# System design

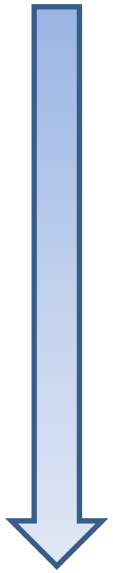
Computing & Information Sciences

W. H. Bell

# Software development lifecycle

## One iteration

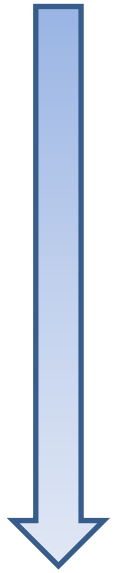
- Requirements definition.
- **Software and systems design.**
- Implementation and unit testing.
- Integration and system testing.
- Operation and maintenance.



# System description

University of Strathclyde Glasgow

- User interface design.
  - Final design loosely coupled to framework choice.
- Data model.
  - Final implementation coupled to data flow/serialisation choices.
- Architecture.
  - Affects how software is implemented.
- Describing functionality.
  - Software agnostic, but expressed within architecture.



# Data model

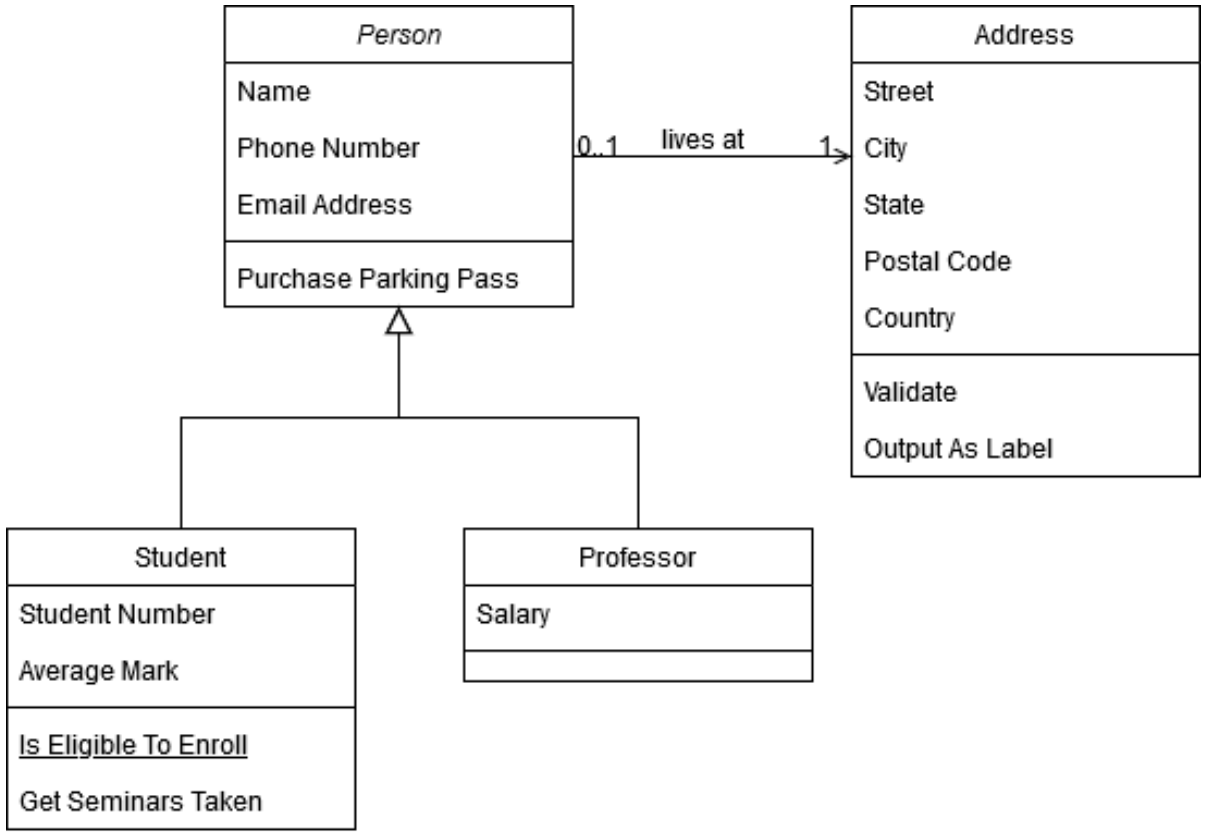
- Each application has a series of data models.
  - Data that are associated with user interface.
  - Data exchange format between services.
  - Data that are associated with key APIs.
  - Data serialisation format.
    - Transient derived data.
- Document data model before implementation.
  - Subset of final data model.

