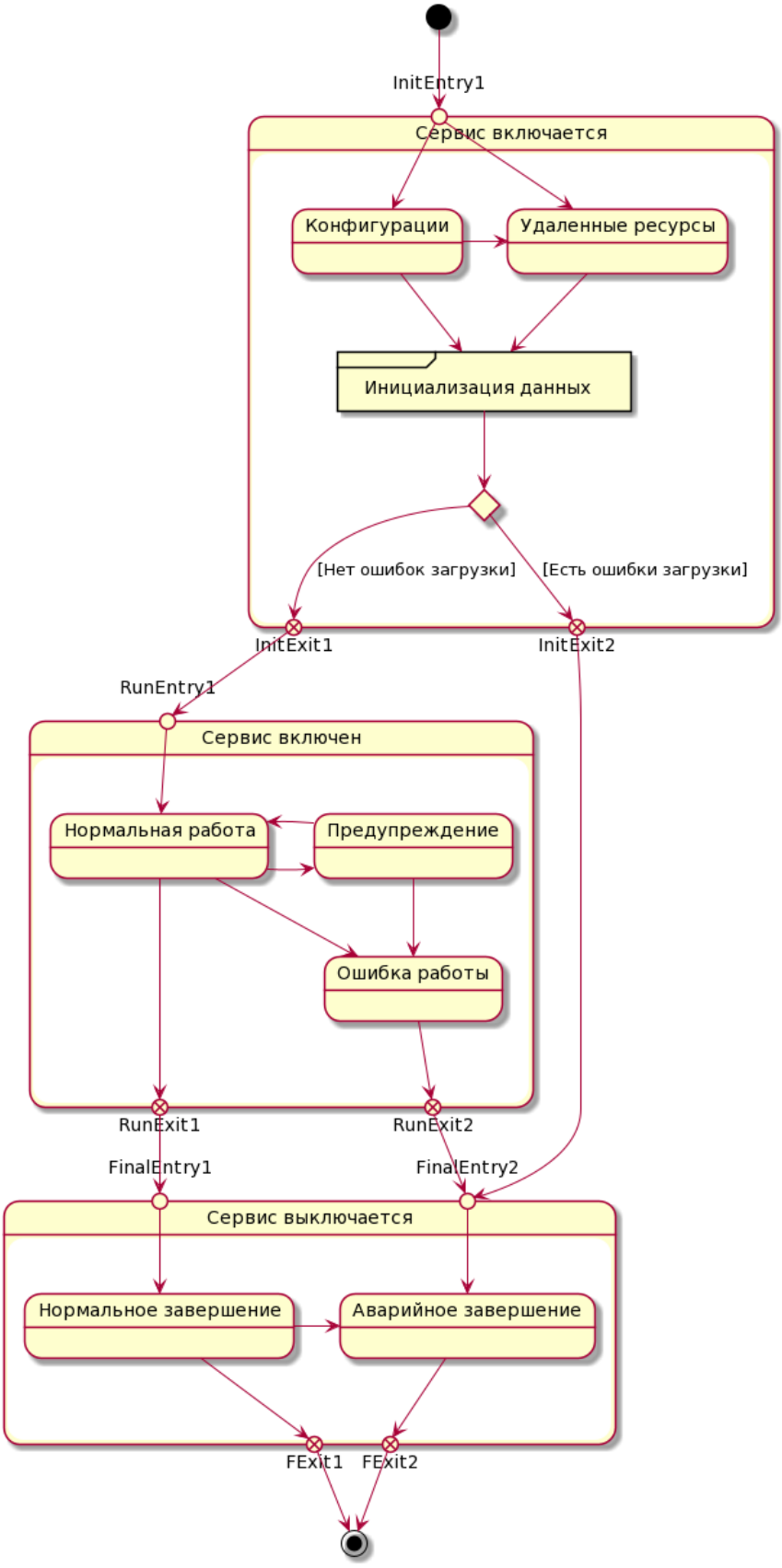

Рис. 5. Основные элементы UML-диаграммы состояний

ЗАДАЧА 04

Выполнить описание состояний сервиса, руководствуясь исходными данными, представленными в табл. 4.

Таблица 4

Исходные данные для определения предметной области и атрибутов сервиса

Значение / Цифра (*yz*) [1]	Параметр по цифре *y*				Параметр по цифре *z*
	Предметная область	Параметры, для которых необходимо установить соотношения «один-к-многим» ("one-many"), «многие-ко-многим» ("many-many") и/или «один-к-одному»			
		"one-many" [2]	"many-many"	"one-one"	Атрибуты [3]
0	Сервис поиска (Search Service) видеорегистраторов	Производители; Диагональ экрана; Варианты разрешения экрана; Варианты наличия / отсутствия заданных опций; Углы обзора	Диагонали экрана и Варианты разрешения экрана	–	QFD
1	Сервис пользовательских настроек (Account Service) операционной системы	Страна; Город; Язык; Операционная система Тема оформления	Страна и язык	–	OWASP
2	Сервис оплаты (Checkout Service) онлайн-курсов повышения квалификации	Вариант оплаты (банковская карточка, подарочный сертификат); Тематика курса; Квалификация	–	Номер участника	CWE
3	Сервис авторизации (Login Service) по логин-паролю с подсказкой	Логин; Фамилия; Имя; Отчество	Логин и 2 символа из Пароля (для подсказки)	–	HATEOAS; ACID
4	Сервис оплаты (Checkout Service) кофе	Вид оплаты (наличные / банковская карточка); Вид кофе; Количество сахара	–	Артикул	HATEOAS; ACID

Таблица 4 (Окончание)

Значение / Цифра (*yz*) [1]	Параметр по цифре *y*				Параметр по цифре *z*
	Предметная область	Параметры, для которых необходимо установить соотношения «один-к-многим» ("one-many"), «многие-ко-многим» ("many-many") и/или «один-к-одному»			
		"one-many" [2]	"many-many"	"one-one"	Атрибуты [3]
5	Сервис пользовательских настроек (Account Service) фитнес-приложения	Категория роста; Категория веса; Язык; Категория активности	Категория роста и Категория активности; Категория веса и Категория активности	–	HATEOAS; ACID
6	Сервис авторизации (Login Service) по номеру телефона /логину и проверочному вопросу	Номер телефона Логин; Фамилия; Имя; Отчество; Проверочный вопрос	Логин и Проверочный вопрос (для подсказки)	–	QFD
7	Сервис поиска (Search Service) автомагнитол	Диагональ дисплея; Тип дисплея; Опции настройки звука; Поддерживаемые интерфейсы	Опции настройки звука и поддерживаемые интерфейсы	–	OWASP
8	Сервис пользовательских настроек (Account Service) предпочтений видеофильмов	Жанр фильма; Возрастные ограничения; Год выхода; Страна; Режиссер	Жанр фильма и Режиссер	–	CWE
9	Сервис оплаты (Checkout Service) бонусными баллами	Уровень участника; Категории товаров	Доступные категории товаров участника	Номер бонусного счета	HATEOAS; ACID

Примечания:

1. Цифры *yz* соответствуют двузначному номеру индивидуального варианта задания, определение которого представлено на с. 7 данного учебного пособия.

