Нижегородский государственный университет им. Н.И. Лобачевского

От требований к архитектуре: разработка архитектуры приложения на примере

При поддержке компании Intel

Козинов Е.А., кафедра математического обеспечения ЭВМ, факультет ВМК

Содержание

- Постановка задачи
- Поэтапная разработка архитектуры:
  - Требования
  - Послойная архитектура, выделение подсистем
  - Применяемые паттерны
- Подсистемы – пример реализации
- Реализация подсистем на разных уровнях архитектуры:
  - Уровень модели приложения. Вопрос расширяемости
  - Уровень инфраструктуры. Преобразование объектов между приложением и системой хранения
  - Уровень представления. Преобразование интерфейса от модели к представлению
  - Уровень отображения. Пример реализации отображений

Цели

- Основная цель – рассмотрение примера приложения разработанного на основе компонентного подхода.
- При демонстрации примера планируется рассмотреть:
  - Пример разработки архитектуры приложения.
  - Пример применения шаблонов проектирования.

Основные требования

ЗАДАЧА

Постановка задачи…

- На рынок выходит новая компания авиаперевозчик.
  - Компания может приобретать свои самолеты для осуществления авиаперевозок. С самолетами связаны рейсы.
  - Каждый конкретный рейс может быть запланирован и отменен.
  - Для перевозок используются аэропорты сторонних компаний.

Постановка задачи…

- Для продвижения своих услуг компания хочет разработать программный комплекс, состоящий из нескольких частей.
- ***Часть первая – управление*** (используется менеджерами)
  - Менеджеры должны уметь управлять самолетами компании.
  - Менеджеры должны уметь через систему регистрировать рейсы.
    - При регистрации рейса менеджер выбирает точку отлета и приземления, а также время вылета и приземления.
  - Рейсы можно отменять.
    - Отменить рейс может менеджер и аэропорт.

Постановка задачи

- ***Часть вторая – клиентская***
  - Клиенты могут через программный комплекс покупать билеты на зарегистрированные рейсы.
  - При покупке предполагается, что покупатель задает точку отправления и точку назначения. Далее покупателю предлагается на выбор несколько возможных маршрутов.
  - Выбрав маршрут, покупатель регистрируется в системе и оплачивает покупку.
  - При отмене рейса покупателю сообщается об изменении маршрута или возможности вернуть потраченные средства.

Основные этапы

РАЗРАБОТКА АРХИТЕКТУРЫ

Разработка архитектуры…

- Что необходимо сделать вначале?

Разработка архитектуры …

- Что необходимо сделать вначале?
  - **Выделить требования.**
    - См. лекцию «Введение в анализ требований к программному продукту».
  - Оценить проект.
  - Составить план проекта.
  - …

Разработка архитектуры…

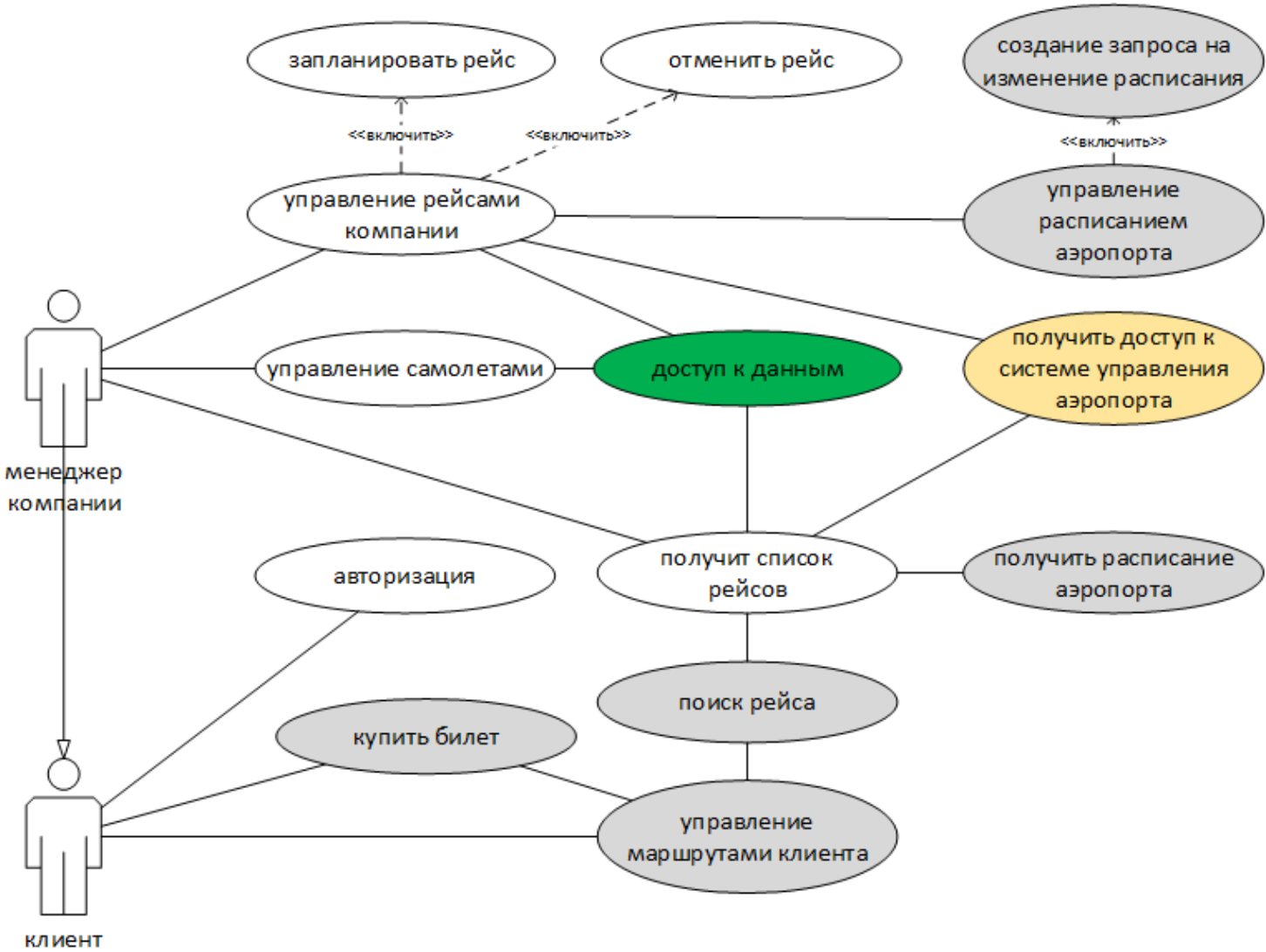
- Какие артефакты получают в результате выделения требований?

Разработка архитектуры

- Какие артефакты получают в результате выделения требований?
  - Документ с описанием требований.
  - Набор UML-диаграмм:
    - Диаграмма прецедентов (use-case).
    - Сценарии.
    - Диаграммы активности.
    - …
  - …

Диаграмма прецедентов

- ***Зеленый*** – модель данных в рамках примера.
- ***Белый*** – уровень бизнес-логики.
- ***Желтый*** – реализуем в виде «заглушек».
- ***Серый*** – не рассматриваем в рамках примера.



Разработка архитектуры…

- Что делать после выделения требований?

Разработка архитектуры…

- Что делать после выделения требований?
  - Необходимо определиться с архитектурным стилем.
- Какие архитектурные стили Вы знаете?