# User interface data model

University of Strathclyde Glasgow

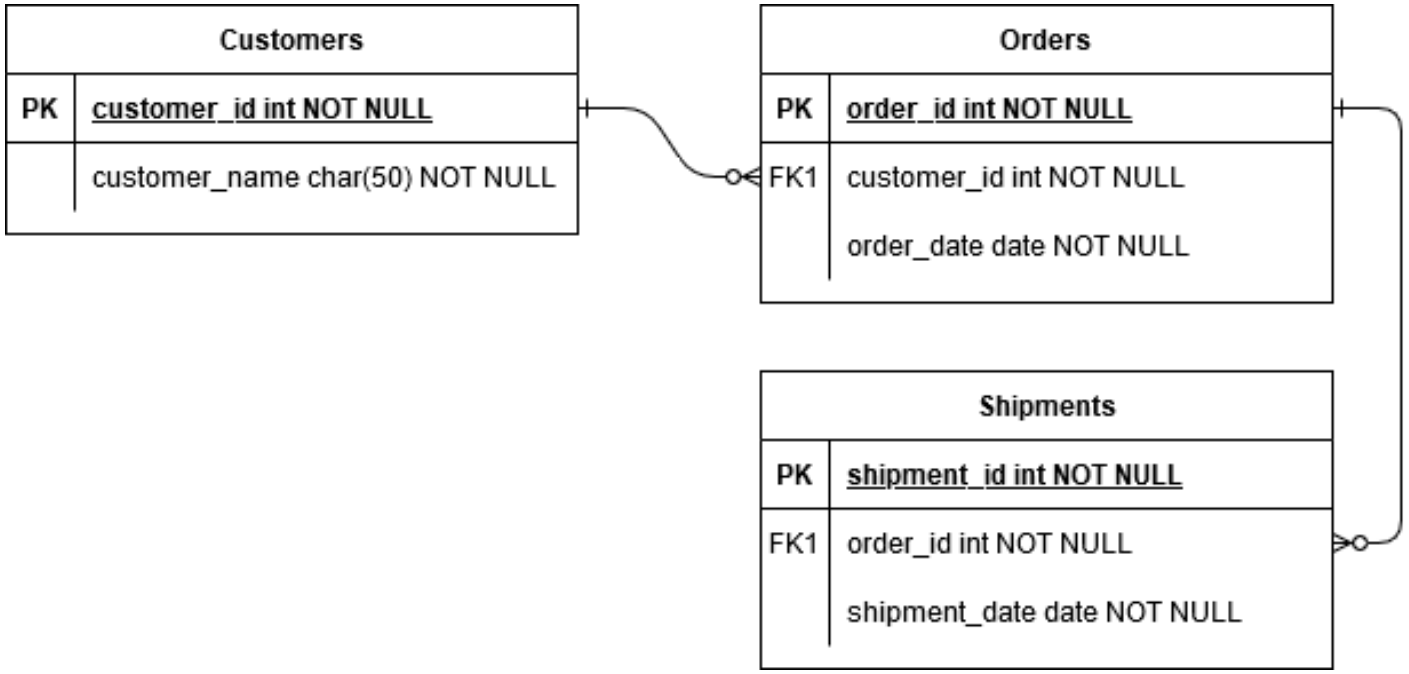
- Capture during user interface design discussions.
  - Type of data.
  - Value limits.
  - Allow Null.
  - Input or output.
  - Association with other data or state in the user interface.
- Create relational entities or class diagram.
  - Share model with client during user interface discussion.

# Class diagram




https://app.diagrams.net/

# Entity diagram




https://app.diagrams.net/

# Data model interviews

## Iterative process, requiring several consultations.

- Interview technical expert stakeholders.
  - Provide initial understanding of data model.
  - Discuss and verify relationships.
- Create data model example with input data.
  - Prototype without software.
  - Demonstrate classes using JSON or tables in Excel.
  - Verify normalisation and relationships between data.
- Need to fix price using complexity of data structure.

# Data model implementation

University of Strathclyde Glasgow

- Connect storage to user interface.
  - Features may require other transient objects/structures.
- Minimise complexity of data model where possible.
  - Use common structures.
- Use automatically generated documentation.
  - Part of developer/maintained documentation.
  - Do not need to include all transient objects in initial design.


# Architecture design patterns

# Architecture design patterns

University of Strathclyde Glasgow

- Structure software around an architecture.
  - Architecture dictates how functions or services interact.
  - May be dictated by software framework.

