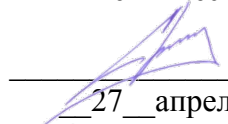


**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук
Образовательная программа бакалавриата «Программная инженерия»

СОГЛАСОВАНО

Научный руководитель, доцент
департамента программной
инженерии ФКН, кандидат
технических наук

 С.А. Виденин
__27__ апреля __2025__ г.

УТВЕРЖДАЮ

Академический руководитель
образовательной программы
«Программная инженерия»
профессор департамента программной
инженерии, кандидат технических наук

____ Н.А. Павлочев
____ 2025 г.


**ИНФОРМАЦИОННО-АНАЛИТИЧЕСКАЯ СИСТЕМА УПРАВЛЕНИЯ
ИНЦИДЕНТАМИ В ОБЛАСТИ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ
(СЕРВЕРНАЯ ЧАСТЬ: УПРАВЛЕНИЕ УЧАСТНИКАМИ СИСТЕМЫ,
УВЕДОМЛЕНИЯМИ И ДЕЖУРСТВАМИ)**

Текст программы

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.05.05-01 12 01-1-ЛУ

Исполнитель
студент группы БПИ214

____  ____ / Е.К. Фортов/
__27__ апреля __2025__ г.

Подп. и дата	
Инв. № дубл.	
Взам. Инв. №	
Подп. и дата	
Инв. № подл.	

УТВЕРЖДЕН
RU.17701729.05.05-01 12 01-1-ЛУ

**ИНФОРМАЦИОННО-АНАЛИТИЧЕСКАЯ СИСТЕМА УПРАВЛЕНИЯ
ИНЦИДЕНТАМИ В ОБЛАСТИ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ (СЕРВЕРНАЯ
ЧАСТЬ: УПРАВЛЕНИЕ УЧАСТНИКАМИ СИСТЕМЫ, УВЕДОМЛЕНИЯМИ
И ДЕЖУРСТВАМИ)**

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл	

Текст программы

RU.17701729.05.05-01 12 01-1-ЛУ

Листов 16

Аннотация

В документе «Текст программы» приведены ссылки на классы, написанные при реализации программы «Информационно-аналитическая система управления инцидентами в области обеспечения безопасности. Серверная часть: управление участниками системы, уведомлениями и дежурствами», разработанной в рамках ВКР. Программа написана на языке программирования Java. Данный документ имеет разделы: «Глоссарий», «Текст программы».

Настоящий документ разработан в соответствии с требованиями:

- 1.ГОСТ 19.101-77 Виды программ и программных документов [1];
- 2.ГОСТ 19.103-77 Обозначения программ и программных документов [3];
- 3.ГОСТ 19.104-78 Основные надписи [4];
- 4.ГОСТ 19.105-78 Общие требования к программным документам [5];
- 5.ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом [6];

Изменения к данному документу оформляются согласно ГОСТ 19.603-78 [8], ГОСТ 19.604-78 [9].

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.05-01 12				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Содержание

Аннотация.....	3
ГЛОССАРИЙ.....	5
1. ВВЕДЕНИЕ.....	6
2. ОПИСАНИЕ ФАЙЛОВ С ИСХОДНОМ КОДОМ ПРОГРАММЫ.....	8
2.1. Модуль duties-service.....	8
2.2. Модуль notifications-service.....	9
2.3. Модуль profiles-service.....	11
2.4. Модуль Common (библиотека).....	12
3. ОПИСАНИЕ СКРИПТА РАЗВОРАЧИВАНИЯ СЕРВИСОВ В ОБЛАКЕ.....	13
СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ.....	14
ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ.....	16

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.05-01 12				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