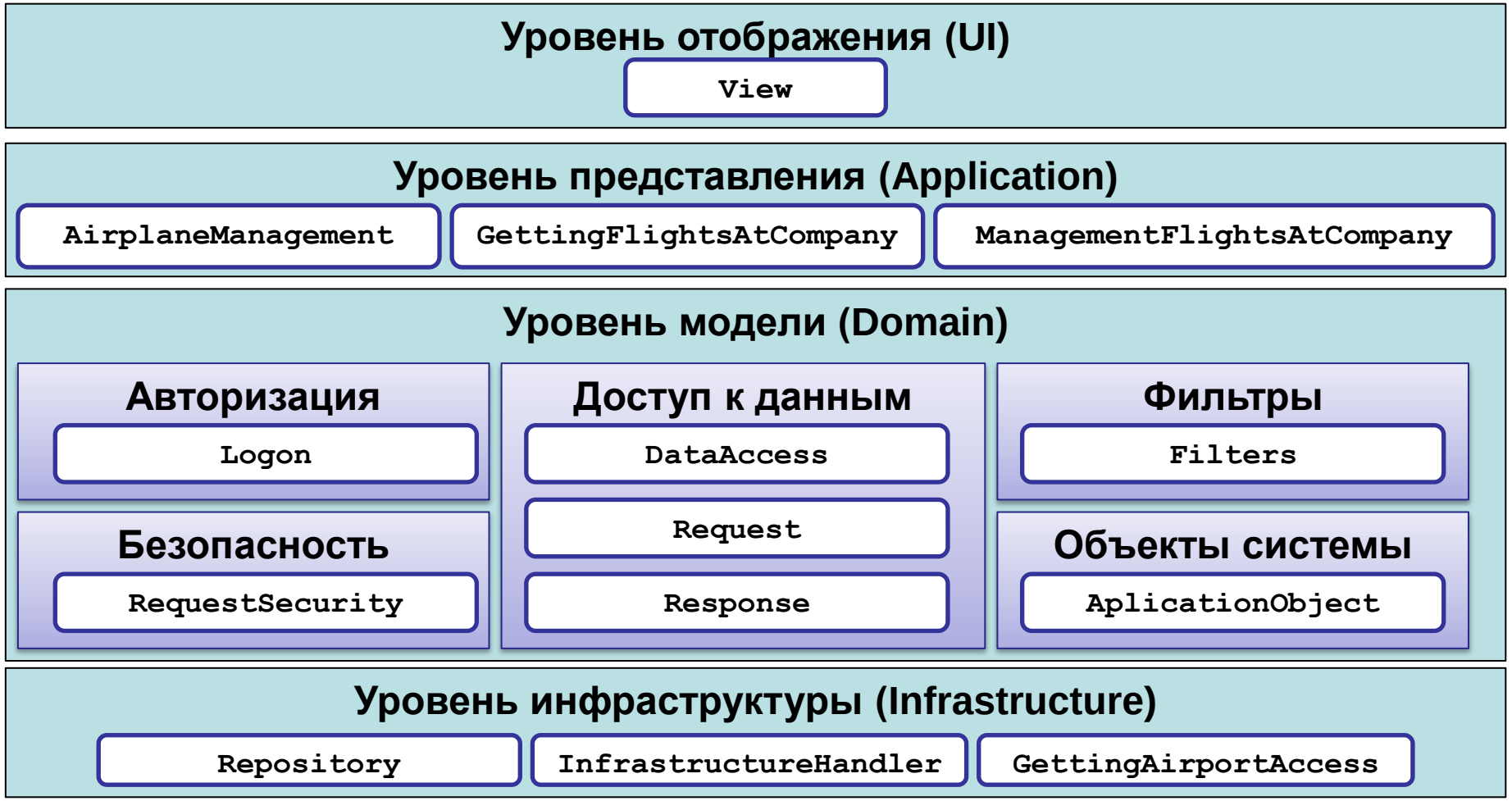
Разработка архитектуры…

- Что делать после выделения требований?
  - Необходимо определится с архитектурным стилем
- Какие архитектурные стили Вы знаете?
  - Архитектура «файл-сервер».
  - Клиент-серверная архитектура.
  - Компонентная архитектура.
  - Архитектура «шина сообщений».
  - Многозвенная архитектура.
  - Объектно-ориентированная архитектура.
  - Сервис-ориентированная архитектура.
  - **Послойная архитектура** (*далее используем*).

Разработка архитектуры

- Также необходимо определиться с набором паттернов, которые будут использоваться при реализации системы.
- В рамках решения будут продемонстрированы:
  - DDD (Domain Driven Design).
  - MVC (Model View Controller).

Послойная архитектура. Схема



Послойная архитектура. Уровни

- ***Уровень отображения*** содержит компоненты пользовательского интерфейса.
- ***Уровень представления*** содержит компоненты помогающие преобразовывать интерфейс модели к интерфейсу используемому в отображениях
- ***Уровень модели*** определяет возможные действия над данными и способ их обработки. Уровень содержит компоненты реализующие всю бизнес-логику разрабатываемой системы.
- ***Уровень инфраструктуры*** содержит компоненты преобразующие данные внешнего мира в объекты используемые в рамках модели.

Послойная архитектура. Подсистемы…

- ***Уровень отображения:***
  - **`View`** – содержит набор классов, отображающих данные в модели.
- ***Уровень представления:***
  - **`AirplaneManagement`** – компонент представляет интерфейс управления самолетами.
  - **`ManagementFlightsAtCompany`** – компонент представляет интерфейс управления рейсами компании (планирование рейсов, отмена и т.д.).
  - **`GettingFlightsAtCompany`** – компонент представляет интерфейс получения списка рейсов.

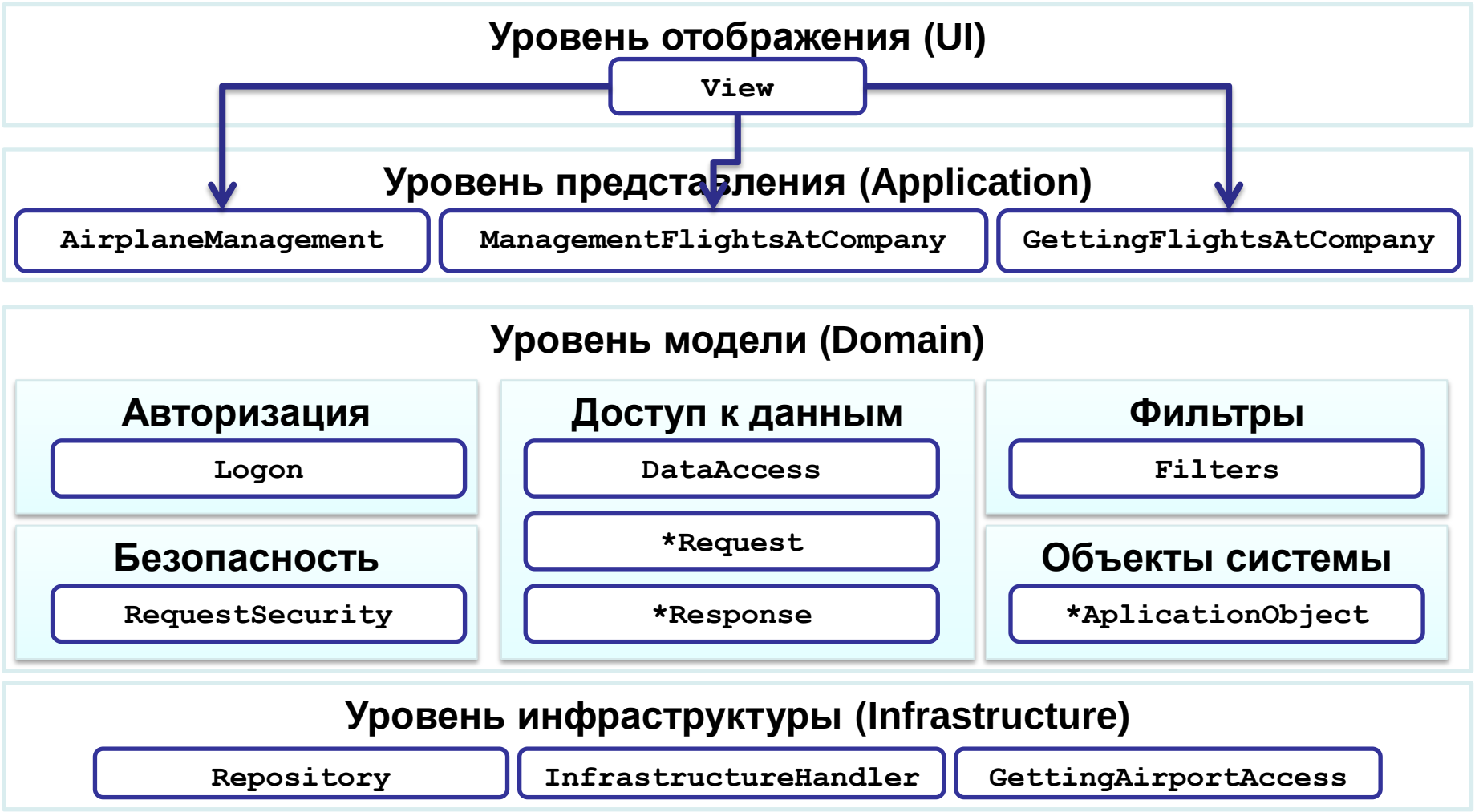
Послойная архитектура. Подсистемы…

- ***Уровень модели:***
  - **`AplicationObject`** – набор интерфейсов объектов, используемых в системе.
  - **`DataAccess`** – компонент доступа к данным.
  - **`Request`** – возможные операции над данными.
  - **`Response`** – классы получения результатов выполнения операций над данными.
  - **`Filters`** – компонент, содержащий функциональность фильтрации данных, например, из списка рейсов выделения рейсов с заданной даты.
  - **`RequestSecurity`** – ограничение возможности выполнения запроса в зависимости от роли.
  - **`Logon`** – авторизация пользователей (роль).