# Architecture design patterns

University of Strathclyde Glasgow

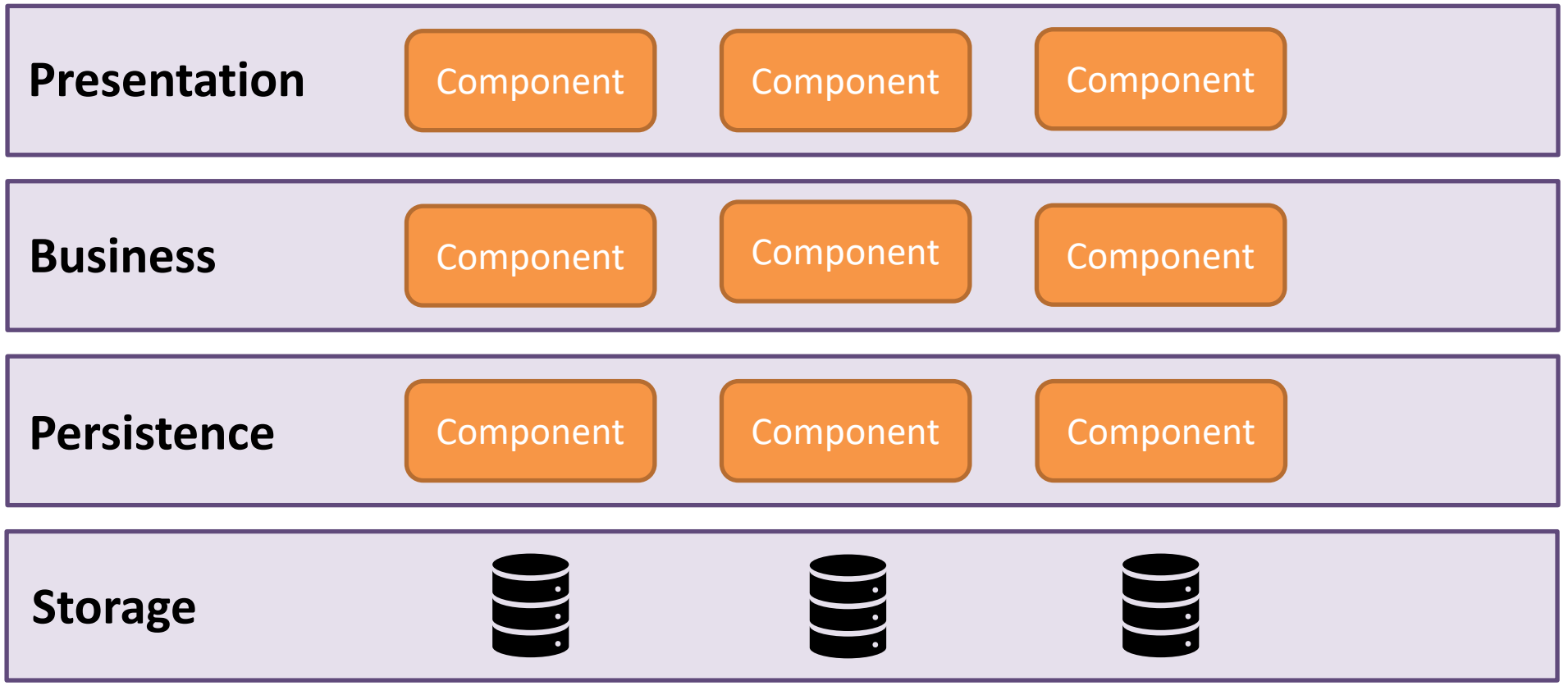
- Benefits.
  - Easy to see the overall structure and expand.
  - Easier to perform integration testing.
- Warnings.
  - Bad architecture choice can cause a project to fail.
  - Software may respond too slowly or be overwhelmed easily.