2. Для всех вариантов добавить параметр «Роль пользователя», проиндексировав его (путем создания соотношения "one-many").

3. Наборы атрибутов [6–8; 15–16]: QFD (Quality Function Deployment; развертывание функции качества), OWASP (Open Web Application Security Project; открытый проект обеспечения безопасности веб-приложений), CWE (Common Weakness Enumeration; перечисление общих отрицательных качеств), HATEOAS (Hypermedia as the Engine of Application State; гипермедиа в качестве средства запуска состояния приложения), ACID

(Atomic, Consistent, Isolated, Durable; атомарность, согласованность, изолированность и устойчивость данных).

ПОЯСНЕНИЯ ПО РЕШЕНИЮ ЗАДАЧИ 04

Для решения данной задачи необходимо выполнить последовательность действий, аналогично представленным на рис. 5, для своего варианта исходных данных. Результатом выполнения практической работы является диаграмма состояний, составление которой следует подробно объяснить.

Практическое занятие № 05. РАЗРАБОТКА КРИТИЧЕСКИХ СЕРВИСОВ ДЛЯ РАЗВЕРТЫВАНИЯ MSA. UML-ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТИ

ПОСТРОЕНИЕ UML-ДИАГРАММЫ ПОСЛЕДОВАТЕЛЬНОСТИ

Диаграмма последовательности (Sequence) устанавливает «определенную последовательность обмена сообщениями, а также соответствие выполнения этих сообщений на «линии жизни» объектов» ("the sequence of Messages that are exchanged, along with their corresponding Occurrence Specifications on the Lifelines") [13, 595].

При построении UML-диаграммы последовательности используются следующие основные элементы (рис. 6), представленные в [13–14; 17].

Диаграмма последовательности включает следующие основные блоки и типовые взаимосвязи [13–14]:

- блоки:

o **lifeline** («линия жизни» объекта-участника) определяет временную последовательность действий объекта-участника, учитывая хронологию действий других объектов-участников, в контексте выполнения общего для них процесса;

o **occurrence specification (**спецификация действий; область действий (region)) определяет типовые конструкции последовательного взаимодействия объектов-участников:

1) **critical** (блок критически важной последовательности действий объектов-участников, выполняемый обособленно от других взаимодействий);

2) **par** (блок параллельно выполняемых действий);

3) **opt** (блок проверки и выполнения условия «если», по результатам действия, предшествующего данному блоку);

4) **alt** (блок проверки и выполнения условия «если-иначе», по результатам действия, предшествующего данному блоку);

5) **loop** (блок выполнения цикла, условия которого задаются в действии, которое предшествует данному блоку);

6) **break** (блок выхода из данной конструкции, если предшествующий блок выполнен успешно (в противном случае блок игнорируется));

7) **seq** (блок приблизительной последовательности действий);

8) **strict** (блок строгой последовательности действий);

- взаимосвязи:

o **message** (сообщение) определяет передачу информации другому объекту участнику или выполнение функции;

o **reply** (ответ на сообщение или вызов процедуры) определяет ответ на полученное сообщение или возвращаемое значение функции;

o **general ordering** (обобщенная последовательность действий) указывается обобщенное взаимодействие между объектами участниками.

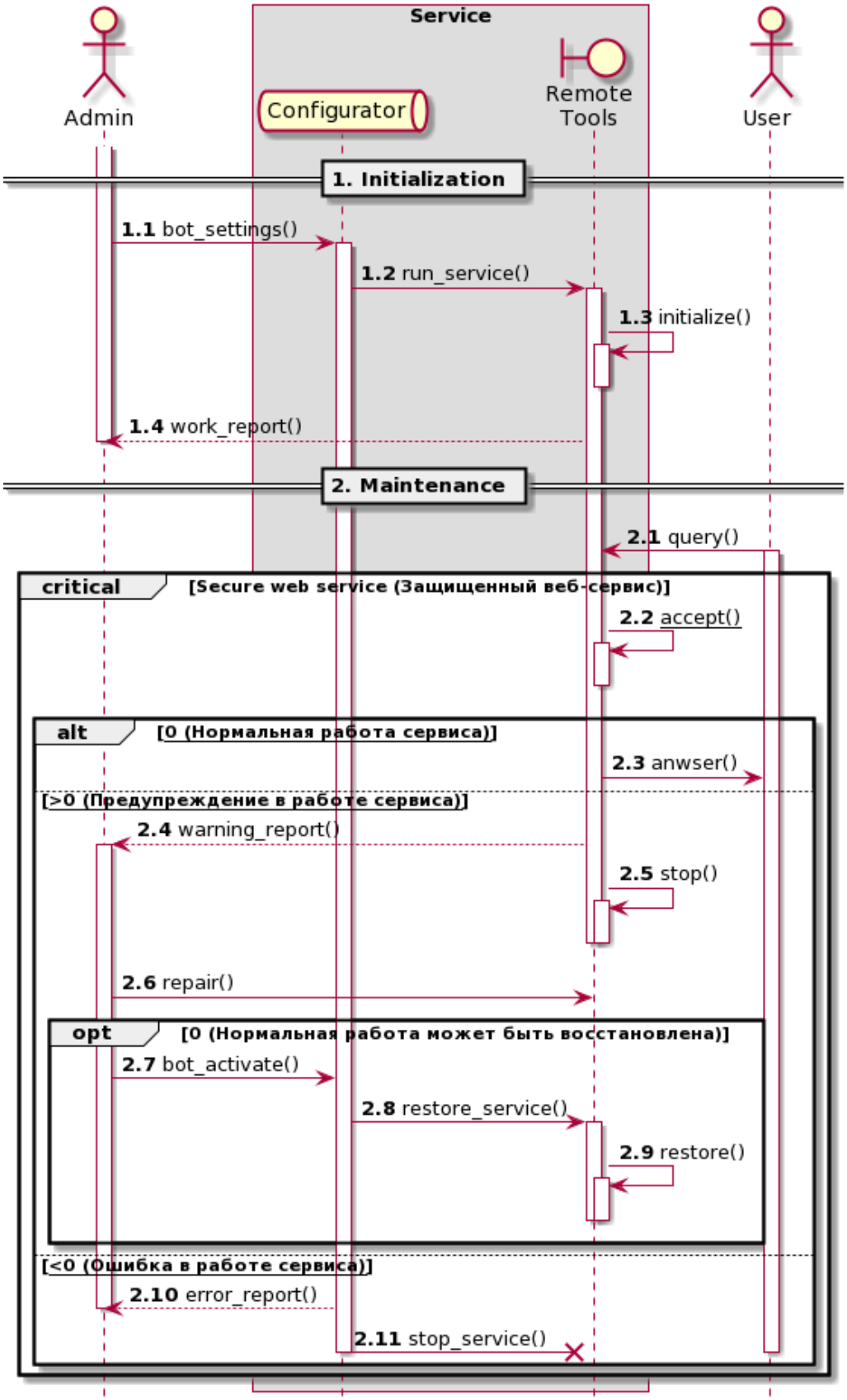

Рис. 6. Основные элементы UML-диаграммы последовательности

ЗАДАЧА 05

Выполнить описание последовательности действий для сопровождения сервиса, исходные данные для которого представлены в табл. 4.

