Java Pro 7.6

TEL-RAN
by Starta Institute

Паттерн MVC. Гексогональная архитектура

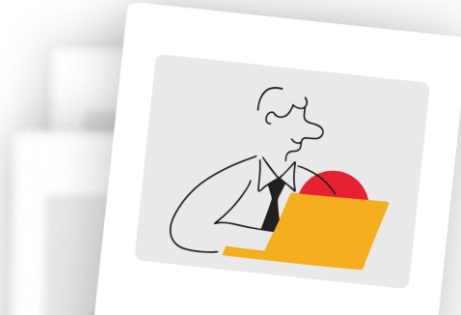
ПРЕПОДАВАТЕЛЬ

TEL-RAN
by Starta Institute

Юрий Костяной

Java/Kotlin backend-разработчик

- 3+ года опыта в коммерческой разработке
- 2+ года опыта в преподавании
- Проекты по интеграции сторонних платформ, CRM
- Проблемно-ориентированный подход в преподавании



ВАЖНО:

- Камера должна быть включена на протяжении всего занятия.
- Если у Вас возник вопрос в процессе занятия, пожалуйста, поднимите руку и дождитесь, пока преподаватель закончит мысль и спросит Вас, также можно задать вопрос в чате или когда преподаватель скажет, что начался блок вопросов.
- Организационные вопросы по обучению решаются с кураторами, а не на тематических занятиях.
- Вести себя уважительно и этично по отношению к остальным участникам занятия.
- Во время занятия будут интерактивные задания, будьте готовы включить камеру или демонстрацию экрана по просьбе преподавателя.

TEL-RAN
by Starta Institute

ПЛАН ЗАНЯТИЯ

1. Повторение
2. Основной блок
3. Вопросы по основному блоку
4. Домашняя работа

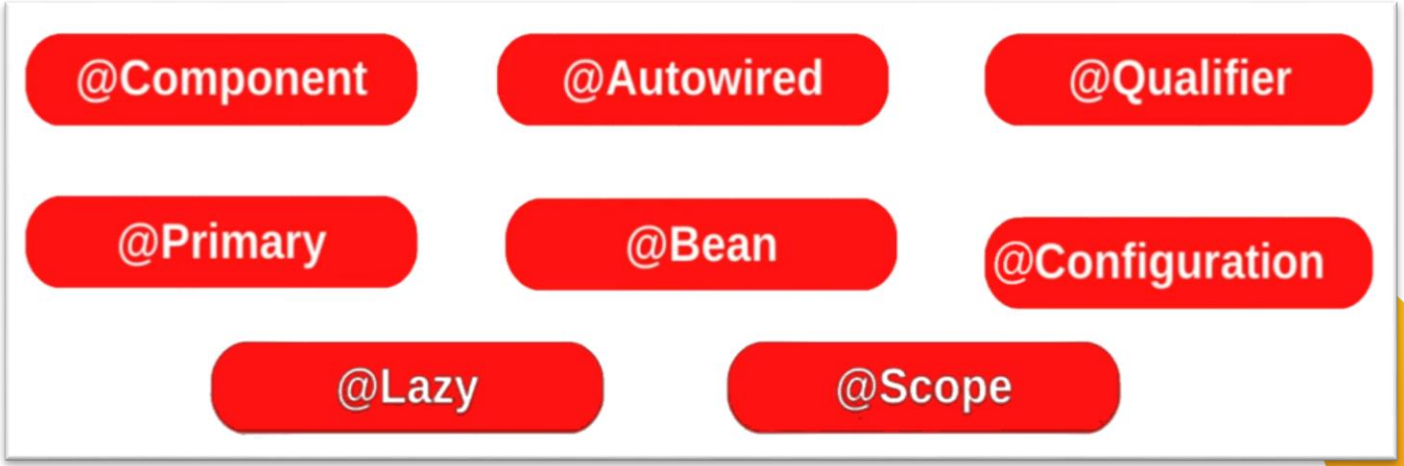
TEL-RAN
by Starta Institute

1 ПОВТОРЕНИЕ ИЗУЧЕННОГО

Повторение

TEL-RAN
by Starta Institute

Расшифруйте значение следующих аннотаций:



Повторение

TEL-RAN
by Starta Institute

Расшифруйте значение следующих аннотаций:

@Component – показывает, что класс является кандидатом для создания бина.

@Autowired – показывает, что нужно внедрить зависимость автоматически.

@Qualifier – уточняет, какой бин нужно автоматически внедрить, если в контексте есть несколько бинов, подходящих для внедрения.

@Primary – указывает, какой бин считать бином для автоматического внедрения, если в контексте присутствуют другие бины, следующие тому же интерфейсу.

@Bean – помечает методы внутри класса конфигурации, которые производят бины.

@Configuration – помечает класс конфигурации контекста.

@Lazy – помечает бины и зависимости, которые можно инициализировать лениво, т.е. при первом запросе этого бина.

@Scope – определяет область видимости бина (singleton или prototype).

Повторение

Исправьте ошибку в коде

```
@Component
public class PassPhraseHolder {
    private String phrase;

    public void receivePhraseFromServer() {
        try {
            URL url = new URL("pass.service-enterprise.com");
            try (InputStream input = url.openStream()) {
                phrase = new String(input.readAllBytes());
            }
        } catch (IOException e) { throw new RuntimeException(e); }
    }
    public String getPhrase() { return phrase; }
}
```

```
public static void main(String[] args) {
    AnnotationConfigApplicationContext context =
        new AnnotationConfigApplicationContext(MyAppConfig.class);
    PassPhraseHolder holder = context.getBean(PassPhraseHolder.class);
    String secretPhrase = holder.getPhrase();
    String prefix = secretPhrase.substring(0, 3);
    context.close();
}
```

Повторение

Исправьте ошибку в коде

Если не указать *@PostConstruct*, то поле *phrase* так и не будет проинициализировано, что приведёт к *NullPointerException* при получении значения переменной *prefix*.

```
@Component
public class PassPhraseHolder {
    private String phrase;
    @PostConstruct
    public void receivePhraseFromServer() {
        try {
            URL url = new URL("pass.service-enterprise.com");
            try (InputStream input = url.openStream()) {
                phrase = new String(input.readAllBytes());
            }
        } catch (IOException e) { throw new RuntimeException(e); }
    }
    public String getPhrase() { return phrase; }
}
```

```
public static void main(String[] args) {
    AnnotationConfigApplicationContext context =
        new AnnotationConfigApplicationContext(MyAppConfig.class);
    PassPhraseHolder holder = context.getBean(PassPhraseHolder.class);
    String secretPhrase = holder.getPhrase();
    String prefix = secretPhrase.substring(0, 3);
    context.close();
}
```

Повторение

TEL-RAN
by Starta Institute

Исправьте ошибки в коде

```
@Component
public class BillingService { }
```

```
public class DataService { }
```

```
@Configuration
public class MyAppConfig {

    public DataService dataService() {
        return new DataService();
    }
}
```

Повторение

TEL-RAN
by Starta Institute

Исправьте ошибки в коде

```
@Component
public class BillingService { }
```

```
public class DataService { }
```

```
@Configuration
@ComponentScan("org.example.pojo")
public class MyAppConfig {
    @Bean
    public DataService dataService() {
        return new DataService();
    }
}
```

Без аннотации *@Bean* Spring не рассматривает метод как поставщика бинов.

Без аннотации *@ComponentScan* Spring не будет искать классы, помеченные аннотацией *@Component*.