# Layered

University of Strathclyde Glasgow

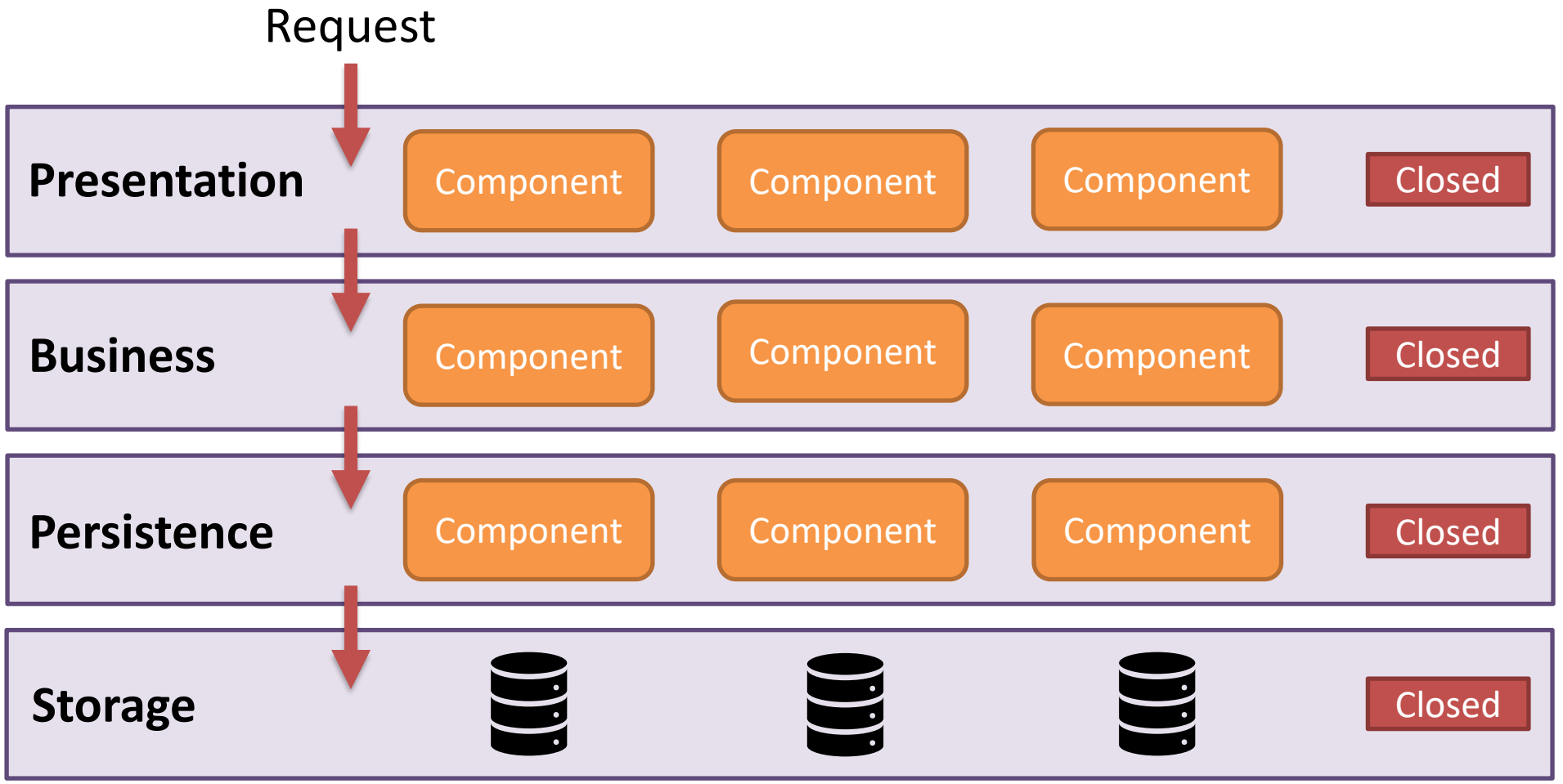
- Separate application into functionality layers.
  - Higher layer can send request and data to another.
  - Lower layer responds with data or confirmation.

