|  |
| --- |
|  |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **"МИРЭА - Российский технологический университет"**  **РТУ МИРЭА** |
| Институт информационных технологий (ИТ) |
| Кафедра инструментального и прикладного программного обеспечения (ИППО) |

|  |  |
| --- | --- |
| **ОТЧЕТ**  **ПО ЛАБОРАТОРНОЙ РАБОТЕ №7** | |
| **по дисциплине** |  |
| **«Разработка серверных частей интернет ресурсов»** | |
| Выполнил студент группы ИКБО-30-20 | Земар А. С. |
| Принял ассистент | Благирев М.М. |

|  |  |  |
| --- | --- | --- |
| Практические работы выполнены | « » 2022г. |  |
| «Зачтено» | « » 2022 г. |  |

Москва 2022

Цель работы

В рамках данной практической работы предполагается провести рефакторинг информационной системы, созданной в практических работах 1-6. Рефакторинг — это процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы. В основе рефакторинга лежит последовательность небольших эквивалентных (то есть сохраняющих поведение) преобразований. Поскольку каждое преобразование маленькое, программисту легче проследить за его правильностью, и в то же время вся последовательность может привести к существенной перестройке программы и улучшению её согласованности и четкости.

Рефакторинг системы следует проводить за счет перевода с процедурой парадигмы на ООП парадигму с переносом на определенную архитектуру.

Также на внешний слой архитектуры, если будет использоваться DDD или на слой интерфейсов, если будет использоваться чистая архитектура, внедрить какой-либо паттерн проектирования. Например, MVC или MVP.

Ход работы

Для выполнения поставленной задачи был преобразован под паттерн MVC, также был структурирован проект по правилам ООП (Рисунок 1)