Повторение

Исправьте ошибку в коде


```
@Configuration
public class DatabaseConfig {

    @Bean
    public DataSource primaryDataSource() {
        return new PrimaryDataSource();
    }

    @Bean
    public DataSource secondaryDataSource() {
        return new SecondaryDataSource();
    }
}
```

```
public static void main(String[] args) {
    AnnotationConfigApplicationContext context = new AnnotationConfigApplicationContext(DatabaseConfig.class);
    DataSource source = context.getBean(DataSource.class);
    context.close();
}
```

Повторение

TEL-RAN
by Starta Institute

Исправьте ошибку в коде

При наличии двух бинов, следующих одному интерфейсу, Spring не может выбрать, какой бин использовать, поэтому нужно либо получать бин по имени, либо использовать *@Primary*.

```
@Configuration
public class DatabaseConfig {
    @Primary
    @Bean
    public DataSource primaryDataSource() {
        return new PrimaryDataSource();
    }

    @Bean
    public DataSource secondaryDataSource() {
        return new SecondaryDataSource();
    }
}
```

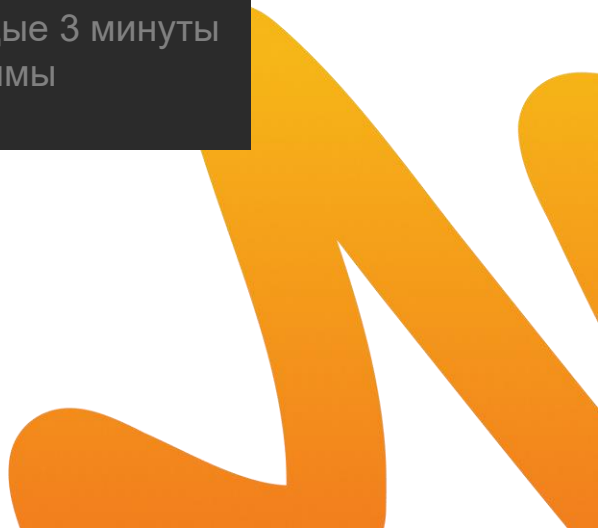
```
public static void main(String[] args) {
    AnnotationConfigApplicationContext context = new AnnotationConfigApplicationContext(DatabaseConfig.class);
    DataSource source = context.getBean(DataSource.class);
    context.close();
}
```

Повторение

Исправьте ошибку в коде.

TEL-RAN
by Starta Institute

```
@Component
@Lazy
public class TaskScheduler {
    // получение данных от стороннего сервиса каждые 3 минуты
    // данные нужны для корректной работы программы
}
```



Повторение

Исправьте ошибку в коде.

TEL-RAN
by Starta Institute

```
@Component
public class TaskScheduler {
    // получение данных от стороннего сервиса каждые 3 минуты
    // данные нужны для корректной работы программы
}
```

Применение аннотации *@Lazy* недопустимо с таким типом классов, т.к. предполагается, что его работа должна начаться немедленно при старте программы. Если приложение будет ожидать вызова бина, то бин так и не начнёт свою работу.

Повторение

В чём прикол мема?

TEL-RAN
by Starta Institute

2 ОСНОВНОЙ БЛОК

Введение

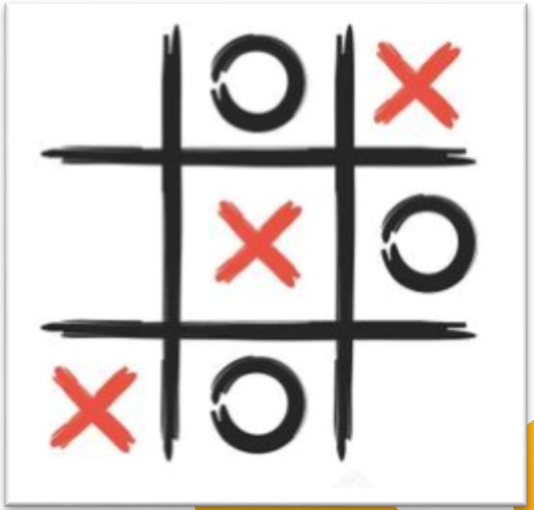
TEL-RAN
by Starta Institute

- Модель, вид и контролёр
- Горе луковое

Проблема

TEL-RAN
by Starta Institute

Представьте, что Вы написали консольную игру «Крестики-нолики». Игра хорошо работает и вот уже первое игровое издательство готово разместить Вашу игру в store. Однако игра не будет популярна у пользователей игровых приставок, потому что у неё нет графического интерфейса (GUI). Вы дописываете GUI для консолей, но пользователи смартфонов с IOS и Android тоже хотят Вашу игру. Есть и желающие поиграть на PC и через браузер онлайн. Логика игры на всех платформах будет той же, но отображение элементов GUI и способ взаимодействия с ними (клик, тап, кнопка джойстика и т.д.) будут разные.



Как можно переиспользовать код основной логики программы, меняя только GUI?

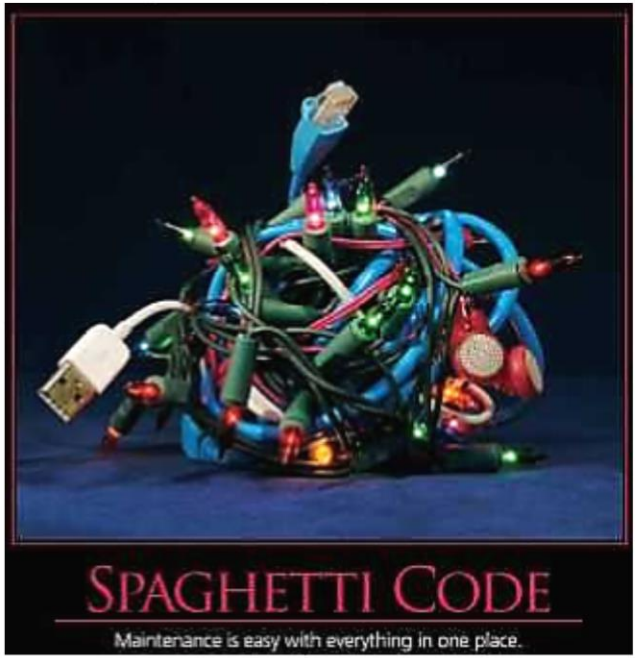
Моделька, вид и контролёр

Спагетти код

TEL-RAN
by Starta Institute

Если игра написана так, что логика работы зависит от элементов GUI, т. е. данные передаются из классов логики апрямую в класс, отрисовывающий GUI, то логика и GUI имеют сильную связь. Обычно такой код сильно запутан и его сложно обслуживать (антипаттерн **Спагетти код**). А заменить в приложении один GUI на другой практически невозможно без полного переписывания программы.

Чтобы избежать Спагетти кода, нужно всё разложить «по полочкам», т.е. отделить логику от GUI.

