Pattern & Practices, Application architecture

Agenda

- 1 OOP Patterns & Practices
- 2 Application & sevice design

Samples of design approaches

1. Object-Oriented Design (Patterns & practices)
2. Service Design (also can be application-monolith)
3. System Design (complex systems with different subdomains)

1. OOP

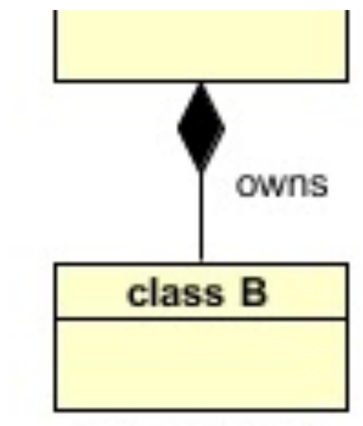
Inheritance

- **Inheritance** can be defined as the **process where one class acquires the properties (methods and fields) of another**. With the use of inheritance the information is made manageable in a hierarchical order.

- The class which inherits the properties of other is known as subclass (derived class, child class) and the class whose properties are inherited is known as superclass (base class, parent class).

Composition

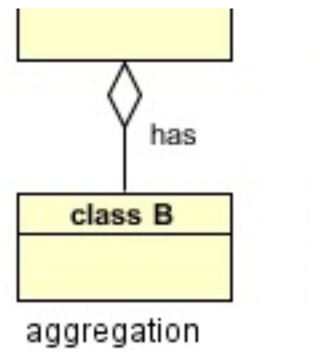
Composition gives us a 'part-of' relationship.



composition

Aggregation

Aggregation gives us a 'has-a' relationship. Within aggregation, the lifetime of the part is not managed by the whole.



______ ______

Usage

Inheritance is when you design your types around what they *are*, and composition/aggregation is when you design types around what they *do*.

If inheritance gives us 'is-a' and composition gives us 'part-of', aggregation gives us a 'has-a' relationship.

(Sample)

Dependency Injection

In software engineering, **dependency injection** is a technique whereby one object (or static method) supplies the **dependencies** of another object. A **dependency** is an object that can be used (a service). An **injection** is the passing of a **dependency** to a dependent object (a client) that would use it.

SOLID

- **S** Single Responsibility (A class should take care of only one responsibility.)
- **O** Open Closed (Extension should be preferred over modification.)
- **L** Liskov Substitution (A parent class object should be able to refer child objects seamlessly during runtime polymorphism, parent should easily replace the child object)
- **I** Interface Segregation (Client should not be forced to use a interface if it does not need it)
- **D** Dependency Inversion (High level modules should not depend on low level modules but should depend on abstraction)

OOP Design Patterns

The *Gang of Four* are the authors of the book, "Design Patterns: Elements of Reusable Object-Oriented Software". This important book describes various development techniques and pitfalls in addition to providing twenty-three object-oriented programming design patterns.

The four authors were Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides.

GoF Groups

- Creational Patterns (5)
- Structural Patterns (7)
- Behavioural Patterns (13)