ПОЯСНЕНИЯ ПО РЕШЕНИЮ ЗАДАЧИ 05

Для решения данной задачи необходимо выполнить последовательность действий, аналогично представленным на рис. 6, для своего варианта исходных данных. Результатом выполнения практической работы является диаграмма последовательности, составление которой следует подробно объяснить.

Практическое занятие № 06. РАЗРАБОТКА КРИТИЧЕСКИХ СЕРВИСОВ ДЛЯ РАЗВЕРТЫВАНИЯ MSA. UML-ДИАГРАММА ДЕЯТЕЛЬНОСТИ

ПОСТРОЕНИЕ UML-ДИАГРАММЫ ДЕЯТЕЛЬНОСТИ

Диаграмма деятельности (Activity) определяет «следование подчиненных блоков, основанное на элементах управления, а также на модели передачи данных» ("sequencing of subordinate units, using a control and data flow model") [13, 373].

При построении UML-диаграммы деятельности используются следующие основные элементы (рис. 7), представленные в [13–14; 17].

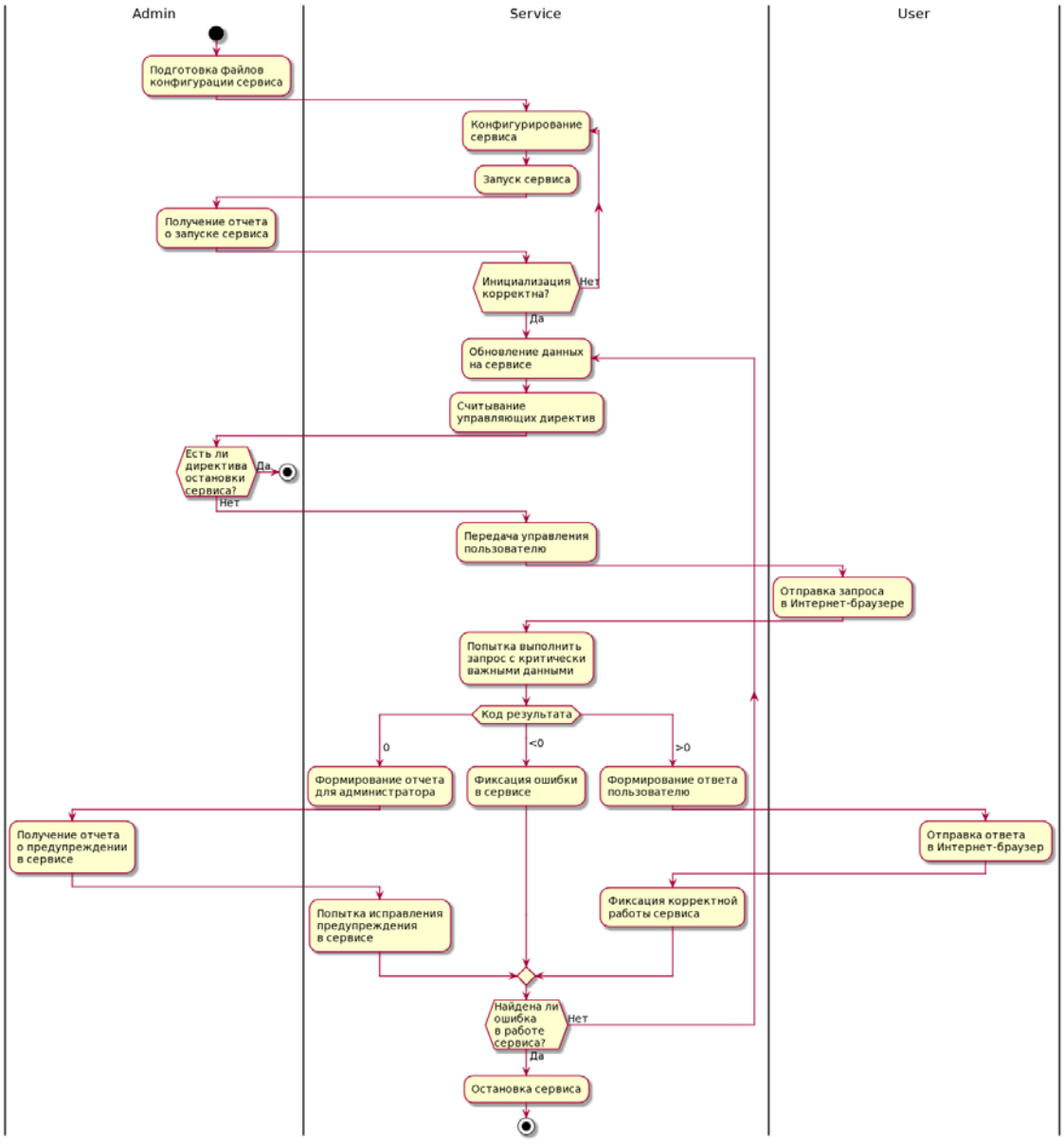

Рис. 7. Основные элементы UML-диаграммы деятельности

Основными блоками и типовыми взаимосвязями диаграммы деятельности являются [13–14]:

- блоки (часто указываются в виде типизированного элемента <<activity>>):
  - **action node** (executable node, «исполняющий узел») «реализует низкоуровневые шаги в рамках всего блока деятельности (activity)» ("embody lower-level steps in the overall Activity");
  - **object node** («объектный узел») хранит и передает данные, получаемые и передаваемые узлу от «исполняющего узла»;
  - **control node** («узел управления») определяет последовательность действий, которые производят «исполняющие узлы»;
- взаимосвязи (представляют собой типизированную взаимосвязь <<swimline>>: на рис. 7 показаны взаимодействия между такими «плавательными» секциями, связанными с соответствующими объектами-участниками):
  - **control flow** («поток данных управления») представляет собой взаимосвязь между «узлами управления»;
  - **object flow** («поток данных объекта») – определяет взаимосвязь между «объектными узлами».