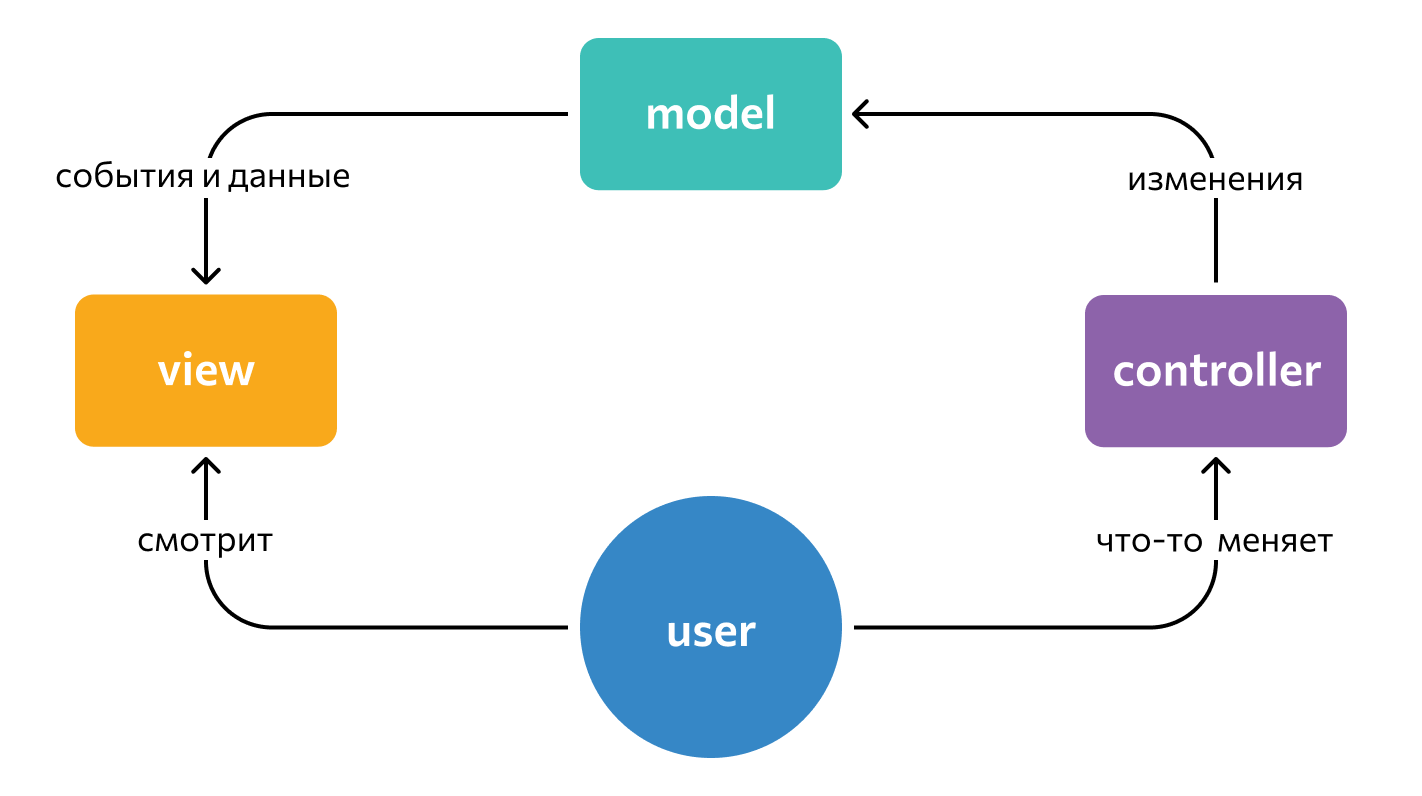


Рисунок 1 – MVC

Далее были сформированы модели, контроллеры и представления, согласно паттерну MVC. (Рисунок 2.2)

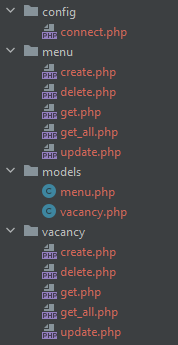


Рисунок 2.2 – модели контроллеры и представления

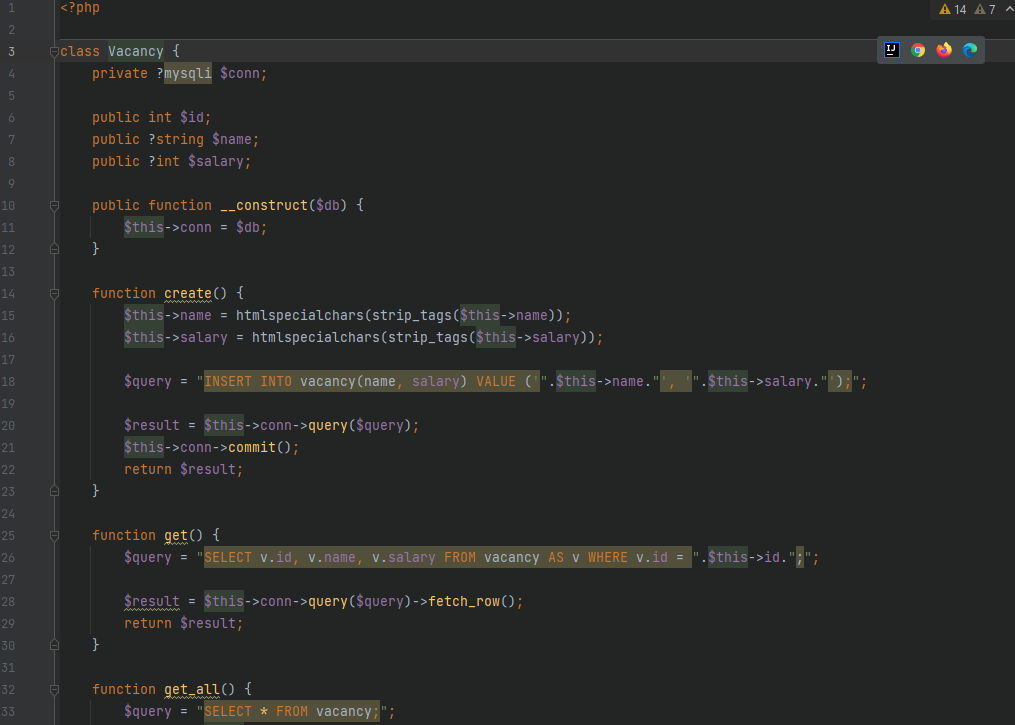


Рисунок 2.3 – Фрагмент класс Vacancy

Ответы на вопросы к практической работе

1. Назовите основные признаки ООП

Абстракция данных (позволяет выделить существенные признаки объекта, отличающие его от других объектов).