ГЛОССАРИЙ

1. ТЗ — техническое задание
2. БД — база данных
3. Пользователь – человек, который использует разрабатываемую систему
4. СУБД – система управления базами данных.
5. Репозиторий — директория со всеми файлами проекта.
6. API (Application Programming Interface) – описание способов (набор классов, методов и т. п.), которыми одна компьютерная программа (в данном случае, клиентская часть приложения) может взаимодействовать с другой (в данном случае, с сервером).
7. Логгер — сущность или фреймворк (в зависимости от контекста), отвечающая за запись лог-файлов (сопроводительная информация о работе системе).
8. Юнит-тест — тест, написанный на определенный метод программы.
9. Контейнер – это модуль, в котором запускается одно приложение. Контейнеры занимают меньше памяти, используют небольшое количество ресурсов и почти не зависят от операционной системы кластера.
10. Docker — система для контейнеризации приложения.
11. Docker-compose — система для оркестрации (управления) Docker-контейнеров.
12. RESTful API — то же, что REST API.
13. Микросервис — это небольшой атомарный самостоятельный сервис, который отвечает за одну функциональную роль в системе&
14. Кастомизированный — особый, сделанный, не копируя аналоги
15. Контроллер — класс, являющийся входной точкой HTTP-запросов в приложение.
16. Бизнес-логика приложения — это основная логика работы приложения, включающая алгоритмы по обработке поступивших данных.
17. Развертывание приложения – запуск приложения в какой-либо среде.
18. Эндпоинт API — это это конкретный адрес (URL), по которому клиент (например, веб- приложение, мобильное приложение или другой сервис) может взаимодействовать с сервером, использующим API.
19. Маппер — класс, преобразующий сущность одного типа в сущность другого типа.
20. Скрипт — файл с кодом, как правило, настраивающим что-то. Например, сборочный скрипт — это файл с инструкциями по разворачиванию сервиса.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.05-01 12				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

1. ВВЕДЕНИЕ

В данном документе представлено описание структуры репозитория, в котором находится исходный код программы «Информационно-аналитическая система управления инцидентами в области обеспечения безопасности. Серверная часть: управление участниками системы, уведомлениями и дежурствами».

Репозиторий программы находится по ссылке:
<https://github.com/FortovEgor/Diploma>.

Сервис представляет из себя многомодульный проект на java, собирающийся системой сборки Maven. В корневой директории репозитория присутствует родительский pom.xml - файл настроек сборки всей системы. Код каждого сервиса располагается в отдельной директории, содержащей свой pom.xml файл — файл настроек сборки каждого сервиса. Директория каждого сервиса имеет следующий состав:

- src
 - main
 - java — директория со всеми исходными файлами модуля
 - resources — директория со всеми ресурсами модуля: настроечным(-ыми) файлом(-ами) проекта, файлом для инициализации базы данных, а также конфигураций логгера logback
 - test — директория с юнит-тестами для модуля
- target — директория с временными файлами, создающимися во время сборки проекта, а также .jar-файлами — результатами сборки проекта
- Dockerfile — скрипт для запуска сервиса в Docker
- pom.xml — файл настроек сборки через Maven

Помимо директорий с модулями-сервисами в проекте присутствуют следующие элементы:

- API_DOCS — директория со спецификациями REST API каждого сервиса
- Common — директория с библиотекой с общими файлами, которые используются несколькими модулями

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.05-01 12				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

- logs — папка с логами (она создается при запуске сервисов, в нее пишутся логи программы)
- .gitignore — файл системы контроля версий, который описывает шаблоны файлов и директорий, которые не должны индексироваться
- .git — главный файл системы контроля версий
- checkstyle.xml — файл, описывающий правила проверки кодстайла проекта
- docker-compose.yml — скрипт для разворачивания сервисов в облаке (в частности, в Docker Compose)
- LICENSE — файл с лицензией MIT, предотвращает использование проекта без указания авторства
- lombok.config — файл с конфигурацией библиотеки lombok, предоставляющей аннотации
- README.md — файл с markdown разметкой для описания репозитория
- start.sh — скрипт для сборки и запуска сервисов системы
- suppressions.xml — это файл конфигурации, используемый в различных программных приложениях и системах для управления процессом подавления или фильтрации определенных данных или событий
- postman — директория с коллекцией end-to-end тестов, запускаемых с помощью утилиты postman

Стоит заметить, что код писался в декларативном стиле и трудностей с его пониманием при чтении возникнуть не должно.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.05-01 12				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

2. ОПИСАНИЕ ФАЙЛОВ С ИСХОДНОМ КОДОМ ПРОГРАММЫ

2.1. Модуль `duties-service`

В данном документе представлено описание структуры репозитория, в котором находится исходный код программы «Информационно-аналитическая система управления инцидентами в области обеспечения безопасности. Серверная часть: управление участниками системы, уведомлениями и дежурствами».