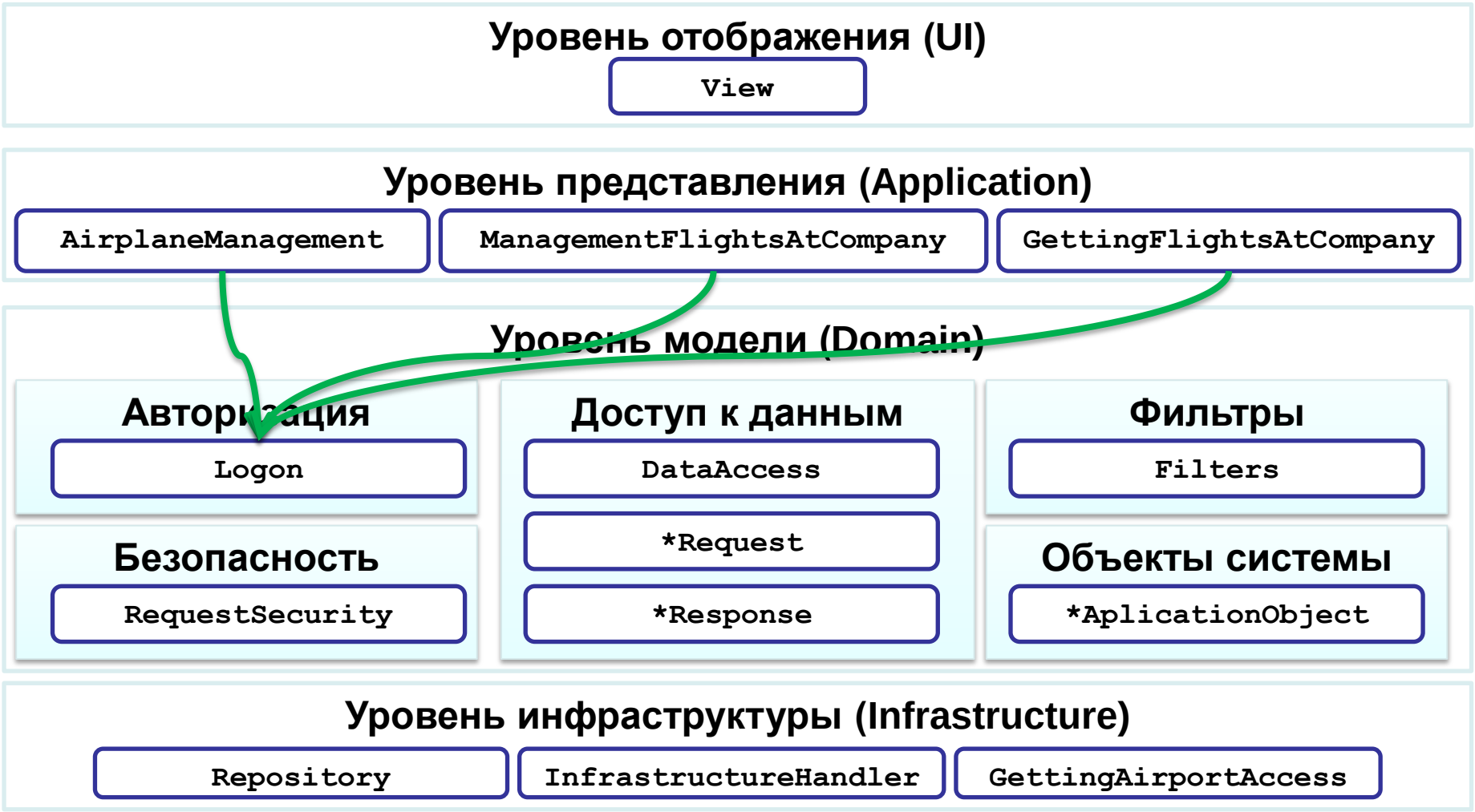
Послойная архитектура. Подсистемы

- ***Уровень инфраструктуры***
  - **`Repository`** – компонент доступа к данным.
  - **`InfrastructureHandler`** – компонент преобразования запросов в операции, используемые в рамках инфраструктуры.
  - **`GettingAirportAccess`** – внешний компонент доступа к данным аэропорта.