ЗАДАЧА 06

Выполнить описание основных действий по сопровождению сервиса, исходные данные для которого представлены в табл. 4.

ПОЯСНЕНИЯ ПО РЕШЕНИЮ ЗАДАЧИ 06

Для решения данной задачи необходимо выполнить последовательность действий, аналогично представленным на рис. 7, для своего варианта исходных данных. Результатом выполнения практической работы является диаграмма деятельности, составление которой следует подробно объяснить.

Практическое занятие № 07. РАЗРАБОТКА КРИТИЧЕСКИХ СЕРВИСОВ ДЛЯ РАЗВЕРТЫВАНИЯ MSA. UML-ДИАГРАММА ОБЪЕКТОВ

ПОСТРОЕНИЕ UML-ДИАГРАММЫ ОБЪЕКТОВ

Диаграмма объектов (Object) определяет то, каким образом объекты «инстанцируются (создаются их экземпляры), используя спецификации по созданию экземпляров объектов» ("are instantiated using Instance Specifications") [13, 99].

При построении UML-диаграммы объектов используются следующие основные элементы (рис. 8), представленные в [13–14; 17].

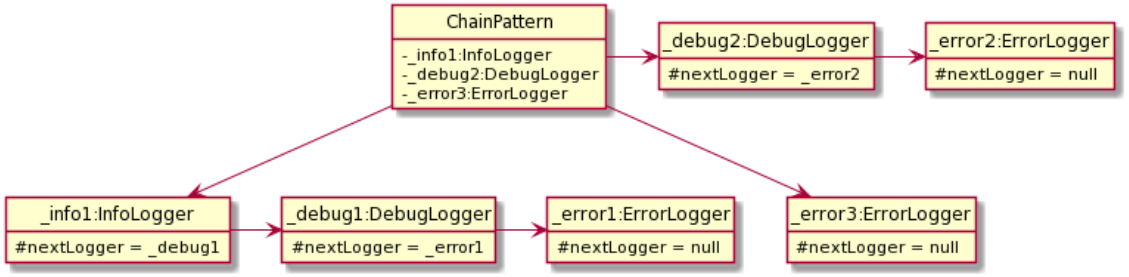

Рис. 8. Основные элементы UML-диаграммы объектов

На рис. 8 показаны следующие основные блоки и типовые взаимосвязи диаграммы объектов [13, 126–129]:

- **instance specification** (спецификация экземпляра объекта) – «представляет возможное или действительное существование экземпляра объекта в моделируемой системе, а также полностью или частично выполняет описание этих объектов» ("represents the possible or actual existence of instances in a modeled system and completely or partially describes those instances"; например, на рис. 8 представлена спецификация экземпляра объекта ChainPattern, которой напрямую или опосредованно подчинены 6 других блоков типа "instance specification");
- **slot** (слот) – «устанавливает, что некоторый экземпляр объекта, смоделированный с помощью блока "instance specification"» ("specifies that an instance modeled by an Instance Specification"), содержит значения, которые «определены на основе спецификации значений» ("are specified using Value Specifications"; на рис. 8 указано 16 слотов (все элементы спецификаций экземпляра объекта), в том числе 6 слотов указаны дважды для их детализации (_info1, _debug1, _error1, _debug2, _error2 и _error3);
- **link** (взаимосвязь) определяет графическую взаимосвязь между двумя спецификациями объектов по их слоту; например, на рис. 8 представлена цепочка из трех взаимосвязей по слотам _info1, _debug1 и _error1, соответственно.

ЗАДАЧА 07

Выполнить описание основных объектов сервиса и взаимосвязей этих объектов по исходным данным, представленным в табл. 4.

ПОЯСНЕНИЯ ПО РЕШЕНИЮ ЗАДАЧИ 07

Для решения данной задачи необходимо выполнить последовательность действий, аналогично представленным на рис. 8, для своего варианта исходных данных. Результатом выполнения практической работы является диаграмма объектов, составление которой следует подробно объяснить.

Практическое занятие № 08. РАЗРАБОТКА КРИТИЧЕСКИХ СЕРВИСОВ ДЛЯ РАЗВЕРТЫВАНИЯ MSA. UML-ДИАГРАММА ПАКЕТОВ ДАННЫХ. ERD

ПОСТРОЕНИЕ UML-ДИАГРАММЫ ПАКЕТОВ ДАННЫХ

Диаграмма пакетов данных (Package) определяет «шаблон объекта и рамки для других шаблонов» ("template and bound to other templates") [13, 242].

При построении UML-диаграммы пакетов данных используются следующие основные элементы (рис. 9), представленные в [13–14; 17].

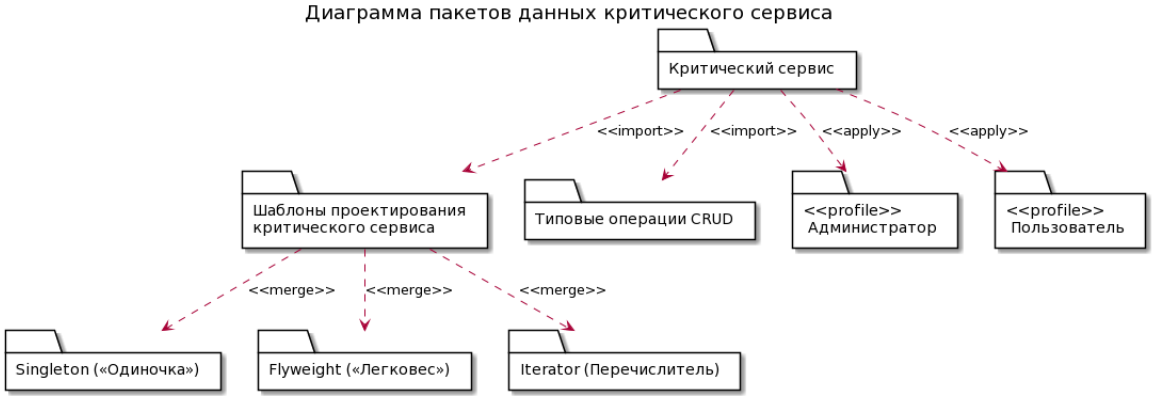

Рис. 9. Основные элементы UML-диаграммы пакетов данных

На рис. 9 показаны следующие основные блоки и типовые взаимосвязи диаграммы пакетов данных [13, 241–252]:

- блоки:

o **package** (пакет данных) – представляет собой обособленное пространство имен (namespace), включающее «собственные» (импортированные) и «расширенные» (объединяемые из других пакетов данных) элементы;

o **element** (элемент) – составная часть пакета данных;

- взаимосвязи:

o <<**merge**>> (объединение) – расширение состава элементов, включаемых в пакет данных: в виде косвенного доступа к этим данным;

o <<**import**>> (импортирование) – подключение других пакетов данных: в виде включения соответствующих программных модулей в состав данного проекта;

o <<**apply**>> (применение) – использование профиля данных для пакета данных (построение диаграммы профилей рассмотрено в практической работе № 3).