Модуль отвечает за управление дежурствами в системе.

Весь код модуля располагается в пакете `fortov.egor.diploma` и его подпакетах:

- `config` — конфигурации для Spring Boot
- `dao` — сущности для работы с базой данных
- `dto` — сущности для взаимодействия с сервисом по API
- `exception` — сущности кастомизированных исключений
- `storage` — интерфейсы взаимодействия с данными в хранилище (базе данных)

Точкой входа в приложение является класс `DiplomaDuties`, содержащий метод `main` и передающий управление объекту класса `SpringBootApplication`. Класс `DiplomaDuties` аннотирован соответствующей аннотацией.

Для непосредственного взаимодействия с сервисом по RESTful API используется класс-контроллер `DutyController`. Каждый его метод соответствует определенному эндпоинту.

Бизнес-логика сервиса расположена в классе `DutyService`. `DutyController` делегирует ему обработку запроса.

Для преобразования сущностей в программе используется интерфейс `DutyMapper`. Он аннотирован аннотацией `@Mapper` из библиотеки `mapstruct`, которая генерирует имплементацию методов-преобразователей.

Центральная сущность модуля представлена классом `Duty`.

Интерфейс общения с базой данных — `DutyStorage` в подпакете `storage`.

Сама реализация методов для взаимодействия с базой данных представлена в классе `DutyStorageDb`.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.05-01 12				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Сущности, необходимые для коммуникации по API представлены классами подпакета dto:

- CreateDutyRequest — класс-сущность, который представляет собой тело запроса на создание дежурства.
- DutyDto — класс-сущность, которая возвращается при запросе информации о дежурстве.
- UpdateDutyRequest — класс-сущность, которая представляет собой тело запроса на обновление информации о дежурстве.
- UserDutyDto — класс-сущность, которая возвращается при запросе информации о дежурстве определенного пользователя.

Включение аудита, позволяющего отслеживать изменения в сущностях JPA, сделано в классе-конфигурации JpaConfig подпакета config.

Юнит-тестирование сервиса реализовано в классе DutyControllerTest.

2.2. Модуль notifications-service

Модуль отвечает за управление уведомлениями, а также за их отправку по различным каналам связи, включающим sms, почту и телефонный вызов.

Весь код модуля располагается в пакете fortov.egor.diploma и его подпакетах:

- dao — сущности для работы с базой данных
- dto — сущности для взаимодействия с сервисом по API
- sendingServices — классы-отправители уведомлений, а также необходимые классы-шаблоны, интерфейсы и класс-менеджер
- storage — интерфейсы взаимодействия с данными в хранилище (базе данных)

Точкой входа в приложение является класс DiplomaNotifications, содержащий метод main и передающий управление объекту класса SpringBootApplication. Класс DiplomaNotifications аннотирован соответствующей аннотацией.

Для непосредственного взаимодействия с сервисом по RESTful API используется класс-контроллер NotificationsController. Каждый его метод соответствует определенному эндпоинту.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.05-01 12				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Базовая бизнес-логика сервиса расположена в классе `NotificationsService`. `NotificationsController` делегирует ему обработку запроса.

Для преобразования сущностей в программе используется интерфейс `NotificationsMapper`. Он аннотирован аннотацией `@Mapper` из библиотеки `mapstruct`, которая генерирует имплементацию методов-преобразователей.

Центральная сущность модуля представлена классом `Notification`.

Интерфейс общения с базой данных — `NotificationStorage` в подпакете `storage`.

Сама реализация методов для взаимодействия с базой данных представлена в классе `NotificationStorageDb`.

За главную бизнес-логику, а именно за обработку и отправку уведомлений, отвечают следующие классы:

- `NotificationsManager` — с помощью этого класса происходит регистрация уведомления, редактирование, делегирование его отправки и удаление.
- `SendingManager` — этот класс инкапсулирует логику по отправке уведомления: он отправляет уведомление, используя все имеющиеся сервисы-отправители (см. далее)
- `SendingService` — интерфейс для отправки уведомления.
- `SendingServiceTemplate` — шаблон для создания класса-отправителя, содержит базовую реализацию отправки, реализацию-заглушку.
- `EmailService` — класс, отвечающий за отправку уведомления по почте.
- `SmsService` — класс, отвечающий за отправку уведомления по смс.