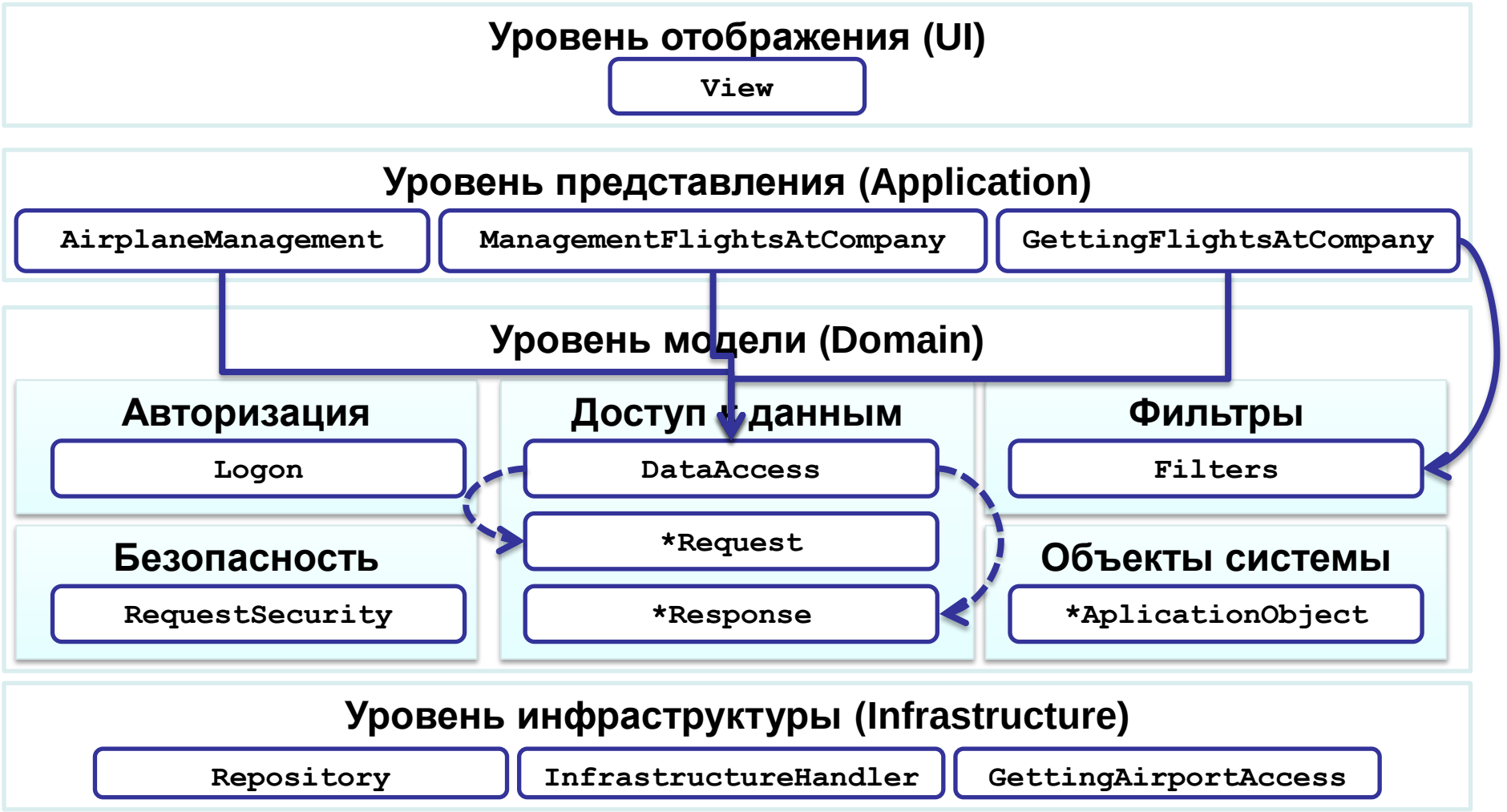
Послойная архитектура. Взаимодействие…



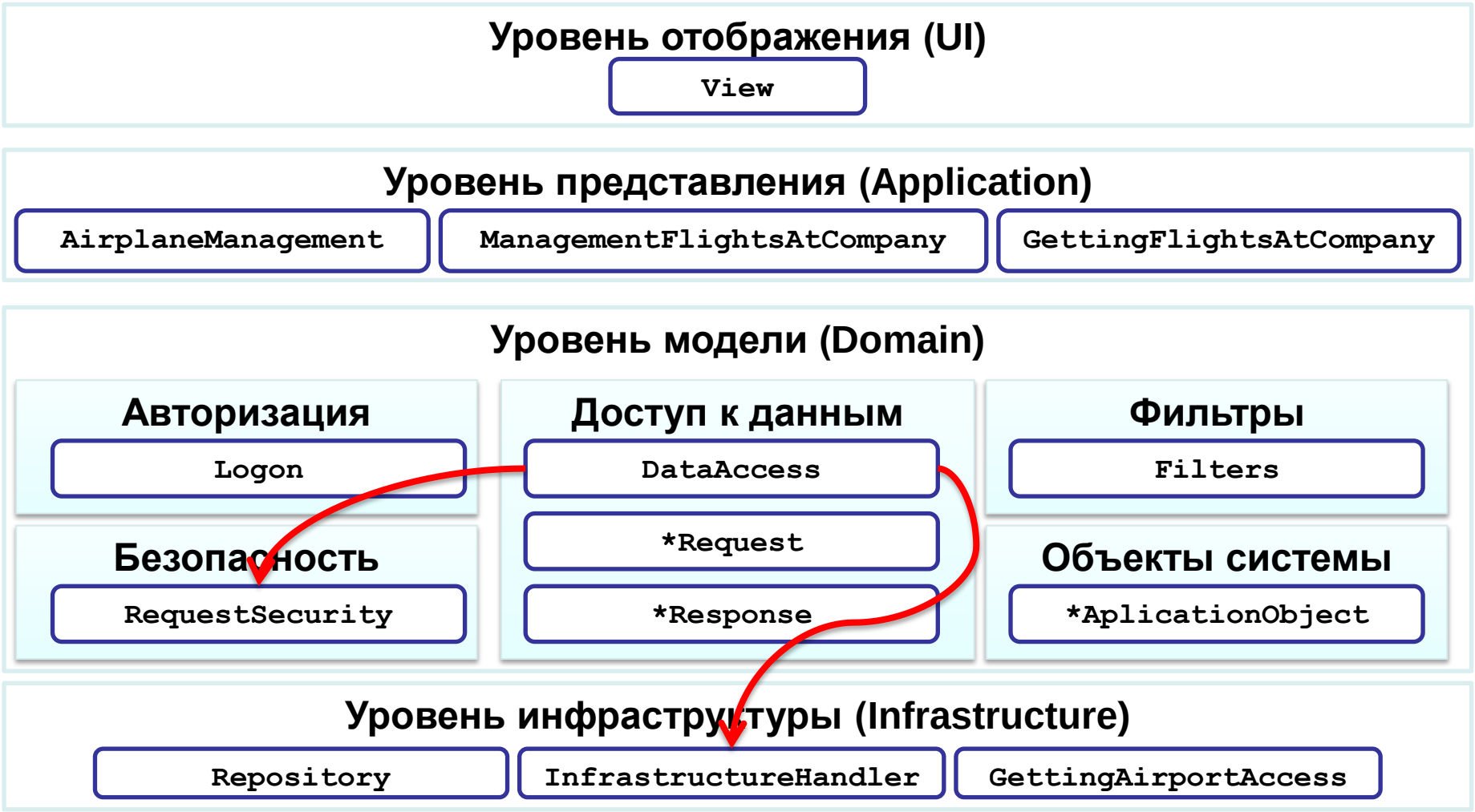
Послойная архитектура. Взаимодействие…



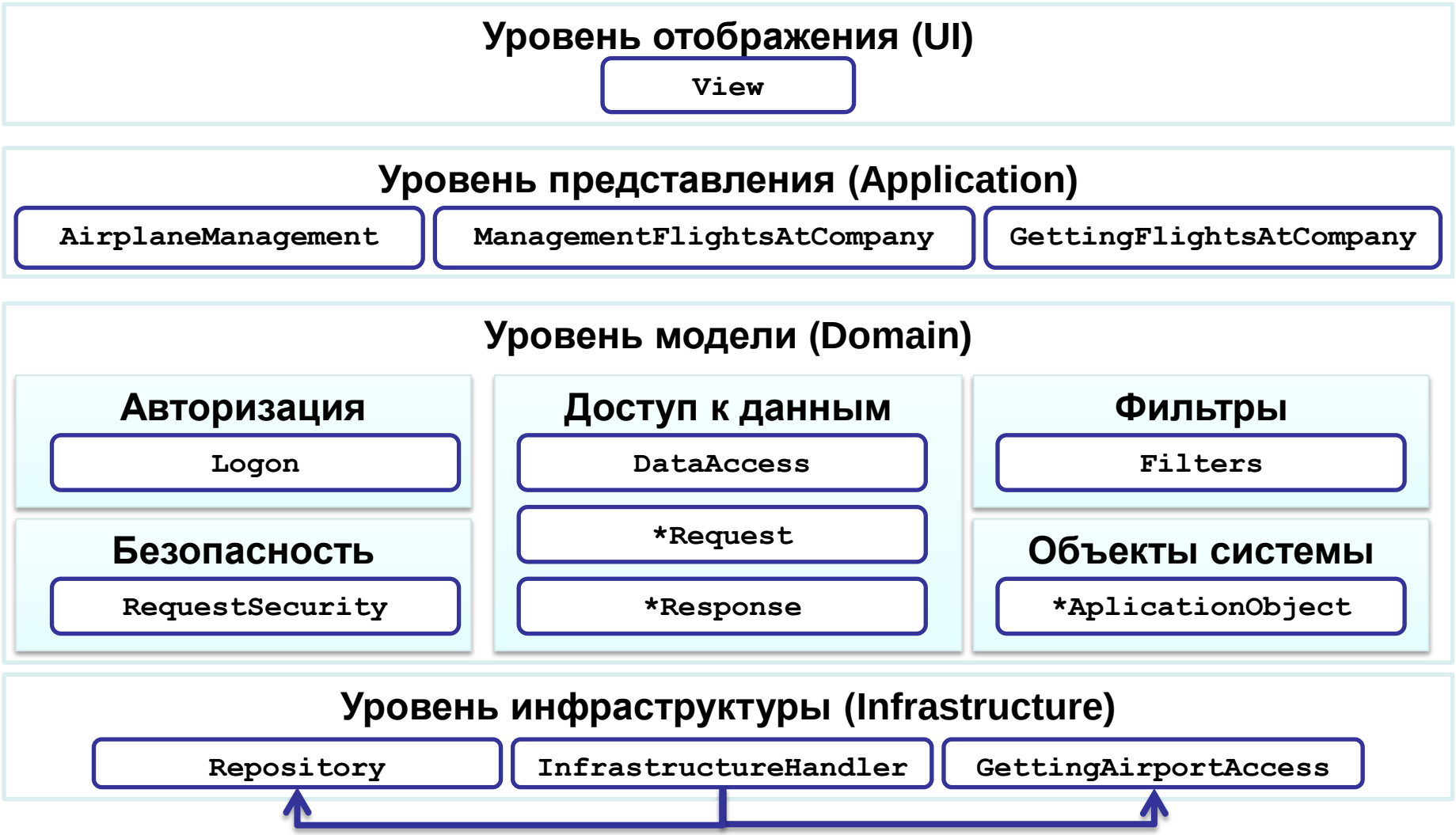
Послойная архитектура. Взаимодействие…



Послойная архитектура. Взаимодействие…

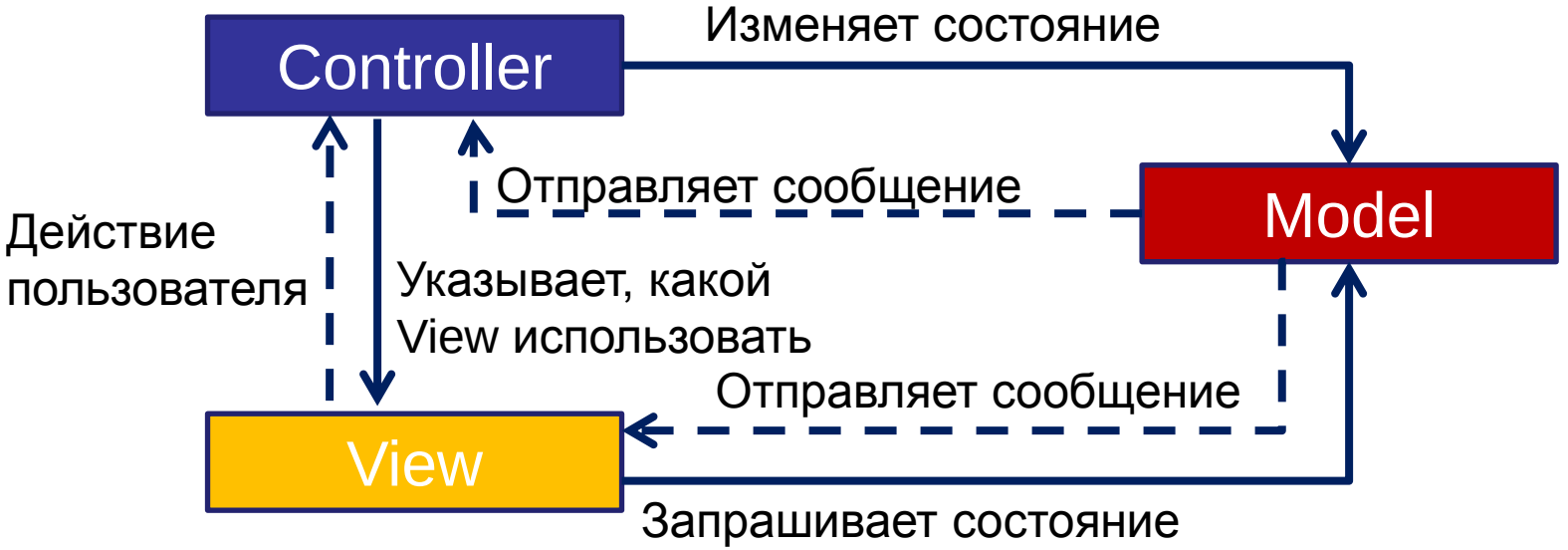


Послойная архитектура. Взаимодействие



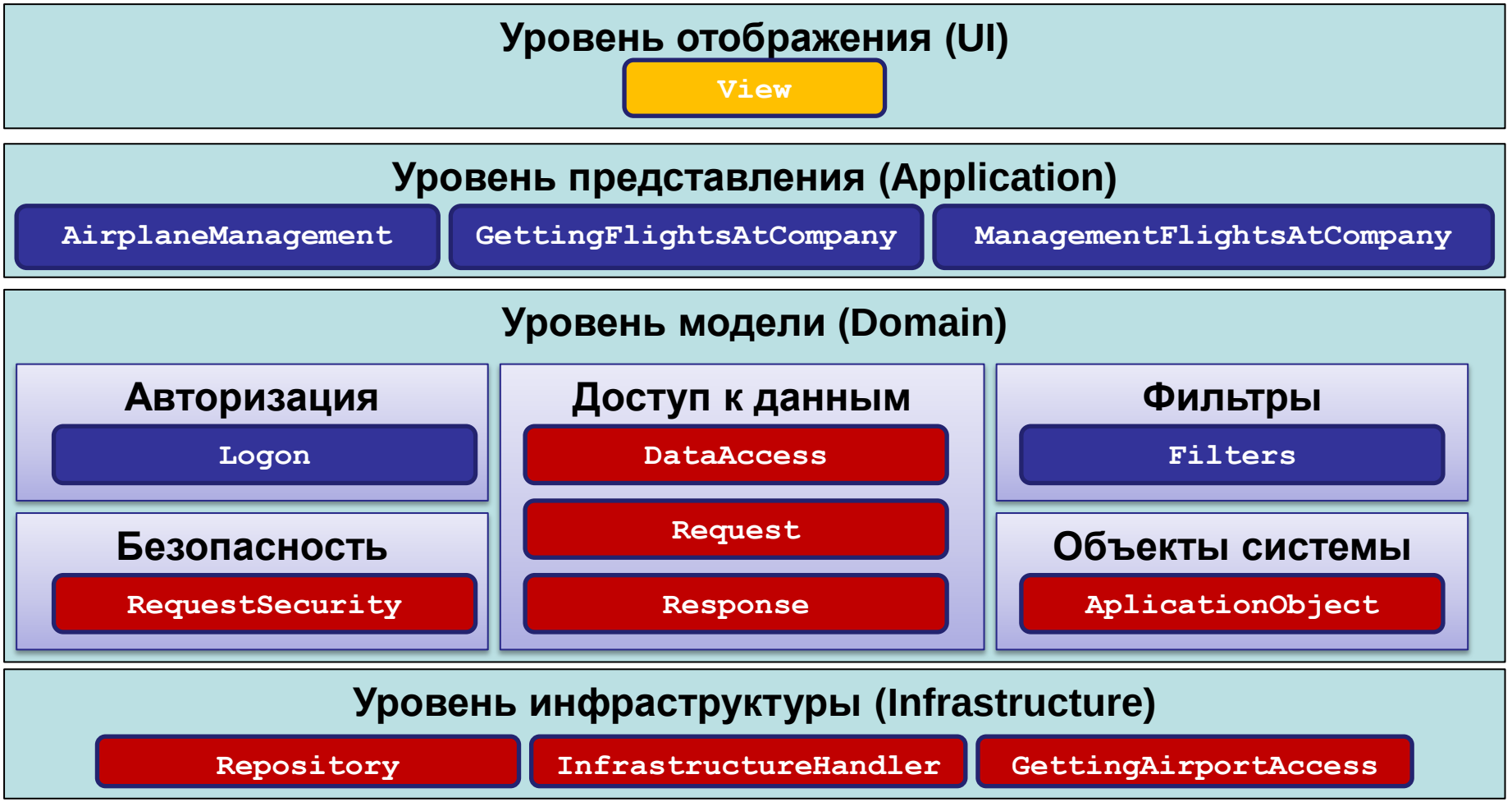
Разработка архитектуры. Паттерн…

- Предполагается, что приложение разработано на основе шаблона проектирования MVC (Model-View-Controller) с рядом упрощений.
  - См. лекцию «Примеры типовых решений. Архитектурные паттерны. Паттерны на примере».