ПОСТРОЕНИЕ IDEF1X-ДИАГРАММЫ «СУЩНОСТЬ–СВЯЗЬ» (ERD)

Диаграмма IDEF1X (View Diagram; диаграмма *представления*) [31–34] определена как «графическое отображение базовых семантических конструкций для их *представления*» ("a graphic representation of the underlying semantics of a *view*"); под *представлением* понимается «коллекция *сущностей* и назначенных атрибутов (областей определения), собранных для определенных целей» ("a collection of *entities* and assigned attributes (domains) assembled for some purpose") [32, 6]. Диаграмму IDEF1X часто называют диаграммой «сущность-связь» или ERD (Entity-Relationship Diagram) [31; 33–34].

Основными элементами IDEF1X-диаграммы «сущность–связь» (ERD) являются [31–33; 17]:

- **entity** (сущность) – представление набора реальных или абстрактных предметов, которые обладают общими характеристиками и для которых могут быть созданы взаимосвязи (например, каждая из девяти таблиц, представленных на рис. 10);
- **attribute** (атрибут) представляет собой разновидность свойства, ассоциированного с набором реальных или абстрактных предметов (например, атрибутами являются VR_id, VR_name, VR_brand и другие элементы данных таблиц, представленных на рис. 10);
- **key** (ключ) указывает на особенность атрибута: уникальность (primary key: первичный ключ) или связанность (foreign key: внешний ключ);
- **relationship** (соотношение) определяет тип взаимосвязи сущностей: «один-ко-многим» ("one-many": все взаимосвязи на рис. 10, кроме двух связей с таблицей VR_Resolution_to_Screen), «многие-ко-многим» ("many-many": соотношение между таблицами VR_Resolution и VR_Screen, представленное на рис. 10 в виде двух связей с таблицей VR_Resolution_to_Screen), «один-к-одному» ("one-one": в этом случае одна таблица является частью другой).

Пример диаграммы, учитывающий современное представление ERD [33–34], представлен на рис. 10 (диаграмма выполнена с помощью онлайн-инструмента PlantUML [17–18]).

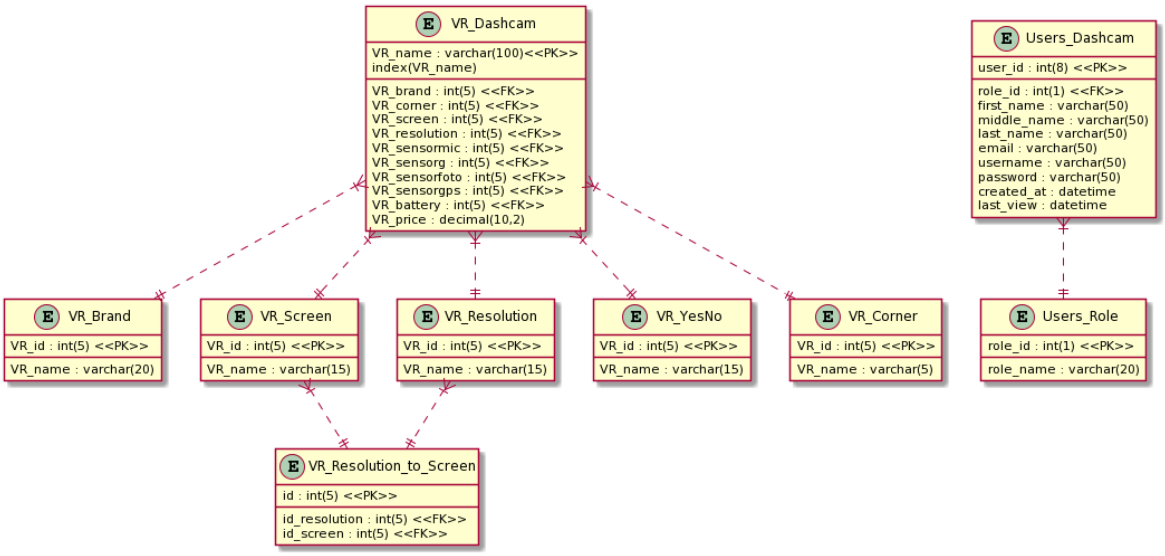

Рис. 10. IDEF1X-диаграмма базы данных о видеорегистраторах (здесь «PK» (Primary Key) – первичный ключ; «FK» (Foreigh Key) – внешний ключ)

ЗАДАЧА 08

Выполнить построение модели данных в виде диаграммы пакетов данных и диаграммы «сущность-связь», используя данные предметной области, представленные в табл. 4.

ПОЯСНЕНИЯ ПО РЕШЕНИЮ ЗАДАЧИ 08

Для варианта № 00, с помощью диаграммы пакетов данных, определены типовые взаимодействия, представленные на рис. 9.

Создание таблиц о видеорегистраторах, с помощью диаграммы «сущность–связь», учитывает следующие соотношения (рис. 10):

- «один-ко-многим» ("one-many") в виде отдельной индексированной таблицы:
  - ролей (role) пользователя;
  - производителей (brand) видеорегистраторов;
  - вариантов диагонали экрана (screen);
  - вариантов разрешения (resolution);
  - вариантов наличия/отсутствия заданных опций (yesno);
  - углов обзора (corner);
- «многие-ко-многим» ("many-many") в виде отдельной таблицы объединяющей свойства:
  - вариантов диагонали экрана и разрешения экрана.

Практическое занятие № 09. РАЗРАБОТКА КРИТИЧЕСКИХ СЕРВИСОВ ДЛЯ РАЗВЕРТЫВАНИЯ MSA. IDEF0-ДИАГРАММА

ПОСТРОЕНИЕ SADT-ДИАГРАММ В НОТАЦИИ IDEF0

Методология моделирования бизнес-процессов – SADT (Structural Analysis and Design Technique) состоит в определении требований и функций для последующей разработки системы, а также для анализа функций, осуществляемых системой, и отображения механизмов, посредством которых эти функции выполняются. Подход SADT представлен в стандарте IDEF0, который регламентирует следующие основные элементы [31; 35–37] (рис. 11):

Рис. 11. Основные элементы IDEF0-диаграммы

- элементы данных:

o стрелки входа (входят в левую грань процесса) отображают данные или объекты, изменяемые в ходе выполнения функции;

o стрелки управления (входят в верхнюю грань процесса) соответствуют правилам и ограничениям, согласно которым выполняется функция;

o стрелки выхода (выходят из правой грани процесса) отображают данные или объекты, появляющиеся в результате выполнения функции;

o стрелки механизма (входят в нижнюю грань процесса) соответствуют ресурсам или механизмам, которые необходимы для выполнения функции, но не изменяются во время ее выполнения (внешние сущности по отношению к данному процессу);

- процессы:

o блоки функций (процессы) определяют действия по преобразованию входных данных в выходные, основываясь на предоставляемых ресурсах и

механизмах, руководствуясь при этом инструкциями от элементов управления: для описания процесса необходимо определить не менее одного элемента для каждого из четырех видов данных;

- стрелки вызова процедуры (выходят из нижней грани процесса) отображают связи между разными диаграммами или моделями (рекомендуется не использовать на диаграммах).