Сущности, необходимые для коммуникации по API представлены классами подпакета `dto`:

- `CreateNotificationRequest` — класс-сущность, который представляет собой тело запроса на создание уведомления.
- `UpdateNotificationRequest` — класс-сущность, который представляет собой тело запроса на изменение информации об уведомлении.

Юнит-тестирование сервиса реализовано в классе `NotificationControllerTest`.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.05-01 12				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

2.3. Модуль profiles-service

Модуль отвечает за управление профилями пользователей в системе.

Весь код модуля располагается в пакете fortov.egor.diploma и его подпакетах:

- user — классы, непосредственно относящиеся к логике сервиса
 - dto — классы-сущности для взаимодействия с сервисом по API
- loggingFilter — классы для логирования запросов и их обработки
- validation — классы с проприетарными аннотациями и классами по их настройке
- exception — сущности кастомизированных исключений

Точкой входа в приложение является класс DiplomaProfiles, содержащий метод main и передающий управление объекту класса SpringBootApplication. Класс DiplomaProfiles аннотирован соответствующей аннотацией.

Для непосредственного взаимодействия с сервисом по RESTful API используется класс-контроллер UserController. Каждый его метод соответствует определенному эндпоинту.

Бизнес-логика сервиса расположена в классе UserService. UserController делегирует ему обработку запроса.

Для преобразования сущностей в программе используется интерфейс UserMapper. Он аннотирован аннотацией @Mapper из библиотеки mapstruct, которая генерирует имплементацию методов-преобразователей.

Центральная сущность модуля представлена классом User.

Для взаимодействия с базой данных используется интерфейс UserRepository.

Сущности, необходимые для коммуникации по API представлены классами подпакета dto:

- NewUserRequest — класс-сущность, который представляет собой тело запроса на создание нового пользователя.
- UserDto — класс-сущность, которая возвращается при запросе информации о пользователе.
- UserDtoPartial — класс-сущность, которая возвращается при запросе частичной информации о пользователе.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.05-01 12				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

За логирование отвечает класс `LoggingFilter` из одноименного пакета.

Проприетарные классы-исключения включают `NotValidException` и класс-обработчик исключений `ControllerExceptionHandler`.

Юнит-тестирование сервиса реализовано в классе `UserControllerTest`.

2.4. Модуль `Common` (библиотека)

Данный модуль содержит общие классы, используемые несколькими модулями сразу. В частности, это классы:

- `NotFoundException` — проприетарный класс-исключение, которое выбрасывается, когда нужный объект не найден.
- `DBException` — проприетарный класс-исключение для любого исключения, возникшего при работе с БД.
- `ConflictException` - проприетарный класс-исключение, которое выбрасывается, когда нарушены условия ограничений в таблице в БД.
- `UserFullInfoDto` — класс-сущность, отвечающий за полную информацию о пользователе.
- `UserClient` — класс-клиент для выполнения http запросов в сервис профилей пользователей.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.05-01 12				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

3. ОПИСАНИЕ СКРИПТА РАЗВОРАЧИВАНИЯ СЕРВИСОВ В ОБЛАКЕ

Данный скрипт поднимает 6 контейнеров в облаке: 3 контейнера с сервисами — на каждый сервис по одному контейнеру и 3 контейнера с СУБД PostgreSQL — для каждого сервиса свой инстанс БД. Также скрипт настраивает проброс портов, передачу необходимых переменных окружения, настройку healthcheck-ов для проверки доступности сервисов и возможность автоматического перезапуска сервиса в случае падения.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.05-01 12				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. ГОСТ 19.101-77 Виды программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2. ГОСТ 19.102-77 Стадии разработки. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.103-77 Обозначения программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.104-78 Основные надписи. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.105-78 Общие требования к программным документам. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. ГОСТ 19.603-78 Общие правила внесения изменений. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
10. ГОСТ 19.404-79 Программа и методика испытаний. Требования к содержанию и оформлений. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
11. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
12. ГОСТ 19.505-79 Руководство оператора. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
13. ГОСТ 19.401-78 Текст программы. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.ГОСТ 19.301-79 Программа и методика испытаний. Требования к

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.05-01 12				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

содержанию и оформлению // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.05-01 12				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

[illegible]