# Layered



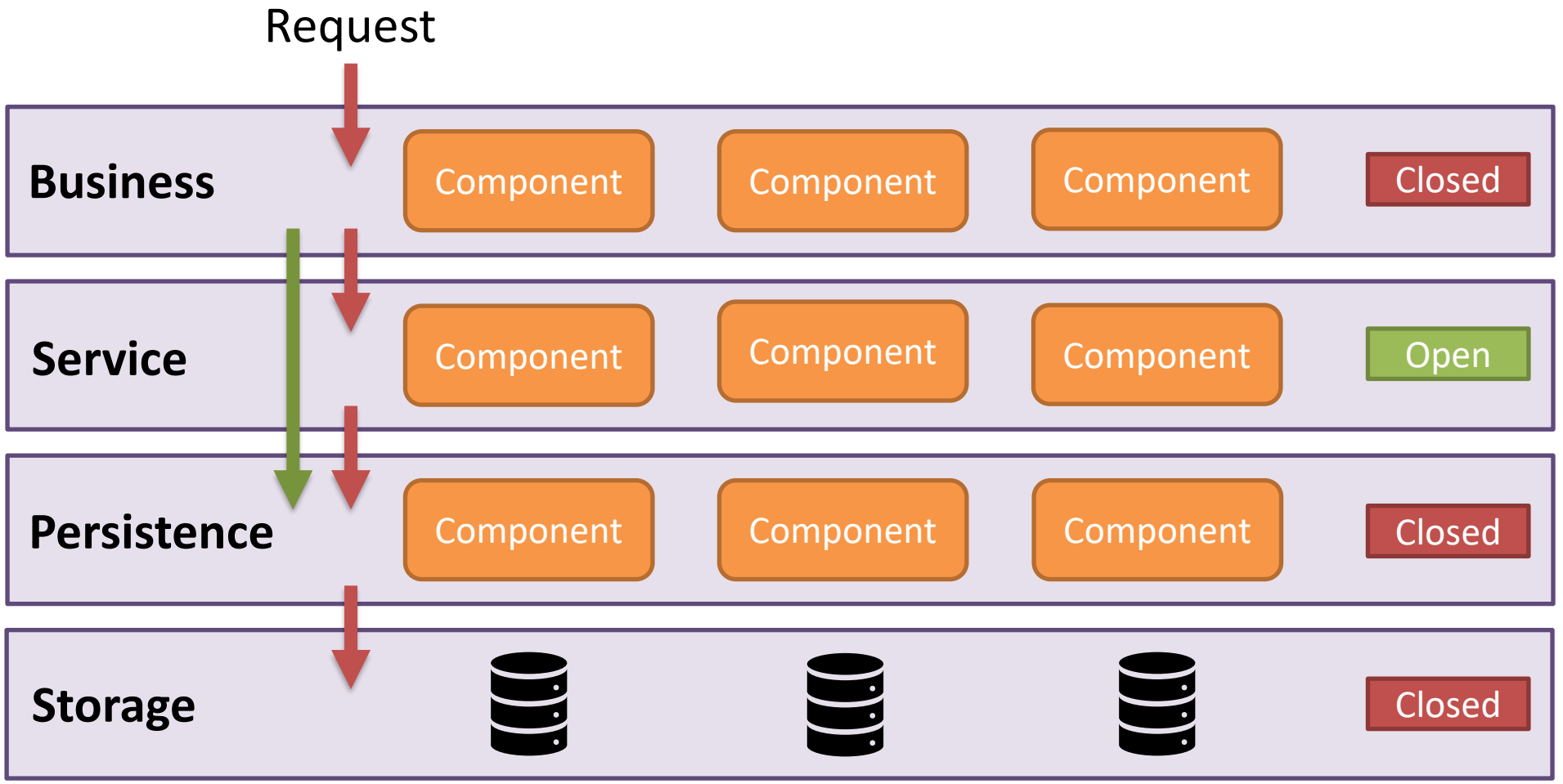

M. Richards, “Software Architecture Patterns”, Report, O'Reilly, 2017.

# Layered




M. Richards, “Software Architecture Patterns”, Report, O'Reilly, 2017.

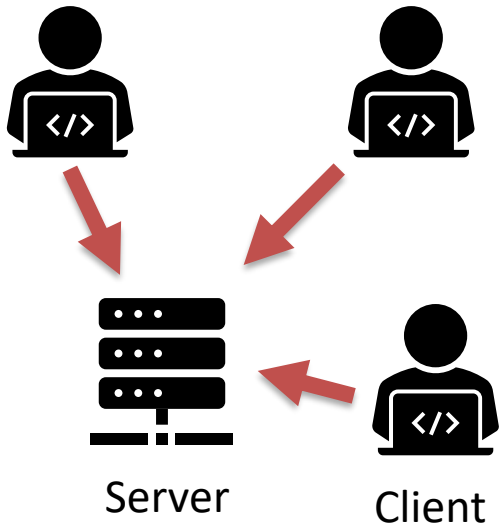
# Layered



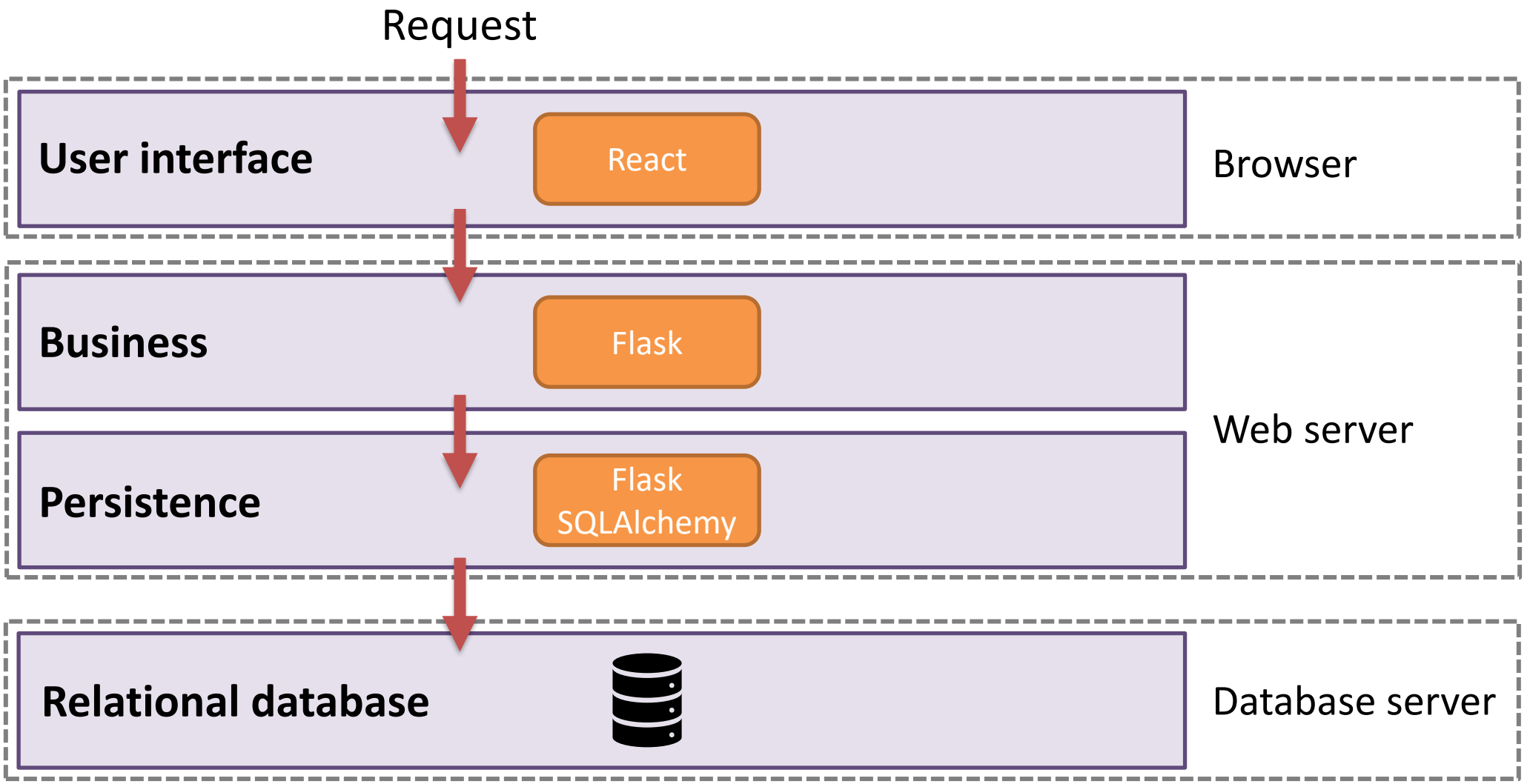

M. Richards, “Software Architecture Patterns”, Report, O'Reilly, 2017.

# Client server

- Separate functionality into client and server.
  - Computer and file server.
  - Email client and email server.
  - Web browser and web server.
- Server listens for connection on port.
  - One thread per client connection.
  - Thread pool and recycling needed.
  - Timeout and close dropped connections.



# Web application



# Publish subscribe

University of Strathclyde Glasgow

- Services publish or subscribe to messages.
  - Small amount of data in a message.
  - Larger data retrieved through direct requests.
- Cache messages within a message bus.
  - Prevent messages being lost when subscriber is not reading.
  - Subscriber might be overloaded or rebooting.
- Larger services or low-level logging applications.