Все элементы следует именовать; главная диаграмма в иерархии диаграмм IDEF0 всегда изображает функционирование системы в целом (диаграмма верхнего уровня; Top diagram), остальные диаграммы называются контекстными (контекстная диаграмма; Context diagram) по отношению к родительской диаграмме (и идентифицируются в зависимости от родительской диаграммы) [31; 34–37].

ЗАДАЧА 09

Выполнить описание задачи, условия которой представлены в предыдущей практической работе.

ПОЯСНЕНИЯ ПО РЕШЕНИЮ ЗАДАЧИ 09

Определение элементов процесса «Разработка сервиса для обработки заказов видеорегистраторов» с помощью [18] приведено на рис. 12.

Рис. 12. Описание процесса разработки сервиса для обработки заказов видеорегистраторов

Как видно из рис. 12, процесс «Разработка сервиса для обработки заказов видеорегистраторов» включает 8 взаимосвязей:

- исходные данные процесса: характеристики видеорегистраторов и предоставляемые интерфейсы, которые следует интегрировать в состав создаваемого сервиса;
- элементы управления процесса: параметры и ограничения видеорегистраторов, а также атрибуты QFD (развертывания функций качества);
- механизмы процесса: инструменты проектирования и разработки, а также ролевые объекты (такие как: Веб-сервер, Администратор, Пользователь).
- выходные данные процесса: созданный сервис и результаты автоматизированного тестирования.

Практическое занятие № 10. РАЗРАБОТКА КРИТИЧЕСКИХ СЕРВИСОВ ДЛЯ РАЗВЕРТЫВАНИЯ MSA. UML-ДИАГРАММА КЛАССОВ

ПОСТРОЕНИЕ UML-ДИАГРАММЫ КЛАССОВ

Диаграмма классов (Class) определяет «абстрактный метакласс, для которого конкретные подклассы используются, чтобы классифицировать различные виды значений» "an abstract metaclass whose concrete subclasses are used to classify different kinds of values" [13, 99].

При построении UML-диаграммы классов используются следующие основные элементы (рис. 13), представленные в [13–14; 17].

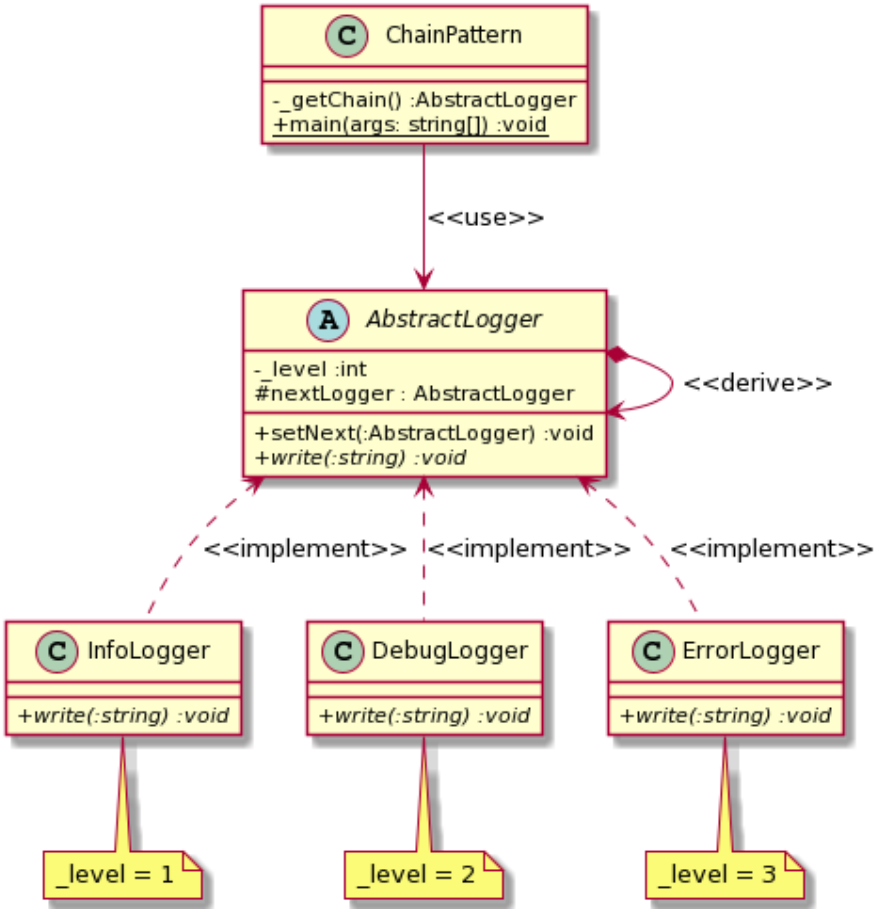

Рис. 13. Основные элементы UML-диаграммы классов

Диаграмма классов включает следующие основные блоки и типовые взаимосвязи диаграммы классов [13, 99–126]:

блоки:

- **class** (класс) – блок, определяющий класс в виде совокупности данных (_level и nextLogger на рис. 13) и методов (main(), write() и setNext() на рис. 13);
- **template** (шаблон, параметризованный класс) определяет типизированный класс, как правило, абстрактный (AbstractLogger на рис. 13);
- **attribute** (атрибут класса) определяет уровни доступа к элементам

класса: «+» – общедоступные (public) данные и методы, «-» – приватные (private) элементы класса, «#» – защищенные (protected) данные и методы класса; спецификация самих данных и параметров методов определяется как для диаграммы объектов (рис. 8);

- взаимосвязи:

o **generalization** (обобщение) определяет взаимосвязь наследования классов (<<inherit>>), при этом особым образом указывается наследование параметризованного класса (<<implement>>: реализация наследуемых интерфейсов);

o **bind** (связывание) определяет степень взаимосвязи данного класса с другими объектами: различают «жесткую» (rigid: указывается в виде составного элемента или взаимосвязи с закрашенным ромбом: например, <<derive>> на рис. 13: типизированная взаимосвязь извлечения данных) и «гибкую» связи (flexible: указывается в виде взаимосвязи без закрашенного ромба: например, <<use>> на рис. 13: типизированная взаимосвязь обращения к данным).