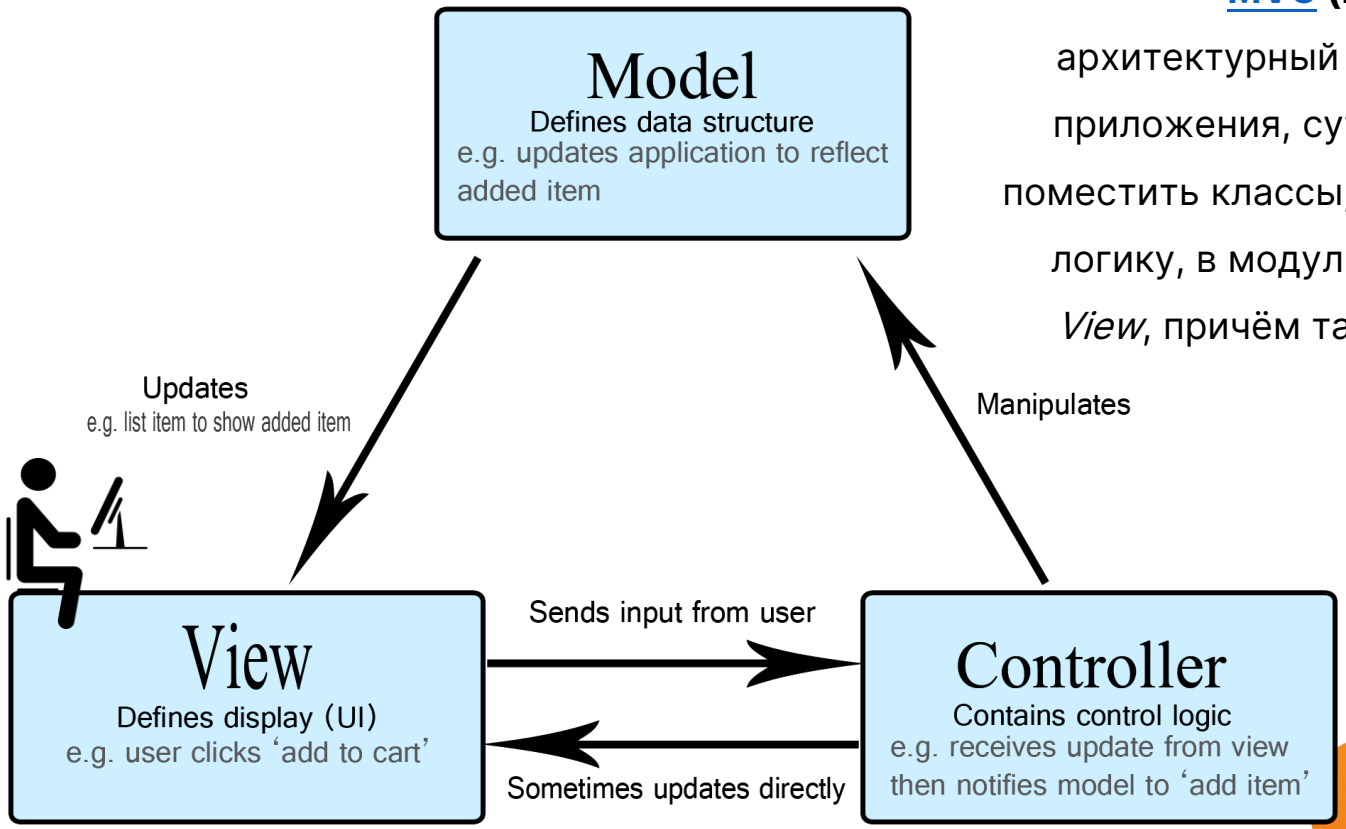


Моделька, вид и контролёр

Паттерн MVC

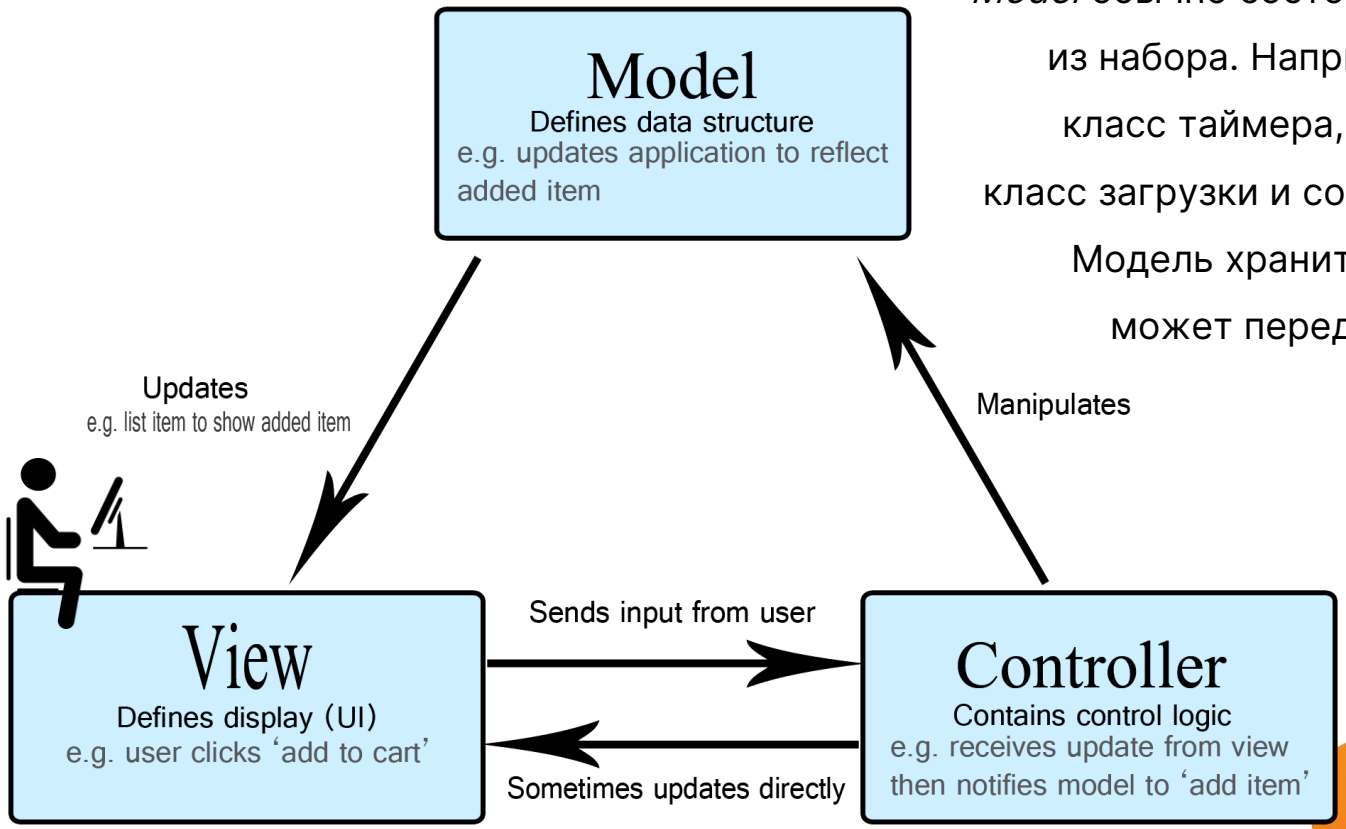
TEL-RAN by Starta Institute

MVC (Model-View-Controller) – архитектурный паттерн проектирования приложения, суть которого в том, чтобы поместить классы, реализующие основную логику, в модуль *Model*, а GUI – в модуль *View*, причём так, чтобы *Model* и *View* не знали друг о друге. Для этого применяется промежуточный класс *Controller*.



Моделька, вид и контролёр

Паттерн MVC



Model обычно состоит не из одного класса, а из набора. Например, класс логики игры, класс таймера, отсчитывающего время, класс загрузки и сохранения рекордов и т.д. Модель хранит состояние программы и может передавать его во View в виде Снимка.