Разработка архитектуры. Паттерн

- Распределение компонент согласно паттерну MVC:



Пример реализации

ПОДСИСТЕМА

Подсистема…

- Что такое подсистема?

Подсистема…

- Что такое подсистема?
  - Это пакет классов с интерфейсом или несколькими интерфейсами.
    - См. лекцию «Проектирование подсистем. Основные принципы объектного подхода».

Подсистема…

- Основное требование к подсистемам?

Подсистема…

- Основное требование к подсистемам?
  - Подсистема должна скрывать реализацию.
  - Доступ к функционалу подсистемы осуществляется через интерфейс.

Подсистема…

- Как создать набор объектов, соответствующих классам подсистемы, если их реализация скрыта?

Подсистема…

- Как создать набор объектов, соответствующих классам подсистемы, если их реализация скрыта?
  - Использовать один из порождающих шаблонов проектирования – ***фабрики***.

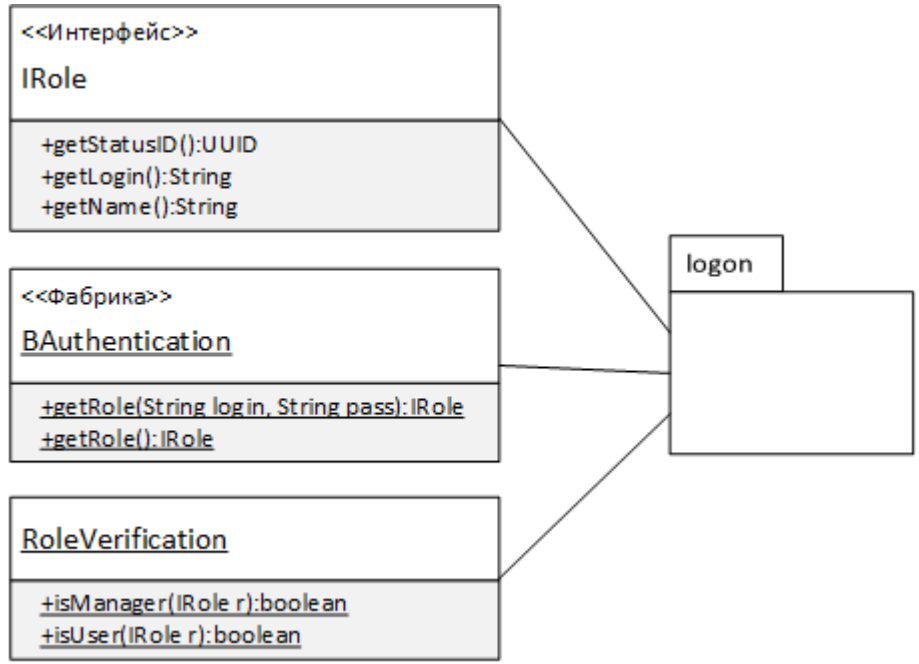
- Зачем нужны фабрики?
  - Фабрики скрывают детали реализации.

Подсистема

- Как заменить реализацию подсистемы?
  - Достаточно заменить «*.dll» или «*.jar» пакет, представляющий подсистему.
  - На работоспособность всей системы замена реализации не должна оказывать влияние.
- Как обеспечить отказоустойчивость при замене подсистем?
  - В требованиях к подсистемам для всех возможных значений параметров интерфейса указываются возможные ответы.
  - При реализации проверка соответствия входных значений выходным является основой для тестирования.

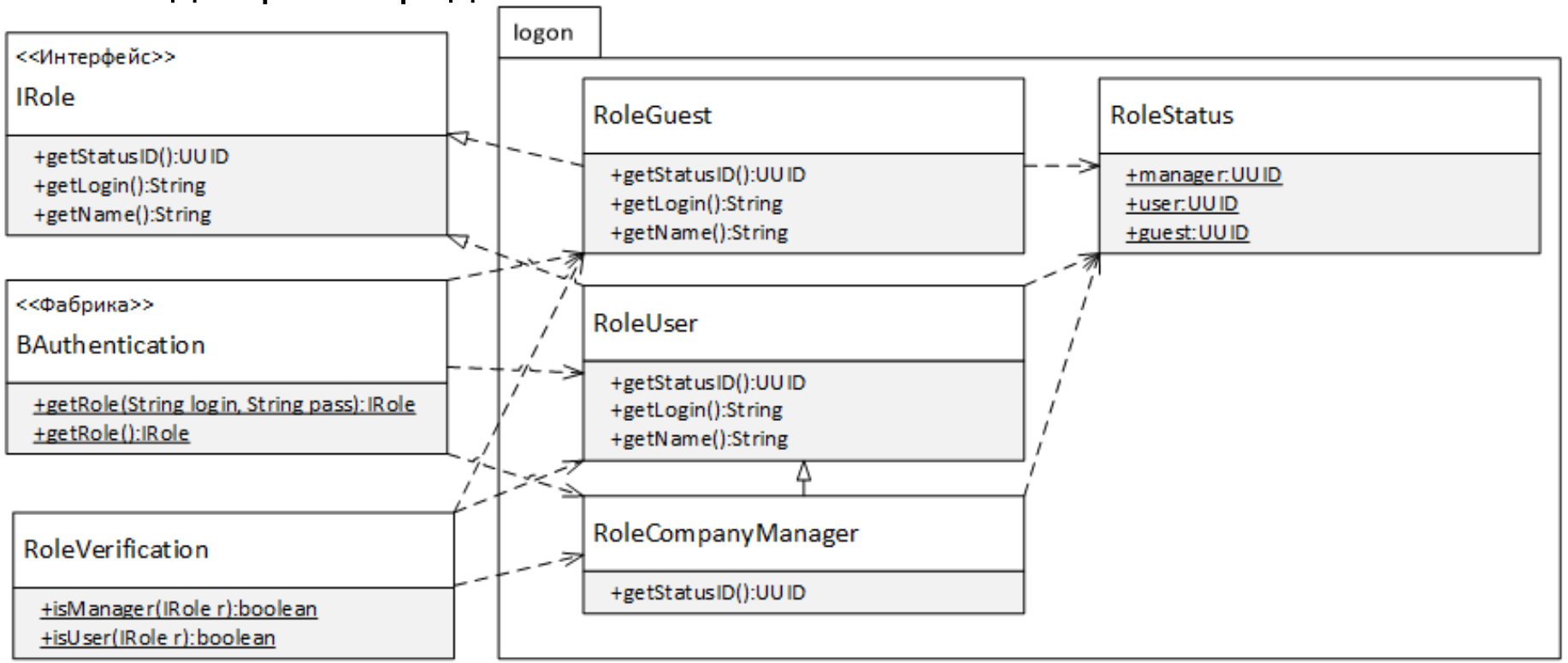
Пример подсистемы авторизации…

- Какой основной объект подсистемы?
  - Роль
    - Роль представлена интерфейсом IRole.
- Подсистема снаружи:
  - Общение через внешний интерфейс.
  - Порождение подсистемы по средствам фабрики.
  - Скрыты детали проверки роли.
  - **Методы статические для уменьшения количества порождаемых объектов.*



Пример подсистемы авторизации

- Состав подсистемы:
  - Есть несколько ролей согласно диаграмме вариантов использования (use-case).
  - Каждая роль представлена своим классом.



Уровень модели

РЕАЛИЗАЦИЯ ПОДСИСТЕМ

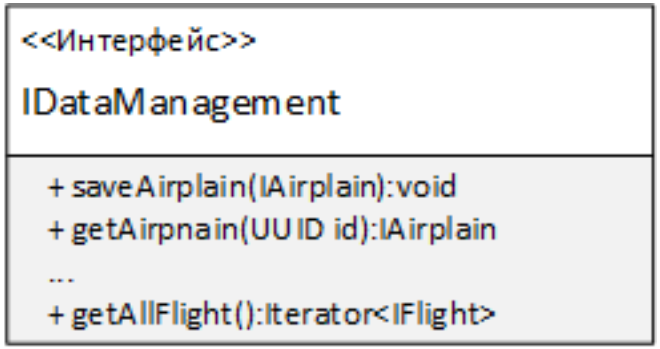
Уровень модели…

- Один из основных уровней в приложении, т.к. уровень представляет возможности будущей системы.
- Уровень должен быть спроектирован так, чтобы по возможности интерфейсы не менялись, но при этом функционал можно было расширять.

- Всегда ли интерфейс может оставаться фиксированным?