ЗАДАЧА 10

Выполнить построение диаграммы классов для задачи, условия которой представлены в практической работе № 8.

ПОЯСНЕНИЯ ПО РЕШЕНИЮ ЗАДАЧИ 10

Для решения данной задачи необходимо выполнить последовательность действий, аналогично представленным на рис. 13, для своего варианта исходных данных. Результатом выполнения практической работы является диаграмма классов, составление которой следует подробно объяснить.

Практическое занятие № 11. РАЗРАБОТКА КРИТИЧЕСКИХ СЕРВИСОВ ДЛЯ РАЗВЕРТЫВАНИЯ MSA. UML-ДИАГРАММА КОММУНИКАЦИЙ

ПОСТРОЕНИЕ UML-ДИАГРАММЫ КОММУНИКАЦИЙ

Диаграмма коммуникаций (Communication) устанавливает «взаимодействие между «линиями жизни» объектов, в то время как архитектура внутренней структуры объектов и способ передачи сообщений остаются в центре внимания; последовательность сообщений нумеруется на схеме» ("the interaction between Lifelines where the architecture of the internal structure and how this corresponds with the message passing is central; the sequencing of Messages is given through a sequence numbering scheme") [13, 599].

При построении UML-диаграммы коммуникаций используются следующие основные элементы (рис. 14), представленные в [13–14; 17].

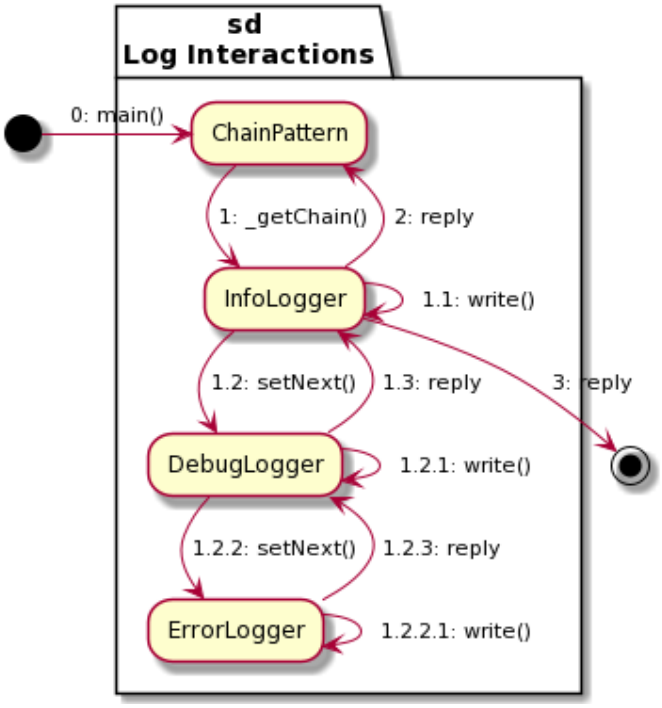

Рис. 14. Основные элементы UML-диаграммы коммуникаций

На рис. 14 показаны следующие основные блоки и типовые взаимосвязи диаграммы коммуникаций [13, 599–601]:

- **frame** (фрейм) – конструкция, объединяющая объекты, обменивающиеся сообщениями в контексте диаграммы последовательности (sd: Sequence diagram) или базовой (типовой) диаграммы взаимодействия (ref: Reference diagram);
- **message** (сообщение) показывает взаимосвязь двух объектов, при этом последовательность выполнения действий определяется благодаря многоуровневой нумерации таких взаимосвязей.

ЗАДАЧА 11

Определить взаимодействие между объектами диаграммы классов, представленной в предыдущей практической работе.

ПОЯСНЕНИЯ ПО РЕШЕНИЮ ЗАДАЧИ 11

Для решения данной задачи необходимо выполнить последовательность действий, аналогично представленным на рис. 14, для своего варианта исходных данных. Результатом выполнения практической работы является диаграмма коммуникаций, составление которой следует подробно объяснить.

ЗАКЛЮЧЕНИЕ

В данном учебном пособии представлены общие указания по выполнению практических работ по дисциплине «Программирование критических сервисов». Предметная область дисциплины в настоящее время (2021 г.) является неустоявшейся, поэтому понятие «критический сервис» определено в контексте задач программирования следующим образом (табл. 1): «обособленный набор программных модулей, содержащий информацию, доступ к которой должен быть строго ограничен».

11 практических работ, в результате выполнения которых создаются 12 различных UML- и SADT-диаграмм, ориентированы на раскрытие предметной области практикума, в контексте постановки и решения актуальных задач, решаемых IT-специалистами. Развертывание микросервисной архитектуры является одной из таких актуальных задач, связанной с проектированием, разработкой и сопровождением сервисов, содержащих важную для пользователя информацию, доступ к которой следует строго ограничить.

В практикуме не представлены 4 диаграммы стандарта UML 2 (Composite Structure, Component, Interaction Overview, Timing), как не раскрыт широкий круг задач, напрямую или косвенно связанных с программированием критических сервисов. Вместе с тем, в данном учебном пособии изложены конкретные задания и типовые решения, позволяющие раскрыть предметную область дисциплины «Программирование критических сервисов» с тем, чтобы применить полученные навыки и умения для смежных дисциплин и соответствующих практических задач.