Инкапсуляция (означает, что каждый объект полностью описывается совокупностью своих свойств и методов).

Наследование (позволяет создавать новые классы на основе существующих).

Полиморфизм (означает способность объектов классов, связанных наследованием, реагировать различным образом на одно и то же сообщение).

1. Опишите как определить класс в РНР

class NAME {}.

1. Как создать экземпляр класса в PHP

$object = new Foo();

1. Опишите механизм наследования в PHP

class A {}, class B extends A

1. Опишите правила совместимости сигнатур

**Сигнатура** – это описание метода, включающее в себя: модификатор доступа, имя метода, список аргументов, тип возвращаемого значения.

Другими словами, сигнатура является совместимой, если она соответствует правилам контравариантности, делает обязательный параметр необязательным и если какие-либо новые параметры являются необязательными

1. Опишите методы и свойства Nullsafe

Данный оператор сокращает огромные куски кода с множественными проверками на null. Работает он так же, как доступ к свойству или методу, как указано выше, за исключением того, что если разыменование объекта выдаёт null, то будет возвращён null, а не выброшено исключение. Если разыменование является частью цепочки, остальная часть цепочки пропускается

1. Опишите понятие автоматическая загрузка классов

Автозагрузка – это процесс автоматической загрузки РНР-классов без явной их загрузки с помощью функций require (), require\_once (), include () или include\_once ().

Работает это следующим образом: как только программа пытается обратиться к несуществующему классу, вызывается специальная функция ＿autoload(), которая загружает класс.

Также в РНР есть более гибкая функция spl\_autoload\_register(), которая позволяет зарегистрировать цепочку из функций автозагрузки. Не найдя класс при помощи первой функции, РНР будет переходить ко второй и последующим функциям и либо найдет класс при помощи одной из функций, либо завершит выполнение программы с ошибкой.

1. Опишите конструкторы и деструкторы в РНР

**Конструктор** – это специальный метод класса, который автоматически выполняется в момент создания объекта до вызова всех остальных нестатических методов класса. Данный метод используется главным образом для инициализации объекта, обеспечивая согласованное состояние объекта перед началом работы.

**Деструктор** – это специальный метод класса, который автоматически выполняется в момент уничтожения объекта

1. Опишите понятие области видимости и модификаторы доступа в PHP

Область видимости для всех элементов класса (констант, свойств и методов) определяется с помощью модификаторов.

1. Опишите оператор разрешения области видимости

**Оператор разрешения области видимости** ( :: ) позволяет обращаться к статическим свойствам, константам и переопределённым свойствам или методам класса. При обращении к этим элементам извне класса, необходимо использовать имя этого класса

1. Опишите абстрактные классы и методы в РНР

Абстрактный класс – это базовый класс, который не предполагает создания экземпляров. На основе абстрактного класса нельзя создавать объекты, и любой класс, содержащий хотя бы один абстрактный метод, должен быть определён как абстрактный.

Методы, объявленные абстрактными, несут, по существу, лишь описательный смысл и не могут включать реализацию. При наследовании от абстрактного класса все методы, помеченные абстрактными в родительском классе, должны быть определены в дочернем классе и следовать обычным правилам наследования и совместимости сигнатуры.

1. Опишите интерфейсы в РНР

Интерфейс – специальная структура, полностью состоящая из абстрактных методов. Класс, реализующий интерфейс, обязан переопределить методы, определенные интерфейсом. Один класс может реализовать несколько интерфейсов.

