**МИНИСТЕРСТВО ОБРАЗОВАНИЯ, КУЛЬТУРЫ И ИССЛЕДОВАНИЙ**

**МОЛДАВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ МАТЕМАТИКИ И ИНФОРМАТИКИ**

**ДЕПАРТАМЕНТ ИНФОРМАТИКИ**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

**ПО ПРЕДМЕТУ:**

**«FRAMEWORKS»**

***Выполнила:*** Спыну Екатерина,

группа IA2102 (Дизайн)

***Проверяющий: Нартя Никита***

**Кишинев, 2023**

# **ВВЕДЕНИЕ**

Данная лабораторная работа знакомит студентов с продвинутой функциональностью фреймворка Symfony.

# **ЦЕЛЬ РАБОТЫ**

Данная лабораторная работа показывает продвинутый процесс разработки приложения, который включает в себя сервисы, безопасность, тестирование.

# **ВЫПОЛНЕНИЕ:**

## **СЕРВИСЫ: СОЗДАНИЕ TASKSERVICE**

1. Создаем сущность Task с помощью Symfony CLI:

**symfony console make:entity Task**

Далее, отвечаем на вопросы, чтобы определить поля сущности

1. Создаем миграцию для нашей базы данных:

**symfony console make:migration**

1. Применяем миграцию, чтобы создать таблицу в базе данных:

**symfony console doctrine:migrations:migrate**

1. Создаем сам сервис TaskService с помощью Symfony CLI:

**symfony console make:service TaskService**

1. В классе TaskService, реализуем методы для создания, редактирования, удаления и получения задач.

**Для создания задачи:**

**public function createTask(Task $task): void**

**{**

**$this->entityManager->persist($task);**

**$this->entityManager->flush();**

**}**

**Для обновления задачи:**

**public function updateTask(Task $task): void**

**{**

**$this->entityManager->flush();**

**}**

**Для удаления задачи:**

**public function deleteTask(Task $task): void**

**{**

**$this->entityManager->remove($task);**

**$this->entityManager->flush();**

**}**

# **БЕЗОПАСНОСТЬ: НАСТРОЙКА АУТЕНТИФИКАЦИИ И АВТОРИЗАЦИИ**

1. Устанавливаем и настраиваем пакет symfony/security-bundle с помощью Composer:

**composer require symfony/security-bundle**

1. Создаем сущность User с помощью Symfony CLI:

**symfony console make:user**

1. В открывшемся окне, отвечаем на вопросы, чтобы определить поля пользователя (логин и пароль).
2. Создаем контроллер и представление для страницы входа:

**symfony console make:controller SecurityController**

1. Настраиваем маршруты и контроль доступа к страницам входа и выхода в **config/routes.yaml.**

6) В контроллерах и методах, где выполняется создание задачи, добавляем проверку авторизации.

# **ТЕСТИРОВАНИЕ**

1. Создаем тестовый класс для TaskService с помощью Symfony CLI:

**symfony console make:unit-test TaskService**

1. Вносим необходимые тесты для функциональности создания, редактирования, удаления и получения задач.
2. После тестинга, запускаем тесты PHPUnit:

**php bin/phpunit**

# **ОТВЕТЫ НА ВОПРОСЫ:**

1. **Что представляет собой `Service Layer` (Слой сервисов) в архитектуре Model-View-Controller (MVC) веб-приложений, и какую роль он играет в разделении бизнес-логики?**

Service Layer (Слой сервисов) в архитектуре Model-View-Controller (MVC) веб-приложений представляет собой слой, который отвечает за обработку бизнес-логики приложения. Его цель - разделить бизнес-логику от представления (View) и уровня доступа к данным (Model). Слой сервисов содержит сервисы, которые предоставляют высокоуровневый интерфейс для выполнения операций приложения, таких как создание, редактирование, удаление и получение данных. Сервисы инкапсулируют логику, что упрощает поддержку, улучшает тестируемость и соблюдение принципов SOLID.

1. **Объясните понятия аутентификации и авторизации в веб-разработке. Как они связаны и почему они важны для безопасности приложений?**

Аутентификация - это процесс проверки подлинности пользователя. Это процесс, при котором приложение удостоверяет, что пользователь действительно тот, за кого себя выдаёт, чаще всего путем проверки логина и пароля.

Авторизация — это процесс установления того, какие действия или ресурсы пользователь имеет право выполнять или получать доступ. Она определяет, какие ресурсы доступны для конкретного пользователя, и основана на его ролях или правах.

Аутентификация и авторизация важны для безопасности приложений, так как они позволяют предотвратить несанкционированный доступ к данным и функциональности приложения, а также защищают конфиденциальность пользовательских данных.

1. **Как, в выбранном Вами фреймворке, можно реализовать авторизацию пользователей и ограничение доступа к определенным маршрутам или действиям на основе их ролей?**

В Symfony, авторизацию пользователей и ограничение доступа к маршрутам или действиям можно реализовать с использованием Symfony Security Bundle. Для этого необходимо:

1. Настроить аутентификацию, определив, как пользователи будут аутентифицироваться (например, через логин и пароль).
2. Создать систему ролей и ролевой иерархии для определения прав доступа пользователей.
3. Применить аннотации @IsGranted к методам контроллеров или использовать конфигурацию маршрутов для ограничения доступа на основе ролей.
4. **В чем заключается разница между юнит-тестированием, интеграционным тестированием?**

**Юнит-тестирование** — это тип тестирования, который проверяет отдельные компоненты (или "юниты") приложения на корректное выполнение их функций. Он изолирует компонент, который тестируется, от остальных частей приложения. Цель - проверить, что каждый компонент работает правильно в изоляции.

**Интеграционное тестирование** — это тип тестирования, который проверяет взаимодействие между различными компонентами или модулями приложения. Он оценивает, как эти компоненты взаимодействуют друг с другом и проверяет, что они работают вместе правильно. Цель - проверить, что компоненты интегрируются правильно и взаимодействуют корректно.

# **БИБЛИОГРАФИЯ**

<https://symfony.com/doc/6.2/setup.html>

<https://moodle.usm.md/course/view.php?id=6440>

<https://testengineer.ru/unit-testy-vs-integracionnye-testy/>

<https://en.wikipedia.org/wiki/Service_layer#:~:text=In%20intelligent%20networks%20(IN)%20and,at%20a%20higher%20application%20layer>.

**Ссылка на репозиторий** -

https://github.com/KaterinaSpinu/Frameworks\_lab2.git