Уровень модели…

- Пример, простого проектирования интерфейса:

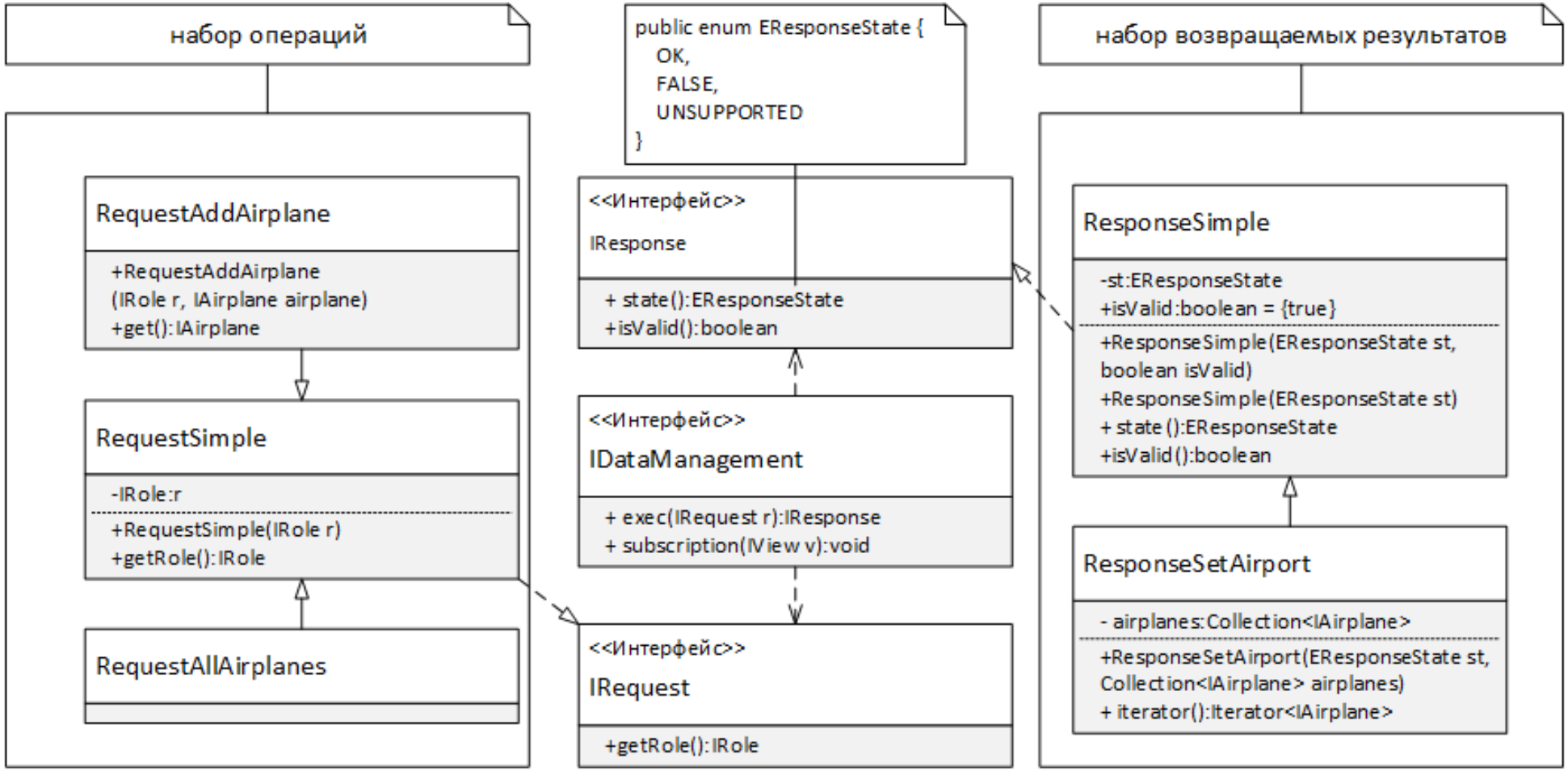


- Чем заключается потенциальный недостаток интерфейса?

Уровень модели…

Уровень модели…

- Пример гибкого интерфейса



Уровень модели…

- Какой интерфейс лучше?

Уровень модели…

- Какой интерфейс лучше?
  - Зависит от приложения и предметной области.
  - Остановимся на гибком интерфейсе.

Уровень модели…

- Проблемы, возникающие при проектировании приложений:
  - Функциональности много.
  - Обработка данных может иметь разную природу.
  - Выполнение операции может затрагивать данные из разных областей приложения.

- Как реализовать необходимую функциональность?

Уровень модели…

- Проблемы, возникающие при проектировании приложений:
  - Функциональности много.
  - Обработка данных может иметь разную природу.
  - Выполнение операции может затрагивать данные из разных областей приложения.

- Как реализовать необходимую функциональность?
  - Всю бизнес-логику нужно разделить на максимально независимую функциональность.

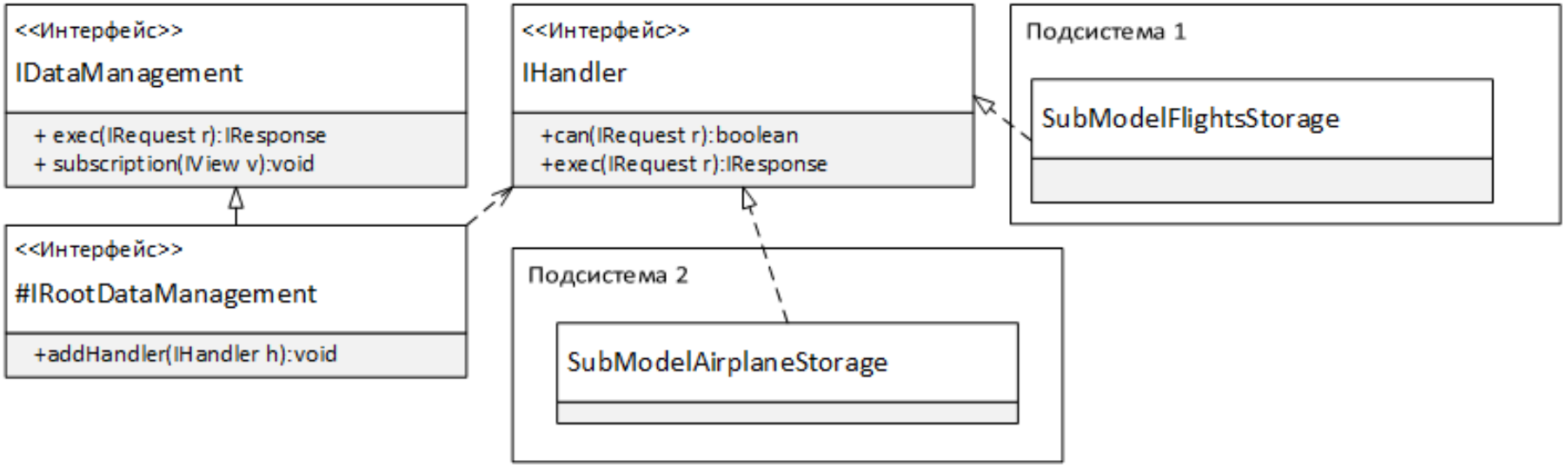
Уровень модели…

- Проблемы, возникающие при проектировании приложений:
  - Функциональности много.
  - Обработка данных может иметь разную природу.
  - Выполнение операции может затрагивать данные из разных областей приложения.

- Как реализовать необходимую функциональность?
  - Каждую функциональность необходимо реализовать в разных компонентах.
  - Для связи функциональности между собой можно ввести специальный класс.

Уровень модели…

- Пример организации распределения обработки по разным компонентам:



- *Замечание: модель содержит метод, позволяющий подписать отображение на изменения модели.*

Уровень модели…

- Какие есть замечания по следующему участку кода?

```
class RootDataManagement implements IRootDataManagement{
@Override
public IResponse exec(IRequest r) {
    if(r.getClass().equals(RequestRefresh.class)) {
        refresh();
        return new ResponseSimple(EResponseState.OK);
    }
    IResponse response = null;
    Iterator<IHandler> itrH = handlers.iterator();
    boolean isValid = true;
    while(itrH.hasNext()) {
        IHandler handler = itrH.next();
        if(handler.can(r)) {
            response = handler.exec(r);
            isValid = isValid && response.isValid();
        }
    }
    if(!isValid) refresh();
    if(response != null) return response;
    return new ResponseSimple(EResponseState.UNSUPPORTED);
}
```

Уровень модели…

- Первая проблема:
  - Нарушены SOLID-принципы.
- Вторая проблема:
  - Один и тот же запрос может быть обработан несколькими обработчиками. Соответственно ответов на запрос может быть несколько.
  - Не возвращаются все результаты, полученные при обработке запроса разными обработчиками.
- Как исправить?

Уровень модели…

- Первая проблема:
  - Нарушены SOLID-принципы.
- Вторая проблема:
  - Один и тот же запрос может быть обработан несколькими обработчиками. Соответственно ответов на запрос может быть несколько.
  - Не возвращаются все результаты, полученные при обработке запроса разными обработчиками.
- Как исправить?
  - Ввести отдельный обработчик для обновления данных.

Уровень модели…

- Первая проблема:
  - Нарушены SOLID-принципы.
- Вторая проблема:
  - Один и тот же запрос может быть обработан несколькими обработчиками. Соответственно ответов на запрос может быть несколько.
  - Не возвращаются все результаты, полученные при обработке запроса разными обработчиками.
- Как исправить?
  - Необходимо ввести специальный ответ, представляющий собой коллекцию результатов.

Уровень модели…

- Как обеспечить безопасность системы в рамках рассматриваемого примера?

Уровень модели…

- Как обеспечить безопасность системы в рамках рассматриваемого примера?
  - Ограничить возможность выполнения запросов.

```
class CanExecute implements ICanExecute{
    ArrayList<Class > guest = new ArrayList();
    ArrayList<Class > user = new ArrayList();
    ArrayList<Class > manager = new ArrayList();

    public CanExecute()
    {
        guest.add(RequestAllAirplanes.class);
        …
        user.addAll(guest);
        …
        manager.add(RequestAddAirplane.class);
```

Уровень модели…

- Как обеспечить безопасность системы в рамках рассматриваемого примера?
  - Ограничить возможность выполнения запросов.

```
@Override
public boolean can(IRequest req) {
    IRole role = req.getRole();
    boolean f = false;

    if(RoleVerification.isManager(role))
    {
        f =  manager.contains(req.getClass());
        if(f) return true;
    }
    …
    return false;
}
```

Уровень модели…

- Как обеспечить безопасность системы в рамках рассматриваемого примера?
  - Ограничить возможность выполнения запросов.

```
class RootDataManagement implements IRootDataManagement{
@Override
public IResponse exec(IRequest r) {
    //Проверка безопастности
    ICanExecute canExecute = BCanExecute.build();
    boolean f = canExecute.can(r);
    if(!f) return new ResponseSimple(EResponseState.FALSE);

    //Выполнение запросов
    …
```

Уровень модели

- Модель должна содержать всю бизнес-логику.
- Даже если функциональность не большая, ее стоит поместить в модель для повторного использования.
- Пример – фильтры.
  - Фильтры только «прореживают» информацию, но реализовывать их каждый раз при обращении к данным не стоит, т.к. это приводит к увеличению кода.