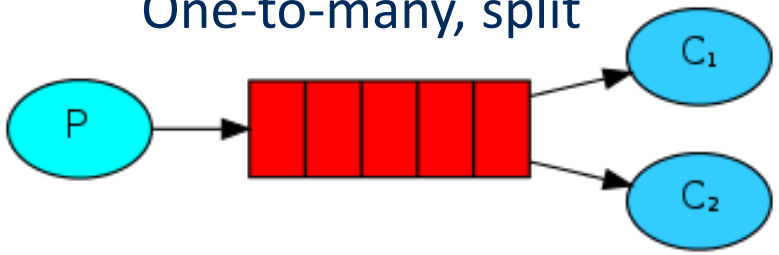
# RabbitMQ

https://www.rabbitmq.com/getstarted.html

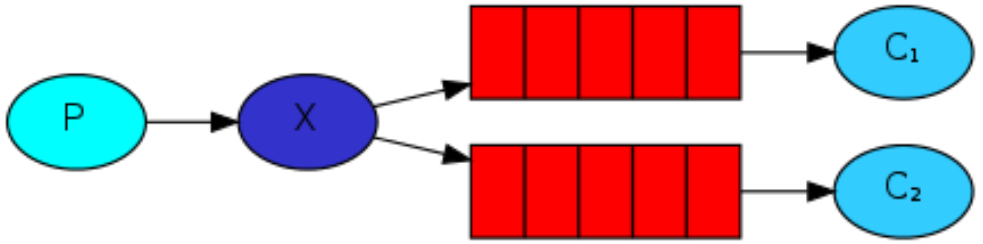
- Highly configurable routing.