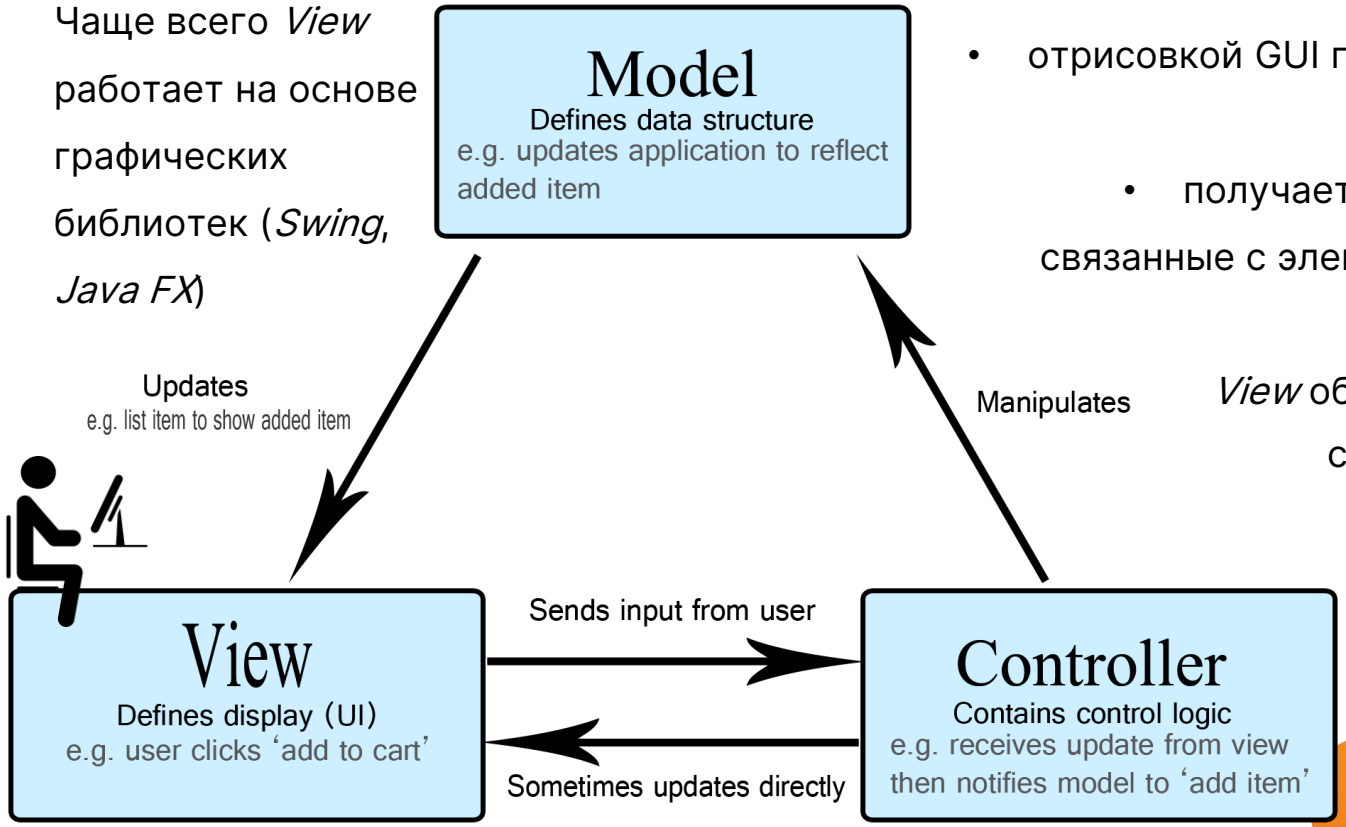
Снимок – это класс данных, помещаемый в общий модуль (например, *commons*)

Моделька, вид и контролёр

Паттерн MVC

TEL-RAN by Starta Institute

Чаще всего *View* работает на основе графических библиотек (*Swing*, *Java FX*)



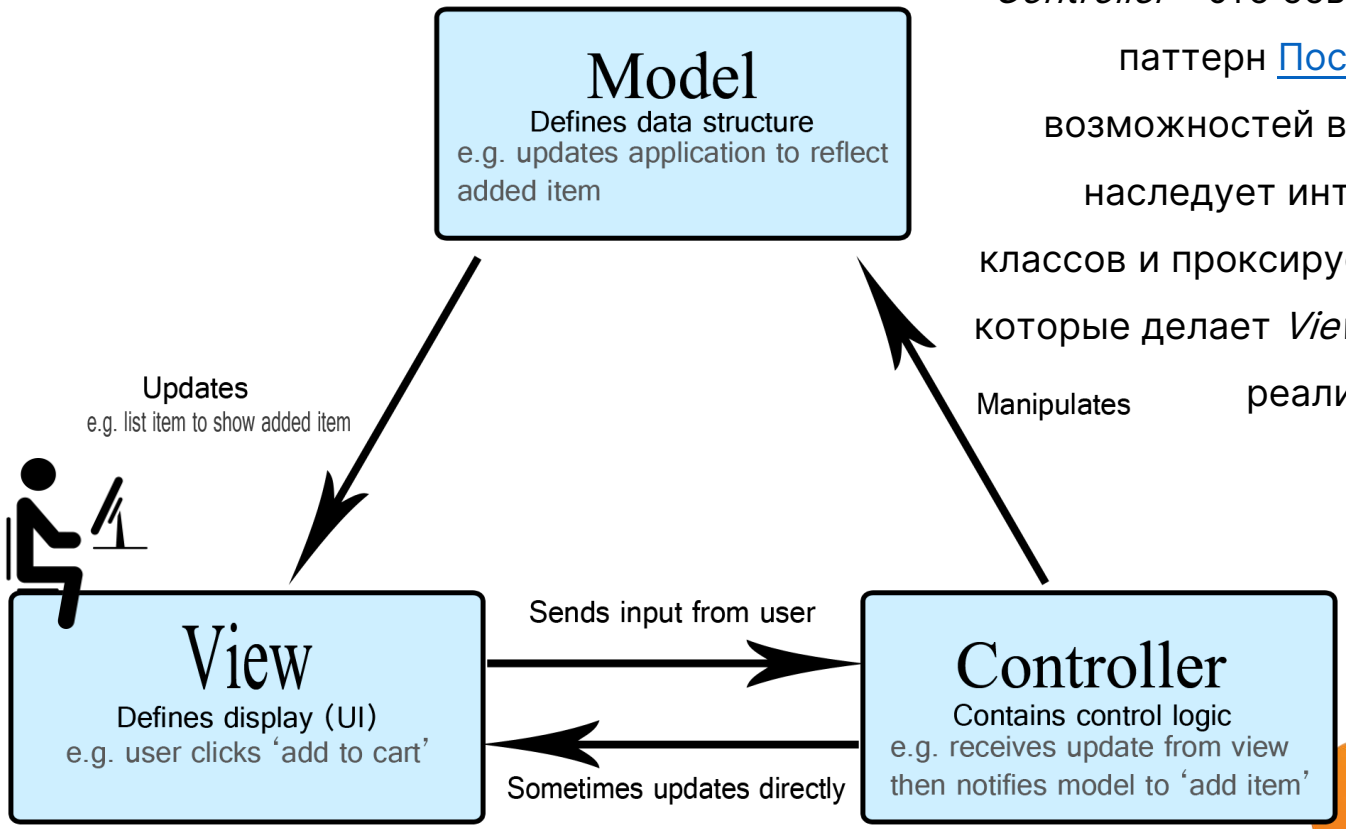
View занимается

- отрисовкой GUI по полученном от модели Снимку.
- получает действия пользователя, связанные с элементами интерфейса или вводом данных.

View обычно имеет событийную структуру (Наблюдатель) и на каждое действие пользователя вызывает какой-либо метод *Controller*'а.

Моделька, вид и контролёр

Паттерн MVC



Controller – это обычно класс, реализующий паттерн Посредник для объединения возможностей всех классов из *Model*. Он наследует интерфейсы всех указанных классов и проксирует этим классам вызовы, которые делает *View*. В контроллере обычно реализована организационная Логика.

Моделька, вид и контролёр

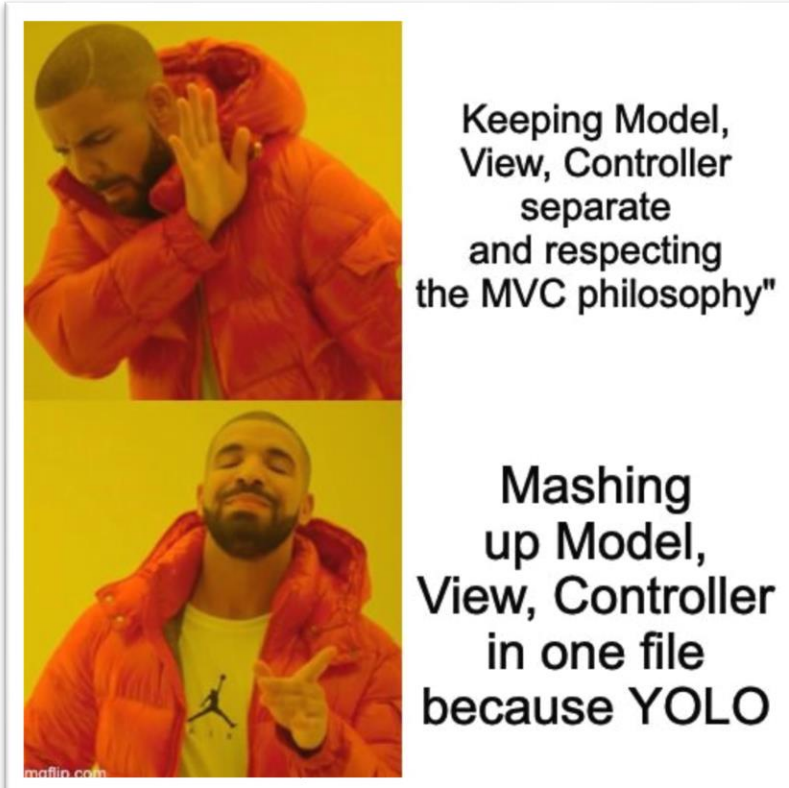
Пример использования паттерна MVC

MinerUISample.zip

Задание

TEL-RAN
by Starta Institute

Напишите приложение Крестики-нолики с использованием паттерна MVC. Модуль *View* можно реализовать с помощью консоли.