http://www.blackwasp.co.uk/gofpatterns.aspx

2. Service

Layered Architecture

- 3-Tier Architecture
- N-Tier & N-Layer Architecture
- Onion Architecture

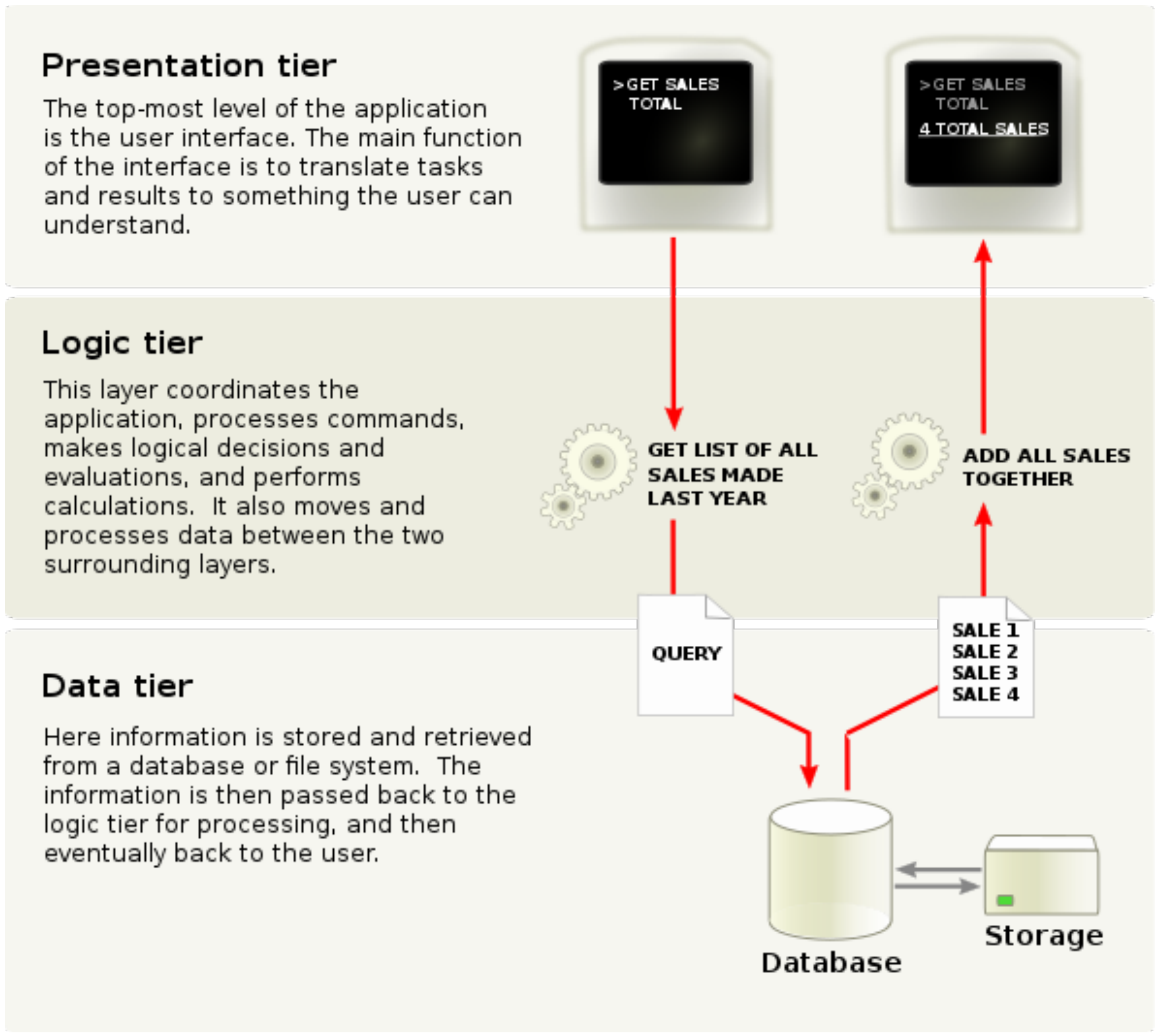
3-Tier Architecture

- **Presentation tier.** This is the topmost level of the application. The presentation tier displays information related to such services as browsing merchandise, purchasing and shopping cart contents. It communicates with other tiers by which it puts out the results to the browser/client tier and all other tiers in the network. In simple terms, it is a layer which users can access directly (such as a web page, or an operating system's GUI).

- **Application tier.** (business logic, logic tier, or middle tier) The logical tier is pulled out from the presentation tier and, as its own layer, it controls an application’s functionality by performing detailed processing.

- **Data tier.** The data tier includes the data persistence mechanisms (database servers, file shares, etc.) and the data access layer that encapsulates the persistence mechanisms and exposes the data. The data access layer should provide an API to the application tier that exposes methods of managing the stored data without exposing or creating dependencies on the data storage mechanisms. Avoiding dependencies on the storage mechanisms allows for updates or changes without the application tier clients being affected by or even aware of the change. As with the separation of any tier, there are costs for implementation and often costs to performance in exchange for improved scalability and maintainability.

3-Tier Architecture



Presentation tier

The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

Logic tier

This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

Data tier

Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.