СПИСОК ЛИТЕРАТУРЫ

1. ГОСТ 33707–2016 (ISO/IEC 2382:2015). Информационные технологии. Словарь [Текст]. – М.: Стандартинформ, 2016. – 206 с.
2. ISO/IEC/IEEE 24765:2017. System and Software Engineering – Vocabulary [Text]. – Jeneva: ISO, 2017. – 522 p.
3. Справочник по Telegram Bot API : [Электронный ресурс]. – URL: https://tlgrm.ru/docs/bots/api
4. Simple Whatsapp Automation Using Python3 and Selenium : [Electronic Resource]. – 2019. – URL: https://medium.com/analytics-vidhya/simple-whatsapp-automation-using-python3-and-selenium-77dad606284b
5. Параскевов А. В. Перспективы и особенности разработки чат-ботов : [Текст] / А. В. Параскевов, А. А. Каденцева, С. И. Мороз // Науч. журнал КубГАУ. – 2017. № 130. с. 395-404.
6. Fowler, M. Microservices [Electronic Resource]. – 2014. – URL: https://martinfowler.com/articles/microservices.html
7. Northwood, C. The Full Stack Developer: Your Essential Guide to the Everyday Skills Expected of a Modern Full Stack Web Developer [Text] / C. Northwood. 1st ed. UK, Manchester: APress, 2018. – 365 p.
8. Pattern: Microservice Architecture [Electronic Resource]. – 2018. – URL: https://microservices.io/patterns/microservices.html
9. Плас, Дж. Вандер. Python для сложных задач: наука о данных и машинное обучение : [Электронный ресурс] / Дж. Вандер Плас. – Санкт-Петербург : Питер, 2018. – 576 с. : ил. – URL: http://ibooks.ru/reading.php?productid=356721. – ISBN 978-5-496-03068-7 : Б. ц.
10. Северенс, Ч. Введение в программирование на Python : [Электронный ресурс]: учебное пособие / Ч. Северенс. – 2-е изд. – Москва : ИНТУИТ, 2016. 231 с. – URL: https://e.lanbook.com/book/100703. - Б. ц.
11. Practical Microservices Development Patterns: CRUD Vs. CQRS : [Electronic Resource]. – 2020. – URL: https://hackernoon.com/practical-microservices-development-patterns-crud-vs-cqrs-h6m3y5y
12. Eventuate example microservices applications : [Electronic Resource]. – 2021. – URL: https://eventuate.io/exampleapps.html
13. About the Unified Modeling Language Specification Version 2.5.1: [Electronic Resource]. – 2017. – 754 p. – URL: https://www.omg.org/spec/UML/2.5.1
14. Fowler, M. UML Distilled: A Brief Guide to the Standard Object Modeling Language: [Text] / M. Fowler. – 3rd Ed. Boston: Addison-Wesley, 2003. – 178 p.
15. Gamma, E. Design Patterns. Elements of Reusable Object-Oriented Software : [Text] / E. Gamma, R. Helm, R. Johnson, J. Vlissides // Forework by Grady Booch. – 37th printing. – Boston: Adisson-Wesley, 2009. 417 p.
16. Фримен, Э. Паттерны проектирования : [Электронный ресурс] / Э. Фримен [и др.]. – Санкт-Петербург : Питер, 2017. – 656 с. : ил. – URL: http://ibooks.ru/reading.php?productid=354827. – ISBN 978-5-496-00782-5 : Б. ц.

17. Drawing UML with PlantUML. Language Reference Guide [Electronic Rersource], 2021. 416 p. –URL: http://plantuml.com/guide
18. Онлайн-редактор диаграмм PlantText [Электронный ресурс]. – URL: https://www.planttext.com/, https://www.plantuml.com/plantuml
19. PyCharm. The Python IDE for Professional Developers : [Electronic Resource]. – URL: https://www.jetbrains.com/pycharm/
20. Driver requirements: Documentation for Selenium [Electronic Resource]. – URL: https://www.selenium.dev/documentation/en/webdriver/driver_requirements/
21. Spyne: PRC that doesn't break your back [Electronic Resource]. – 2021. – URL: http://spyne.io/
22. SOAP и REST сервисы с помощью Python-библиотеки Spyne [Electronic Resource]. – 2017. – URL: https://habr.com/ru/post/334290/
23. The Top 10 Python Frameworks For Web Development [Electronic Resource]. – 2020. – URL: https://www.activestate.com/blog/the-top-10-python-frameworks-for-web-development/
24. Regexp101: build, test and debug regexp [Electronic Resource]. – 2021. – URL: https://regex101.com/
25. PuTTY: a free SSH and telnet client for Windows [Electronic Resource]. – 2021. – URL: https://www.putty.org/
26. UML Use Case Include [Electronic Resource]. – 2020. – https://www.uml-diagrams.org/use-case-include.html
27. Walderhaug, S. Experiences from Model-Driven Development of Homecare Services: UML Profiles and Domain Models [Text] / S. Walderhaug, , E. Stav, M. Mikalsen // Chaudron M.R.V. (eds) Models in Software Engineering. MODELS 2008. Lecture Notes in Computer Science, vol. 5421. Berlin: Springer. – URL: https://www.researchgate.net/publication/221223886_Experiences_from_Model-Driven_Development_of_Homecare_Services_UML_Profiles_and_Domain_Models
28. Параничев, А. В. Опыт проектирования учебного программного продукта с помощью инструментария PlantUML [Текст] / А. В. Параничев // Системы проектирования, технологической подготовки производства и управления этапами жизненного цикла промышленного продукта (CAD/CAM/PDM – 2017) : Труды XVII международной научно-практической конференции, Москва, 12–14 декабря 2017 года / Под ред. А.В. Толока, Институт проблем упр. им. В. А. Трапезникова. – Москва: Институт проблем управления им. В.А. Трапезникова РАН, 2017. – С. 224-227. – URL: https://www.elibrary.ru/download/elibrary_32807215_95747964.pdf
29. Шпаргалка по шаблонам проектирования [Электронный ресурс]. – 2014. – URL: https://habr.com/ru/post/210288/
30. Паттерны/шаблоны проектирования [Электронный ресурс]. – 2021. – URL: https://refactoring.guru/ru/design-patterns

31. Barkmeyer, E. SIMA Reference Architecture Part I: Activity Models [Text] / E. Barkmeyer, N. Christopher, S. Feng [etc.] // NIST Interagency / Internal Report (NISTIR). – US, Gaithersburg: National Institute for Standards and Technology, 1996. – 76 p. – URL: https://doi.org/10.6028/NIST.IR.5939
32. FIPS Publication 184. Integration Definition for Information Modeling (IDEF1X) [Text]. – US, Gaithersburg: National Institute for Standards and Technology, 1993. – 135 p.
33. ISO/IEC/IEEE 31320-2:2012. Information technology – Modeling Languages – Part 2: Syntax and Semantics for IDEF1X97 (IDEFobject) [Text]. – Jeneva: ISO, 2012. – 305 p.
34. Коломец, Н. В. О методах и средствах проектирования программного обеспечения (обзор и примеры) [Текст] / Н. В. Коломец // Ростовский научный журнал. – 2017. – № 4. – С. 249–265.
35. FIPS Publication 183. Integration Definition for Function Modeling (IDEF0) [Text]. – US, Gaithersburg: National Institute for Standards and Technology, 1993. – 116 p.
36. ISO/IEC/IEEE 31320-1:2012. Information technology – Modeling Languages – Part 1: Syntax and Semantics for IDEF0 [Text]. – Jeneva: ISO, 2012. – 106 p.
37. Jung, K. Mapping Strategic Goals and Operational Performance Metrics for Smart Manufacturing Systems [Text] / K. Jung, K. Lyons, S. Leong [etc.] // Procedia Computer Science. – 2015. – Vol. 44. – P. 184–193. URL: https://doi.org/10.1016/j.procs.2015.03.051