Проблема

TEL-RAN
by Starta Institute

Spring помогает автоматизировать enterprise-приложения.

Но что конкретно делают такие приложения?

Горе луковое
Enterprise-приложения

TEL-RAN
by Starta Institute

Работа практически всех современных Enterprise-приложений сводится к

- генерации, накапливанию и хранению данных о клиентах, пользователях и бизнес-процессах;
- выполнению бизнес-логики, т.е. организации взаимодействия между клиентским приложением, сторонними сервисами, базой данных;
- обмену информации с клиентскими приложениями, БД и сторонними сервисами.

Например, сервер стримингового сервиса типа Youtube должен уметь регистрировать нового клиента и хранить его аккаунт, получать запросы от пользователей на просмотр видео, собирать информацию о просмотренных видео для выдачи рекомендаций. По решению пользователя добавить видео в понравившиеся или плейлист, ассоциируя его с клиентом. По предпочтениям клиента сервер должен подбирать подходящую рекламу, которую он запрашивает у стороннего рекламного сервера. В случае нарушения правил сервиса, сервер должен ограничивать деятельность пользователя и т.д.

Горе луковое

Чистая архитектура

TEL-RAN
by Starta Institute

Т.к. всё, что делает приложение, можно выделить в отдельные группы (слои) – хранение данных, бизнес-логика и обмен данными – возникла идея создать универсальную удобную архитектуру Enterprise-приложения. Это направление назвали **Чистая архитектура** (по аналогии с *Чистым кодом*) – это философия дизайна программного обеспечения, которая разделяет компоненты на определенные уровни.

Если компонент зависит от всех других компонентов, мы не знаем, какие побочные эффекты будет иметь изменение одного компонента, что затрудняет поддержку кодовой базы и еще более затрудняет ее расширение и декомпозицию.

Горе луковое
DDD

TEL-RAN
by Starta Institute

В процессе разработки чистой архитектуры возник вопрос, какое место отводить бизнес-логике. Здесь на помощь пришёл [Domain Drive Design] **(DDD)** – **предметно-ориентированное проектирование** – подход к разработке приложений, основанный на выделении доменов, т.е. предметных областей.

Это позволяет нам структурировать программное обеспечение в форме, которая имеет смысл не только для программистов, но и для всех, кто вовлечен в предметную область. Использование DDD при проектировании архитектуры сложных корпоративных информационных систем позволяет сделать проще поддержание изменений в проекте и эффективнее тестировать новые релизы.

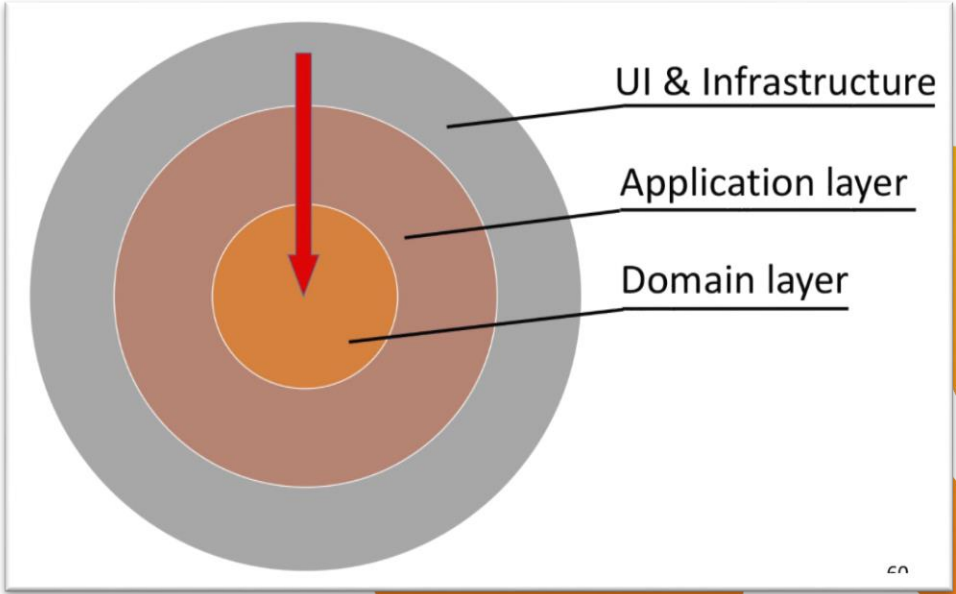
Горе луковое

Луковая (onion) архитектура

TEL-RAN by Starta Institute

DDD позволил выделить предметную область в отдельный слой и постулировал, что этот слой не должен зависеть от того, какая база данных использована или по какому каналу и протоколу к нам поступили данные, т.к. эти технологии меняются (вчера реляционные БД, сегодня – NoSQL; вчера – SOAP, сегодня – REST и т.д.)

Запрос пользователя (красная стрелка) поступает через слой инфраструктуры (условно, каналы получения данных – как передать) и интерфейса пользователя (не обязательно графического, это может быть API – что передать). Затем приложение обрабатывает полученные данные, делает первичную валидацию. И только после этого данные поступают в бизнес-логику.

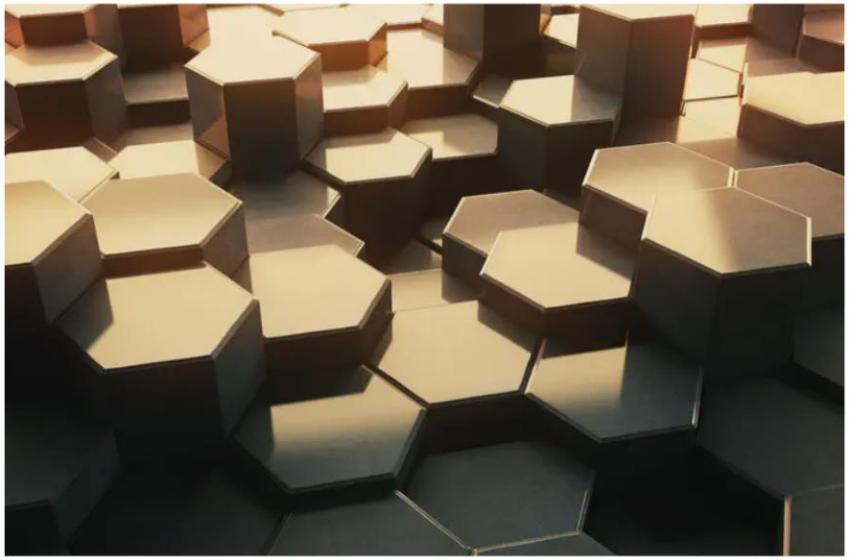


Горе луковое

Гексагональная архитектура

TEL-RAN
by Starta Institute

Если луковая архитектура – это идея, то **Гексагональная архитектура** – это её реализация, представляющая архитектурный паттерн, позволяющий структурировать приложение таким образом, чтобы это приложение можно было разрабатывать и тестировать в изоляции, не завися от внешних инструментов и технологий (фреймворков, баз данных или внешних сервисов).