Уровень инфраструктуры

РЕАЛИЗАЦИЯ ПОДСИСТЕМ

Уровень инфраструктуры…

- Основная цель – конвертация объектов системы в представление системы хранения и обратно.
- Рассмотрим конкретные примеры…

Уровень инфраструктуры…

```
public class AddFlightsToStorage {
    public static IResponse add(RequestAddFlights r) {
        if(!RoleVerification.isManager(r.getRole())) {
            //исключение
            return new ResponseSimple(EResponseState.FALSE, false);
        }
        IFlights flights = r.get();
        Connection c = DatabaseConnection.getConnection();
        try {
            Statement stmt = c.createStatement();
            String sql = "INSERT INTO FLIGHTS (ID_FLIGHTS, AIRPLANE, "
                    + " A_FROM, A_TO, TIME_START, TIME_FINISH)"
                    + " VALUES (" + "'"+flights.flightsID().toString()+"',"
                    + "'"+flights.airplane().getID().toString()+"',"
                    + "'"+flights.startAirport().getID().toString()+"',"
                    + "'"+flights.finishAirport().getID().toString()+"',"
                    + "'"+flights.departureTime().getTime()+"',"
                    + "'"+flights.arrivalTime().getTime()+"'" + ")";
            stmt.executeUpdate(sql);
            stmt.close();
            return new ResponseSimple(EResponseState.OK, false);
        } catch (SQLException ex) {}
        return new ResponseSimple(EResponseState.FALSE, false);
    }
}
```

Уровень инфраструктуры

```
public class GetAllFlights {
    public static IResponse get(RequestAllFlights r) {
        Map<UUID, IFlights> flights = new HashMap<>();
        Connection c = DatabaseConnection.getConnection();
        IAirportAccess aa= BAirportAccess.build();
        try {
            Statement stmt = c.createStatement();
            String sql = "SELECT * FROM FLIGHTS";
            ResultSet rs = stmt.executeQuery(sql);
            while(rs.next()) {
                UUID id = UUID.fromString(rs.getString("ID_FLIGHTS"));
                IAirport from = aa.getAccess(null,
                                    UUID.fromString(rs.getString("A_FROM")));
...
                IFlights fl = BFlights.build(...);
                flights.put(id, fl);
            }
            stmt.close();
        } catch (SQLException ex) { }
        return new ResponseSetFlights(EResponseState.OK, flights.values());
    }
}
```

DEMO

- Рассмотрим пример работы кода при наличии только уровней модели и инфраструктуры:
  - См. AddAirplane.java.
  - См. AddFlights.java.

- *Реализация уровня представления рассматривается позже.*

Уровень представления

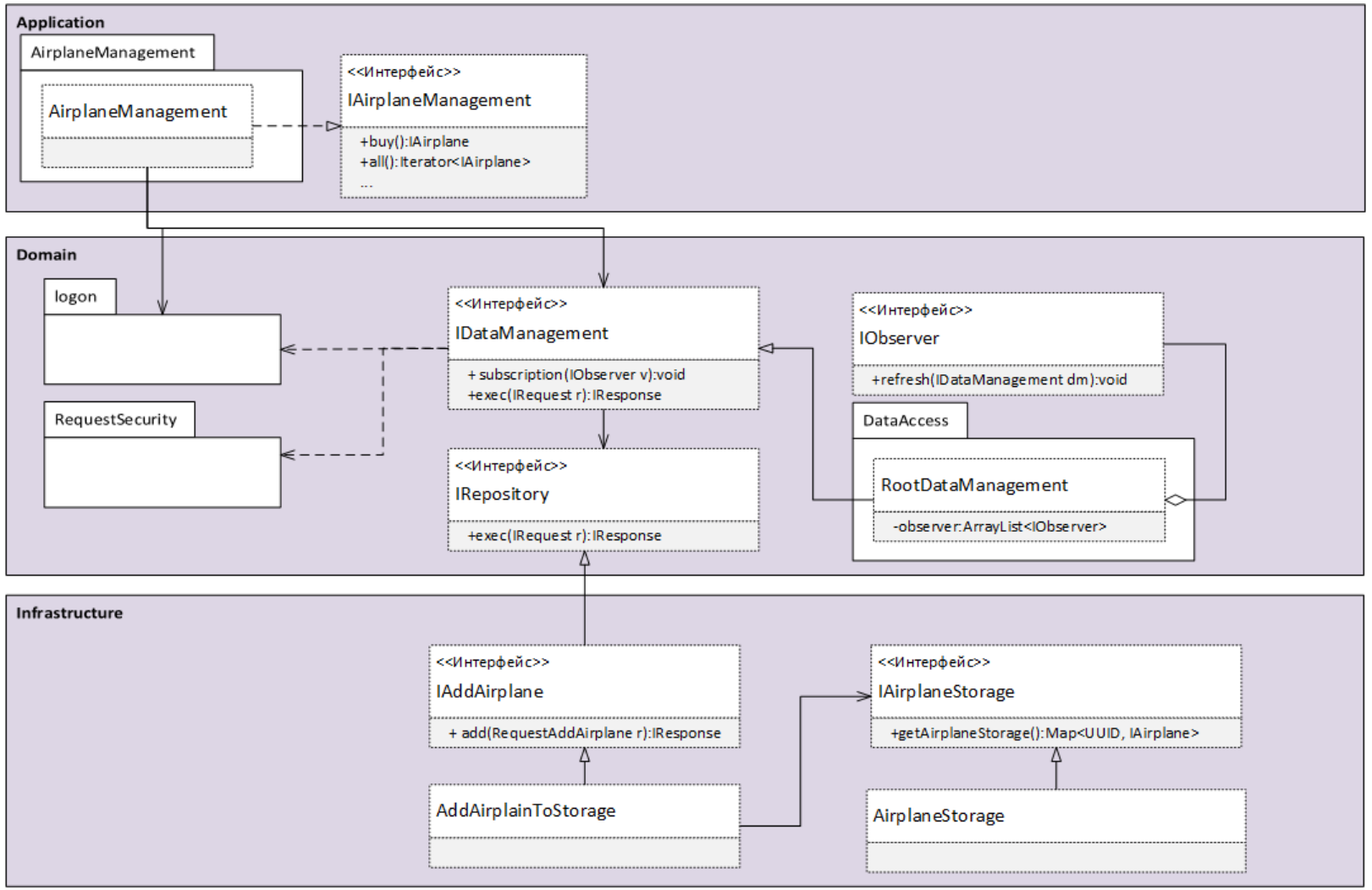
РЕАЛИЗАЦИЯ ПОДСИСТЕМ

Уровень представления…

- Уровень представления позволяет получить более естественный и удобный с точки зрения отображения интерфейс к модели.
- Представление может обращаться к нескольким компонентам модели.
- Ниже представлены несколько интерфейсов уровня приложения:

- Рассмотрим детальную диаграмму на основе компонента управления самолетами.

Уровень представления…



Уровень представления

- Также можно рассмотреть участок кода доступа к рейсам:

```
@Override
public Iterator<IFlight> all() {
    Iterator<IFlight> itr;

    ResponseSetFlights res= (ResponseSetFlights)
            dm.exec(new RequestAllFlights(r));

    itr = res.iterator();
    return itr;
}

@Override
public Iterator<IFlight> since(Date from)
{
    IFilters filter = BFilters.build();
    return filter.flightsSinceDate(all(), from);
}
```

DEMO

- Рассмотрим пример работы кода при наличии уровня представления:
  - См. AddAirplane2.java.
  - См. AddFlights2.java.