N-Tier & N-Layer Architecture

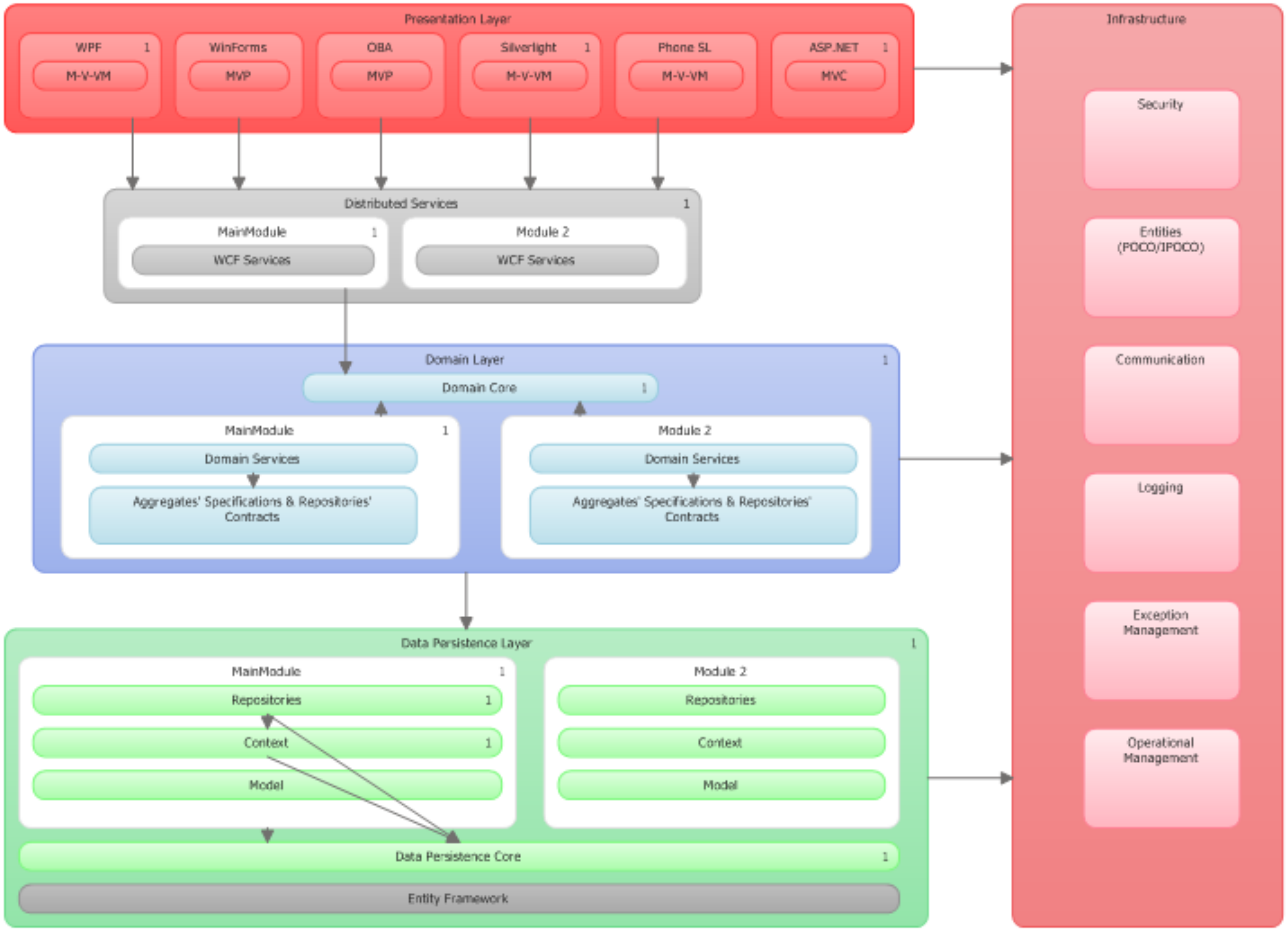
- **Presentation layer** (a.k.a. UI layer, view layer, presentation tier in multitier architecture)

- **Application layer** (a.k.a. service layer or GRASP Controller Layer)

- **Business layer** (a.k.a. business logic layer (BLL), domain layer)

- **Data access layer** (a.k.a. persistence layer, logging, networking, and other services which are required to support a particular business layer)