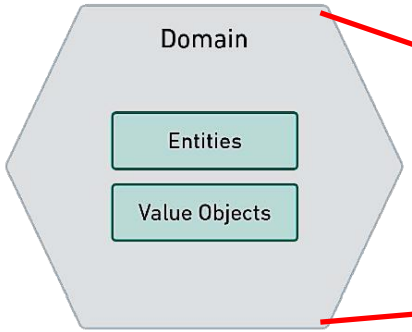


Горе луковое

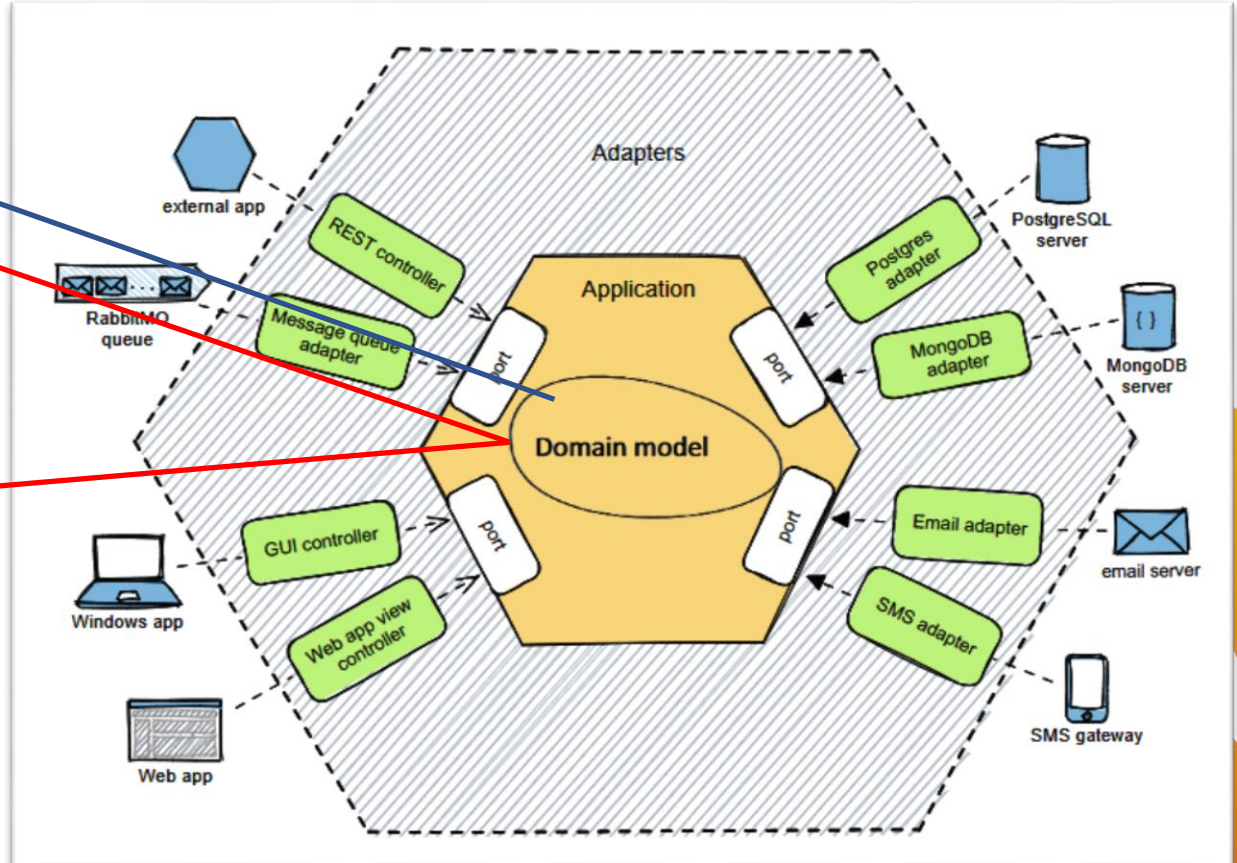
Гексагональная архитектура

TEL-RAN
by Starta Institute

Вся бизнес-логика заключена внутри приложения



Внутри домена находятся сущности и объекты, играющие важную роль в предметной области.

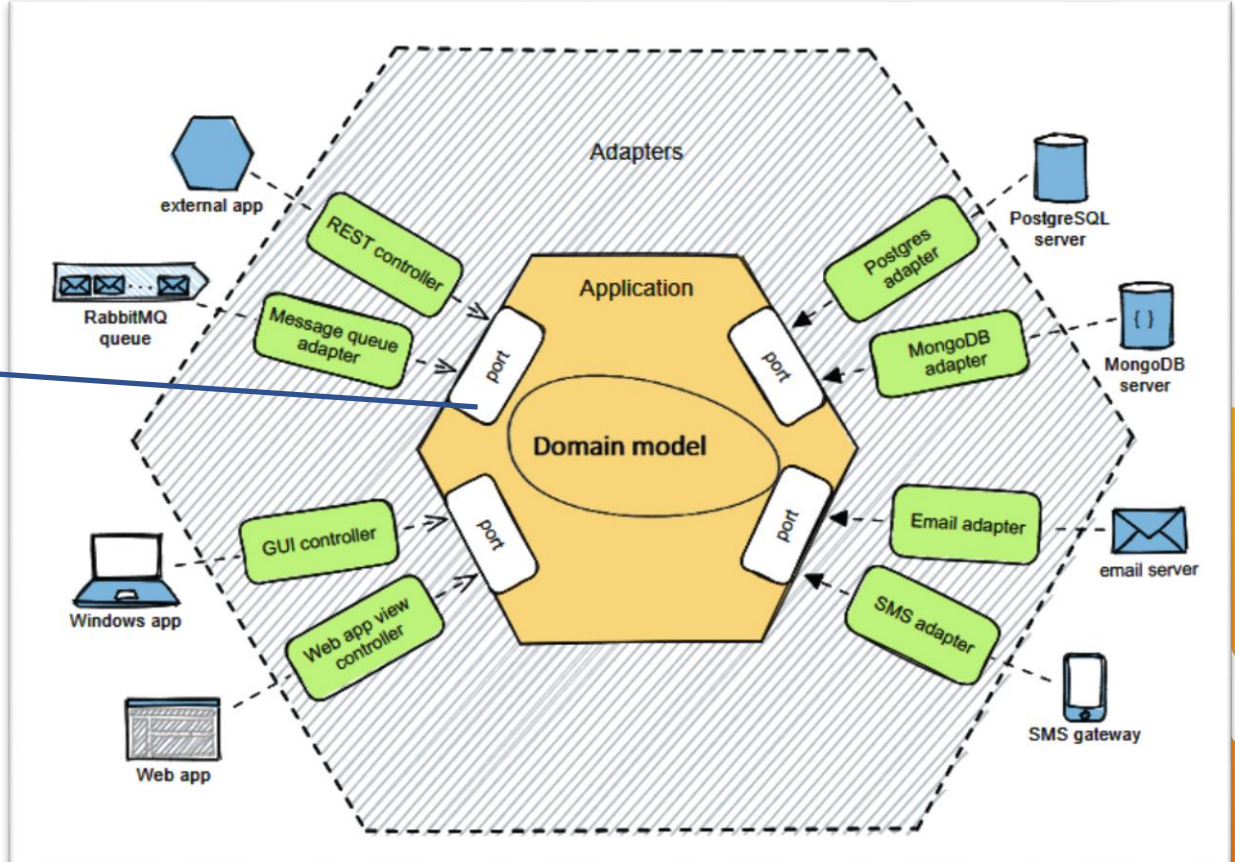


Горе луковое

Гексагональная архитектура

TEL-RAN by Starta Institute

Чтобы отделить бизнес-логику от внешнего мира, сделаем так, чтобы приложение взаимодействовало с внешним миром только через порты. Эти порты описывают суть коммуникации между двумя сторонами. Для приложения не имеет значения, каковы технические детали реализации портов. Порты чаще всего реализуются в виде интерфейсов. Так достигается инверсия зависимостей (буква D акронима SOLID)