- *Реализация отображения рассматривается позже.*

Уровень отображения

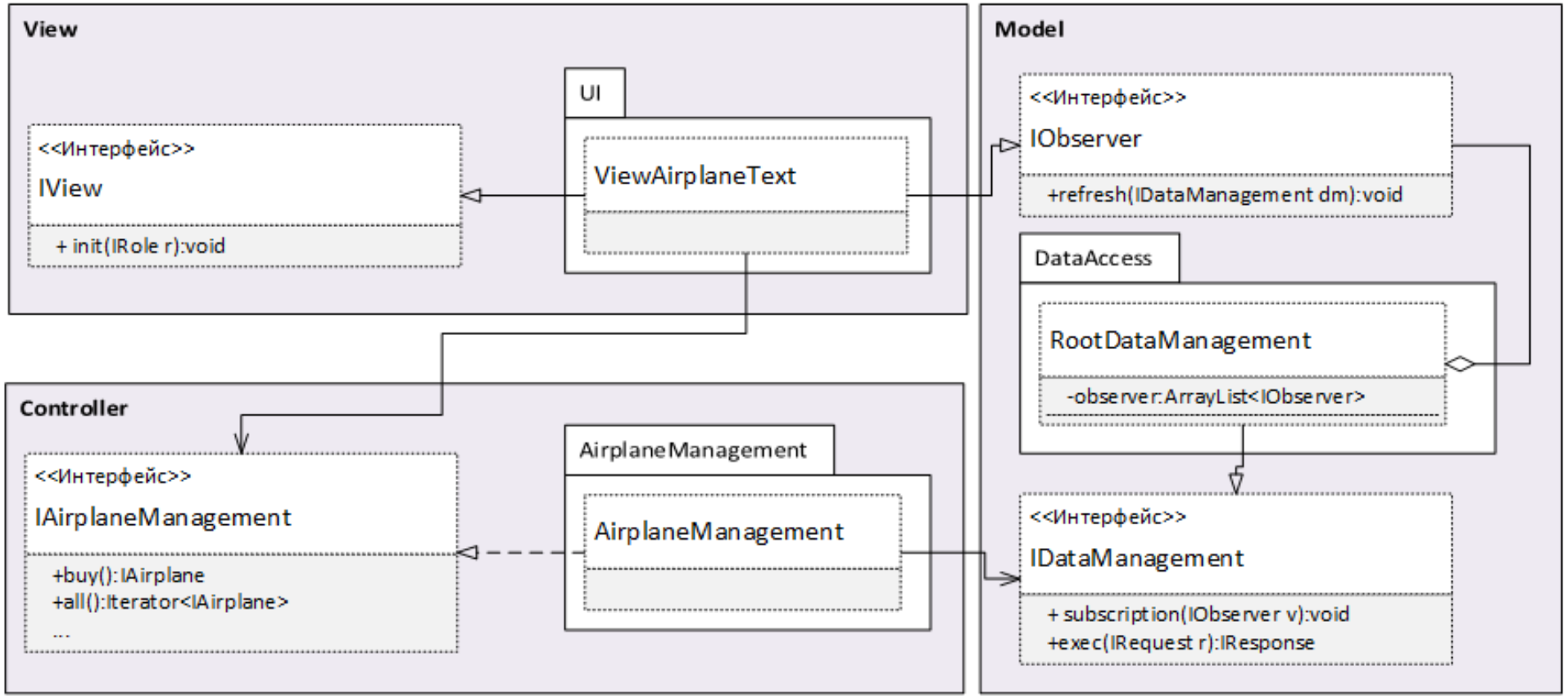
РЕАЛИЗАЦИЯ ПОДСИСТЕМ

Уровень отображения…

- Уровень отображения содержит набор классов, отображающих модель с разной точки зрения и в разных форматах.
- Для доступа к данным используются компоненты уровня представления.
- В рамках примера используется шаблон MVC (при любых изменениях модели все подписанные отображения узнают об изменениях).
- Каждое представление само запрашивает только те данные, которые необходимы. В примере представлены отображения как в консольном виде, так и виде визуальных компонент.
- **Рассмотрим пример текстового отображения.**

Уровень отображения…

- MVC в рассматриваемом примере:



Уровень отображения…

```
public class ViewAirplaneText implements IView, IObserver{

    IRole r = BAuthentication.getRole();
    IAirplaneManagement am;

    public ViewAirplaneText()
    {
        am = BAirplaneManagement.build(r);
    }
    @Override
    public void init(IRole r) {
        this.r = r;
    }
```

…

Уровень отображения…

```
@Override
public void refresh(IDataManagement dm)
{
    Iterator<IAirplane> itr = am.all();
    int i = 1;
    System.out.println("Самолеты компании:");
    while(itr.hasNext())
    {
        System.out.println(i +
                ". (" + itr.next().getID() + ")");
        i++;
    }
    System.out.println();
}
```

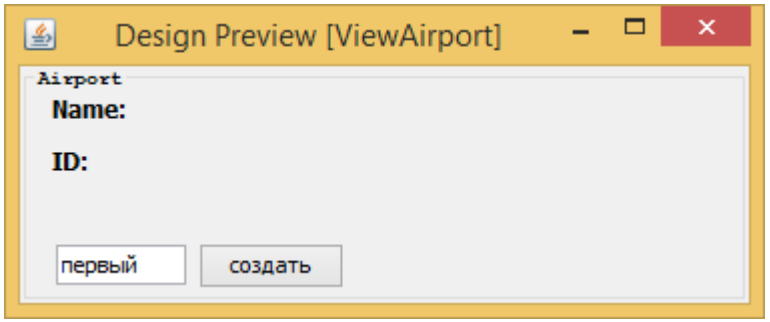
Уровень отображения…

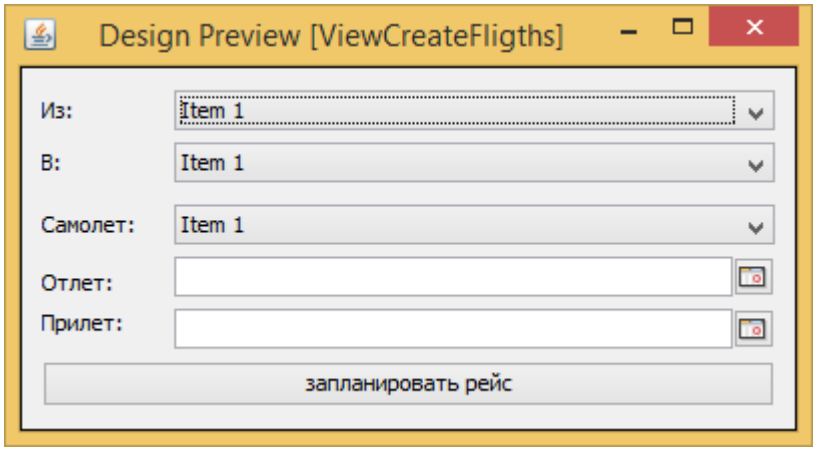
- Пример создания отображения:

```
IRole userRole = BAuthentication.getRole("user", "123");
IDataManagement dt = BDataManagement.build();

IView view = new ViewAirplaneText();
view.init(userRole);

dt.subscription(view);
```

Уровень отображения

- Все приложение в визуальных компонентах редко разрабатывается. Как правило, вначале разрабатываются небольшие пользовательские компоненты, а уже на основе них финальное приложение.
- Пользовательские компоненты позволяют повысить повторное использование кода.
- Примеры визуальных компонент:

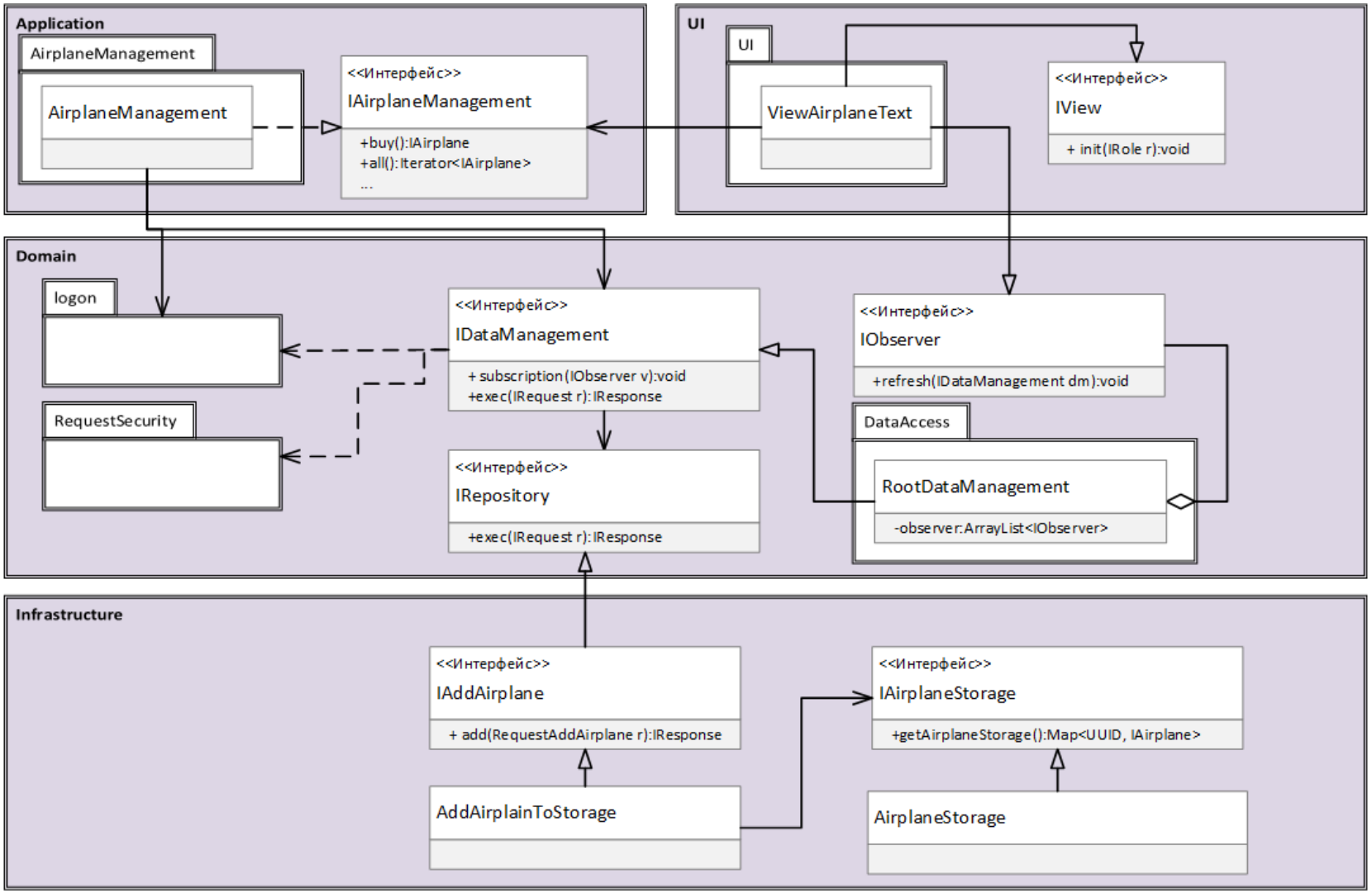




Общий взгляд на приложение. Паттерн DDD

АРХИТЕКТУРА

Общая схема приложения



DEMO

- Рассмотрим итоговое визуальное приложение:
  - См. MainFrame.java.

Заключение

- Рассмотрен пример разработки приложения на основе компонентного подхода.
- Представлены примеры использования ряда шаблонов проектирования.

Контрольные вопросы

- При разработке системы выделяют подсистемы. Затем подсистемы распределяются по слоям. Какая цель преследуется при распределении подсистем по слоям?
- Рассмотрите учебный пример и выделите в нем объекты системы.
- Подумайте, как обеспечить расширяемость модели при неизменности интерфейса?
- За какой функционал отвечает уровень инфраструктуры в рассмотренном примере?
- Для каких целей необходим уровень приложения в рассмотренном примере?
- Для каких целей необходим уровень представления в рассмотренном примере?