N-Tier & N-Layer Architecture

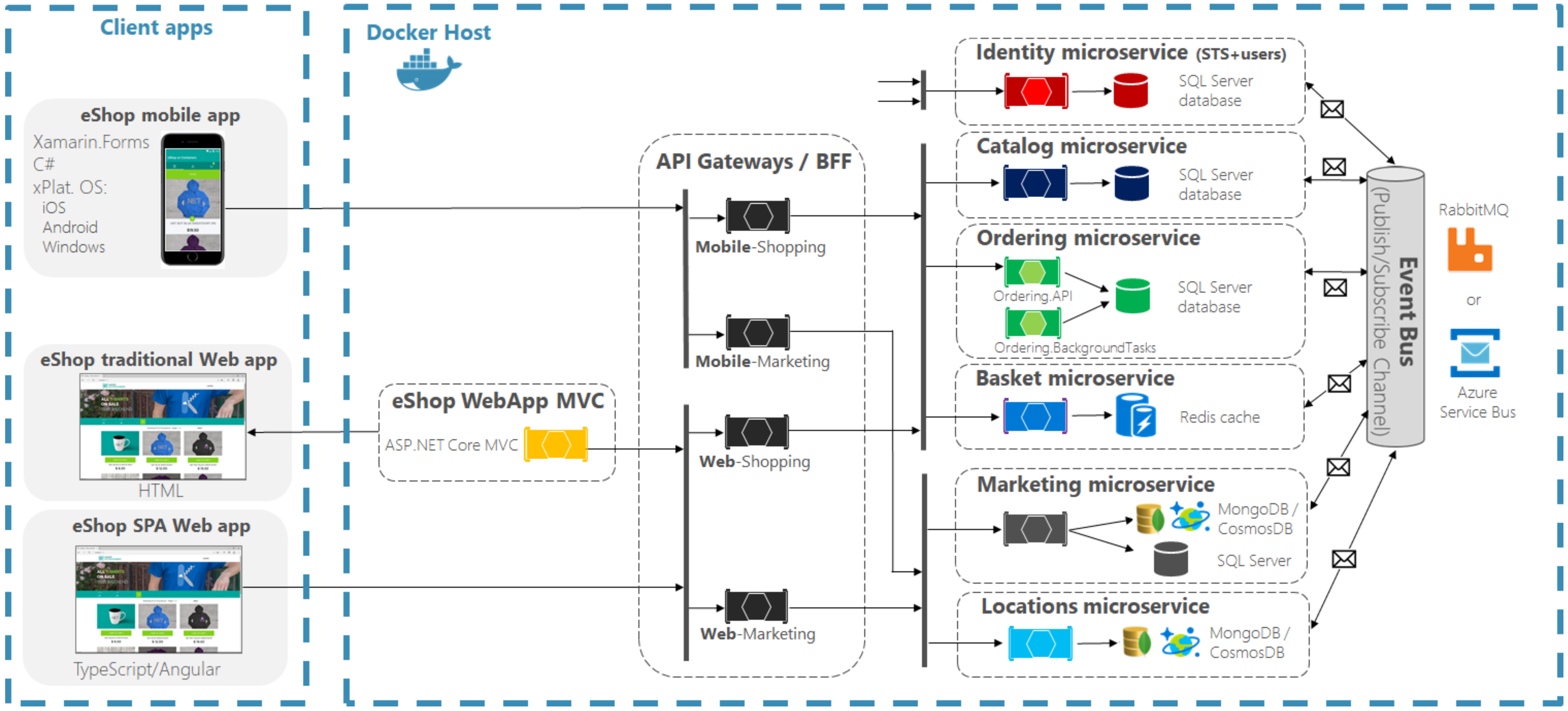


3. System Design

System design questions

- How many services do you have in the system?
- What are theirs responsibility and role?
- Apply design approach for each of them. (services design)
- Describe communication between them. (service-to-service, message broker, shared database as an option (caching))

System Design Sample


Client apps
eShop mobile app
Xamarin.Forms
C#
xPlat. OS:
iOS
Android
Windows
eShop traditional Web app
HTML
eShop SPA Web app
TypeScript/Angular
Docker Host
eShop WebApp MVC
ASP.NET Core MVC
API Gateways / BFF
Mobile-Shopping
Mobile-Marketing
Web-Shopping
Web-Marketing
Identity microservice (STS+users)
SQL Server database
Catalog microservice
SQL Server database
Ordering microservice
Ordering.API
Ordering.BackgroundTasks
SQL Server database
Basket microservice
Redis cache
Marketing microservice
MongoDB / CosmosDB
SQL Server
Locations microservice
MongoDB / CosmosDB
Event Bus
(Publish/Subscribe Channel)
RabbitMQ
or
Azure Service Bus