Для создания интерфейса используется ключевое слово interface. Для того чтобы объявить класс, реализующий интерфейс, необходимо воспользоваться ключевым словом implements, который располагается в конце конструкции class

1. Что такое трейт и как это используется

**Трейт** – набор свойств и методов, который можно добавить в произвольный класс

1. Что такое магические методы? Приведите примеры

Магическими называются методы, которые переопределяют действия производимые над объектами. Вызываются такие методы в особых ситуациях – например, при обращении к неподдерживаемому свойству, вызове неподдерживаемого метода, попытке преобразования объекта в строку.

Магические методы объявляются в классе так же, как и обычные методы, однако должны иметь строго определенные имена: \_\_construct(), \_\_destruct(), \_\_call(), \_\_callStatic(), \_\_get(), \_\_set(), \_\_isset(), \_\_unset(), \_\_sleep(), \_\_wakeup(), \_\_serialize(), \_\_unserialize(), \_\_toString(), \_\_invoke(), \_\_set\_state(), \_\_clone() и \_\_debugInfo.

1. Что такое позднее статическое связывание

**Позднее статическое связывание** – это получение ссылки на вызываемый класс в контексте статического наследования. Более точно: это сохранение имени класса указанного в последнем «не перенаправленном вызове».

1. Что такое ковариантность и контравариантность

**Ковариантностью** называется сохранение иерархии наследования исходных типов в производных типах в том же порядке. Ковариантность позволяет дочернему методу возвращать более конкретный тип, чем тип возвращаемого значения его родительского метода.

**Контравариантностью** называется обращение иерархии исходных типов на противоположную в производных типах. Контравариантность позволяет типу параметра в дочернем методе быть менее специфичным, чем в родительском.

1. Опишите понятие чистой архитектуры?

Чистая архитектура – это способ организации программного кода, который способствует строгому разделению ответственности. Это достигается разделением на слои и следованием Dependency Rule (правилу зависимостей).

Чистая архитектура объединяет в себе идеи нескольких других архитектурных подходов, которые сходятся в том, что архитектура должна.

1. Сформулируйте правило зависимостей

Правило зависимостей (dependency rule): «Зависимости в исходном коде должны быть направлены внутрь, в сторону высокоуровневых политик». То есть ничто во внешнем уровне не должно влиять на внутренние уровни.

1. Чем определяются сущности, чем они могут быть?

Сущности определяются бизнес-правилами предприятия. Сущность может быть объектом с методами или она может представлять собой набор структур данных и функций (заключающих в себе критические бизнес-правила уровня предприятия). Они должны быть доступны для использования различными приложениями.

1. Что такое слой сценариев?

В **слое сценариев** реализуется специфика бизнес-правил. Он инкапсулирует и реализует все случаи использования системы. Эти сценарии реализуют поток данных в и из слоя Cущностей для реализации бизнес-правил.

1. Что такое DTO?

**DTO** (Data Transfer Object) являются простейшими объектами (без логики), описывающими структуры данных, передаваемых через границу

1. Что является деталью в рамках чистой архитектуры?

К **деталям** можно отнести устройства ввода/вывода, базы данных, веб-системы, серверы, фреймворки, протоколы обмена данными и т.д. В общем случае, детали – это всё, что позволяет людям, другим системам и программистам взаимодействовать с политикой, никак не влияя на её поведение

1. Опишите принципы организации компонентов?

Компоненты представляют наименьшие сущности, которые можно развёртывать в составе системы. Компоненты должны следовать принципам связности компонентов и принципам сочетаемости компонентов.

Принципы, определяющие связность компонентов:

* принцип эквивалентности повторного использования и выпусков (единица повторного использования есть единица выпуска);
* принцип согласованного изменения (в один компонент должны включаться классы, изменяющиеся по одним причинам и в одно время; в разные компоненты должны включаться классы, изменяющиеся в разное время и по разным причина);
* принцип совместного повторного использования (не вынуждайте пользователей компонента зависеть от того, чего им не требуется).