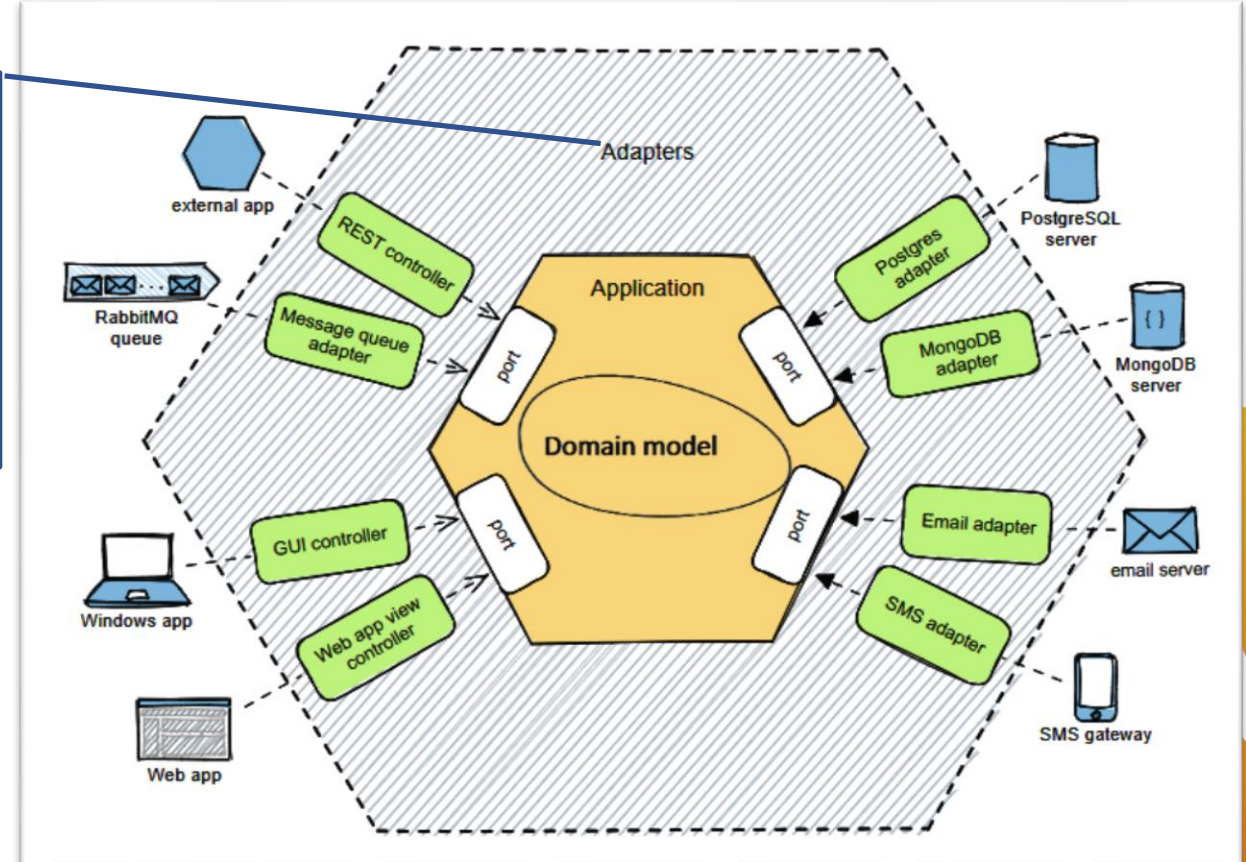


Горе луковое

Гексагональная архитектура

TEL-RAN
by Starta Institute

Адаптеры обеспечивают связь приложения с внешним миром. Они преобразуют внешние сигналы в форму, понятную приложению. Адаптеры взаимодействуют с приложением только через порты.