Принципы, определяющие сочетаемость компонентов:

* принцип ацикличности зависимостей (циклы в графе зависимостей компонентов недопустимы);
* принцип устойчивых зависимостей (зависимости должны быть направлены в сторону устойчивости);

принцип устойчивости абстракций (устойчивость компонента пропорциональна его абстрактности).

1. Опишите принципы дизайна архитектуры?

Принципы дизайна архитектуры (именуемые SOLID):

* принцип единственной ответственности (каждый программный модуль имеет одну и только одну причину для изменения);
* принцип открытости/закрытости (система должна быть легко расширяемой и безопасной от влияния изменений);
* принцип подстановки Барбары Лисков (при создании программных систем из взаимозаменяемых частей эти части должны соответствовать контракту, который позволяет заменять эти части друг другом);
* принцип разделения интерфейса (следует избегать зависимости от всего, что не используется);

принцип инверсии зависимостей (код, реализующий высокоуровневую политику, не должен зависеть от кода, реализующего низкоуровневые детали. Напротив, детали должны зависеть от политики).

1. Опишите понятие DDD (Domain Driven Design, предметно-ориентированное проектирование).

**Предметно-ориентированное проектирование** (DDD) – это набор принципов и схем, направленных на создание оптимальных систем объектов. Сводится к созданию моделей предметных областей. В эти модели входит бизнес-логика, устанавливающая связь между реальными условиями области применения продукта и кодом

1. Что такое ограниченный контекст (Bounded Context)?

**Ограниченный контекст** – это явная граница, внутри которой существует модель предметной области, которая отображает единый язык в модель программного обеспечения. В каждом ограниченном контексте существует только один единый язык.

1. Что такое Ubiquitous Language (Единый язык)?

**Единый язык** – это коллективный язык терминов, который совместно используется экспертами в предметной области и разработчиками. Фактически на нём говорят все участники команды, разрабатывающей проект. Цель определения единого языка – устранить вероятность неоднозначного толкования высказываний при обмене информацией между заказчиками и пользователями.

1. Что такое Смысловое ядро (Core domain)?

**Смысловое ядро** – такая часть продукта, которая обеспечивает его конкурентные преимущества и формирует его истинную ценность для бизнеса. Большинство DDD проектов сосредоточены как раз на смысловом ядре, поскольку именно оно и представляет первостепенное значение для организации. Без смыслового ядра не будет и продукта

1. Что такое Предметная область (Domain).

**Предметная область** (domain) – область знаний или некоторой деятельности, характеризуемая множеством понятий и терминологией, понятной специалистам в этой предметной области. Предметная область определяет рамки, в которых проявляются возможности программы, а также требования, которым эта программа должна удовлетворять

1. Что такое Пространство задач и пространство решений

Пространство задач – части предметной области, которые необходимы, чтобы создать смысловое ядро. Это комбинация смыслового ядра и подобластей, которое это ядро должно использовать.

Пространство решений – один или несколько ограниченных контекстов, набор конкретных моделей программного обеспечения. Разработанный ограниченный контекст – это конкретное решение, представление реализации

Ссылка на удаленный репозиторий проекта

Полный код проекта можно найти по ссылке:

https://github.com/MycroftBrooks/PHP\_mirea

ЗАКЛЮЧЕНИЕ

Была создана конфигурация серверного программного обеспечения на наборе LAMP и проверена ее работоспособность. Данная конфигурация будет использоваться для выполнения следующих практических работ по данной дисциплине и для выполнения курсового проектирования.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. «Конспект лекции №1»: [Электронный ресурс]. URL: <https://online-edu.mirea.ru/mod/resource/view.php?id=403993> (дата обращения 08.09.2022)
2. «Руководство по Docker Compose»: [Электронный ресурс]. URL: <https://habr.com/ru/company/ruvds/blog/450312/> (дата обращения 08.09.2022)
3. «Docker Hub»: [Электронный ресурс]. URL: <https://hub.docker.com/> (дата обращения 08.09.2022)