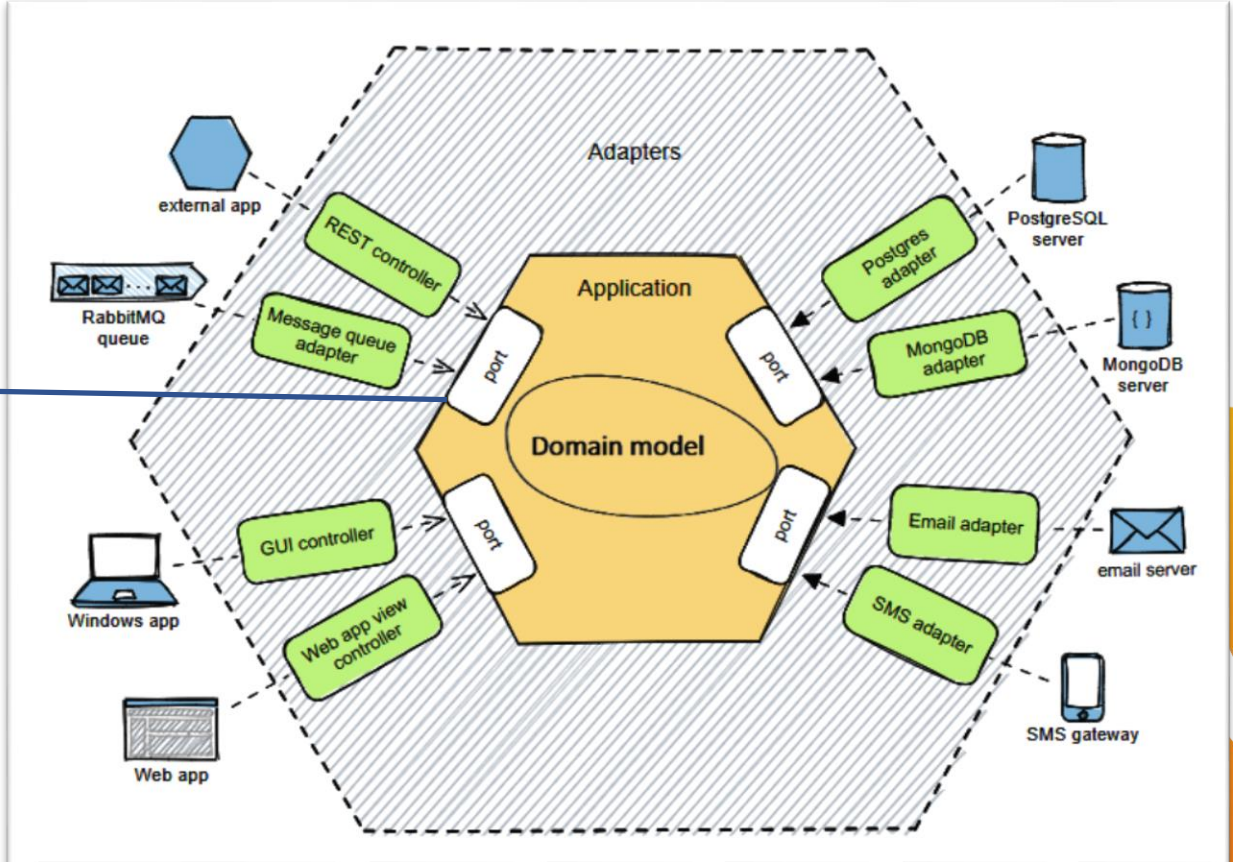


Горе луковое

Гексагональная архитектура

TEL-RAN
by Starta Institute

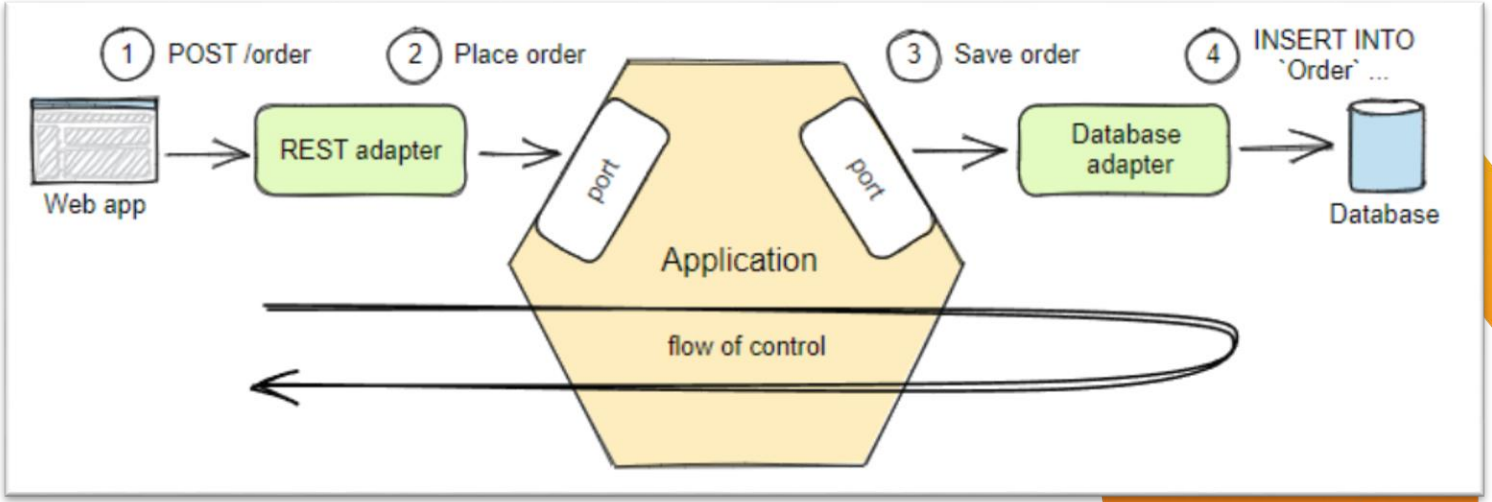
Любой порт может иметь несколько адаптеров. Адаптеры могут быть взаимозаменяемыми с обеих сторон, не затрагивая бизнес-логику. Это позволяет легко масштабировать решение для использования новых интерфейсов или технологий.



Горе луковое

Пример гексагональной архитектуры

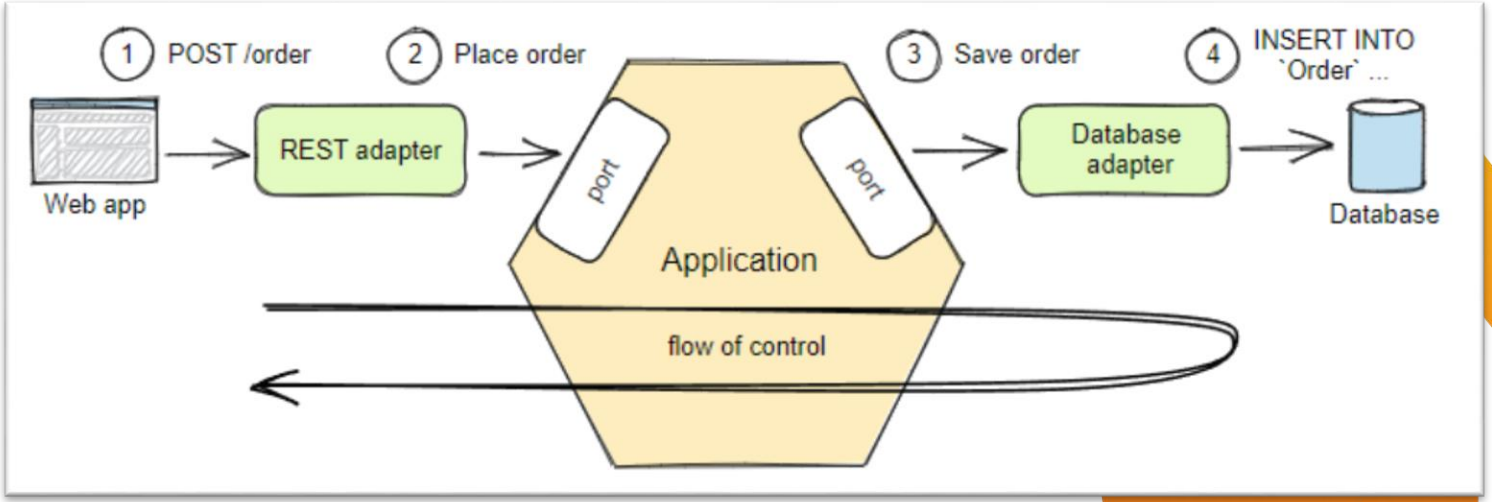
В приложении для кофейни может быть пользовательский интерфейс кассы, который обрабатывает прием заказов на кофе. Когда бариста отправляет заказ, REST-адаптер принимает HTTP-запрос POST и преобразует его в форму, понятную порту. При вызове порта запускается бизнес-логика, связанная с оформлением заказа внутри приложения. Само приложение не знает, что оно работает через REST API.



Горе луковое

Пример гексагональной архитектуры

С другой стороны, приложение взаимодействует с портом, который позволяет сохранять заказы. Подключение к реляционной БД реализовано через адаптер базы данных. Адаптер принимает информацию, поступающую из порта, и преобразует её в SQL-запрос для хранения заказа в базе данных. Само приложение не знает, как реализован данный функционал, и какие технологии при этом используются.



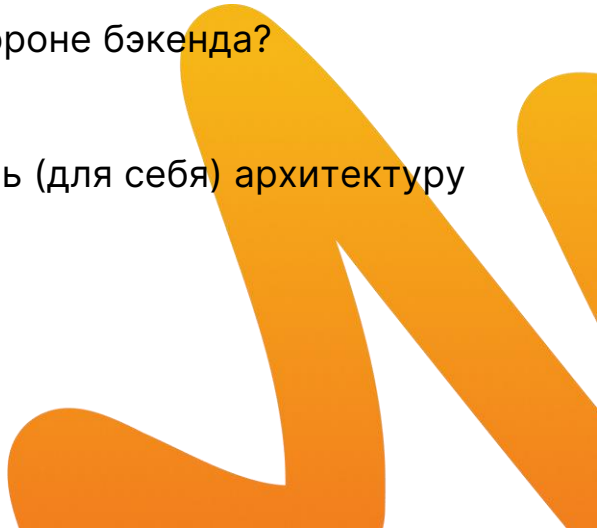
Задание

TEL-RAN
by Starta Institute

Придумайте идею собственного приложения:

- Что будет делать Ваше приложение?
- Какую информацию от пользователя оно будет принимать?
- Каким образом оно будет принимать эту информацию?
- Каким образом оно будет хранить эту информацию?
- Потребуется ли взаимодействие с другими сервисами на стороне бэкенда?

Изложите ответы на вопросы письменно. Попробуйте зарисовать (для себя) архитектуру будущего приложения. Время подготовки 15 минут.



TEL-RAN
by Starta Institute

3 Домашнее задание

Домашнее задание

TEL-RAN
by Starta Institute

1 Напишите программу для игры с компьютером по правилам классического *Лото* (без ставок).

https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D1%82%D0%BE

У пользователя и у компьютера есть свои наборы полей. Выполните приложение, используя паттерн MVC. В качестве View реализуйте взаимодействие через консоль. По Вашему желанию, попробуйте сделать графический интерфейс с помощью библиотеки Swing https://java-online.ru/swing-windows.xhtml

2 В придуманной ранее на лекции архитектуре приложения максимально подробно опишите доменную модель (основные сущности и процессы их взаимодействия). Какие поля важны для сущностей? Пропишите, что может делать пользователь (use cases) и что программа будет делать в ответ на его действия (входные проверки, шаги выполнения, в каком виде будет результат, высылаемый как ответ пользователю).

ЗАКЛЮЧЕНИЕ

TEL-RAN
by Starta Institute