One-to-one

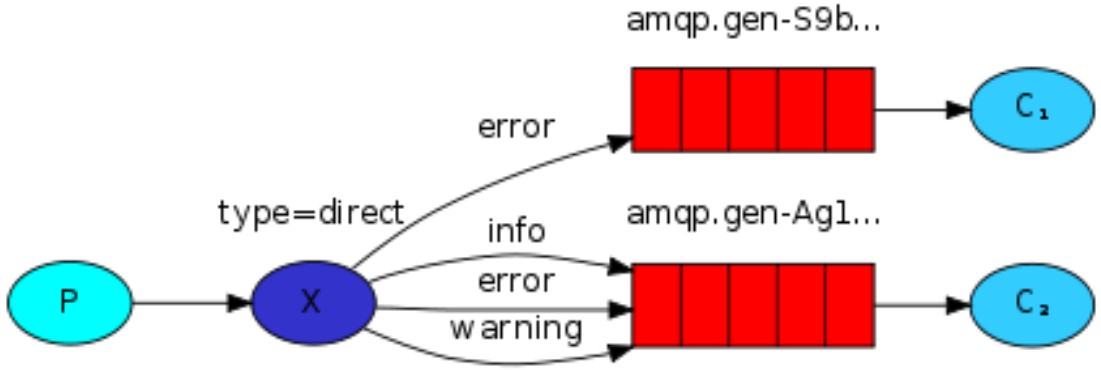

One-to-many, split




One to many, share




One to many, share and filtering

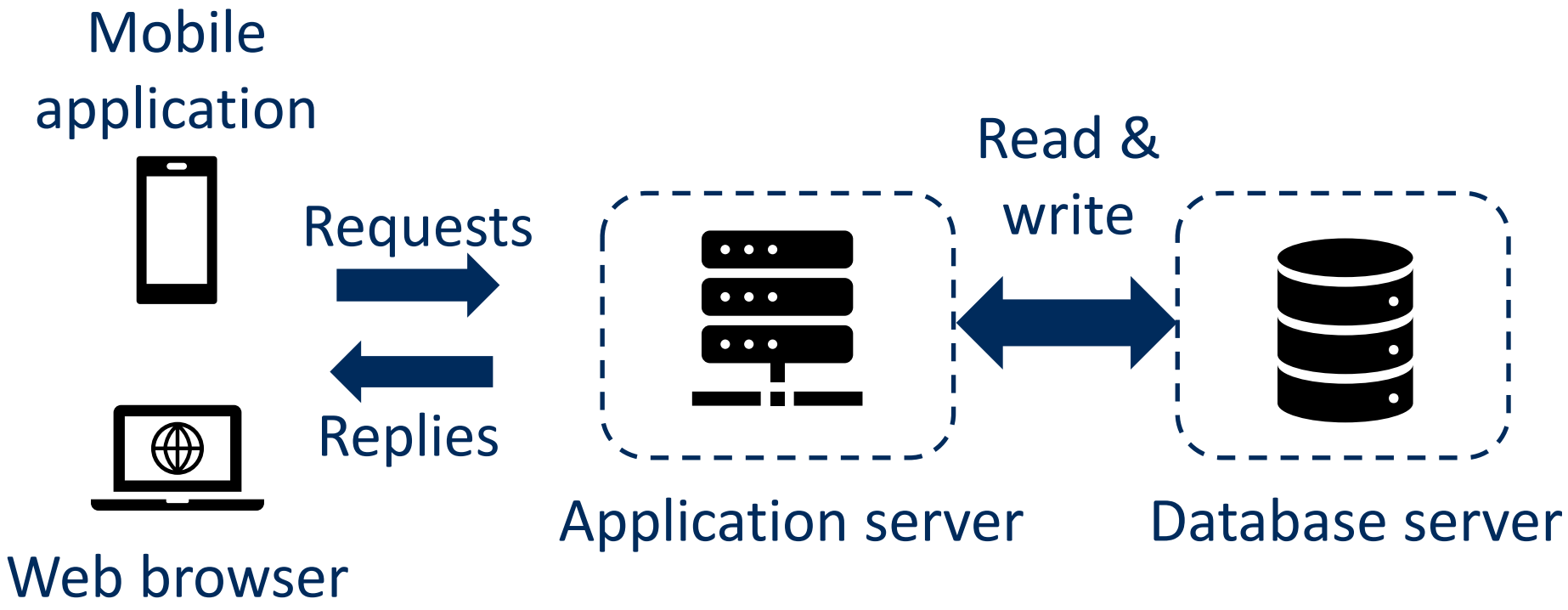


# Microservices

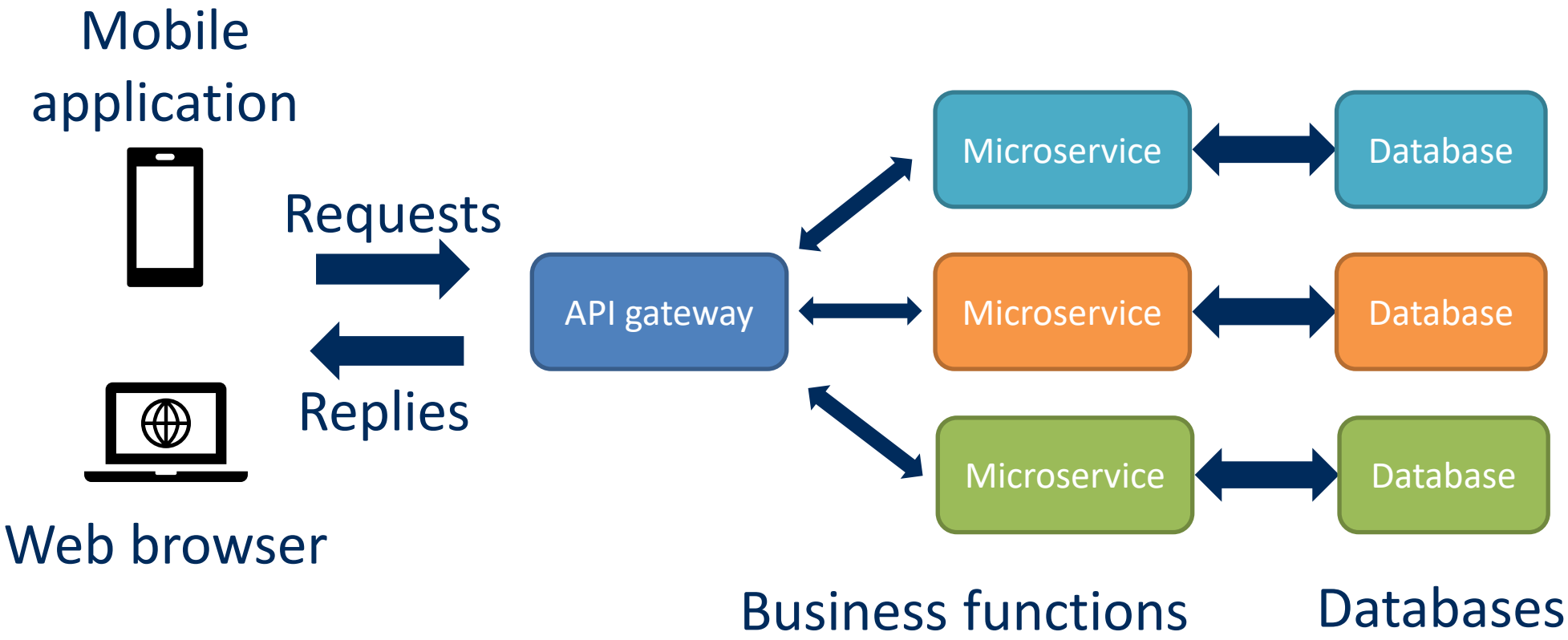
University of Strathclyde Glasgow

- Large web services become less efficient.
  - Cannot duplicate service to allow horizontal scaling.
  - More difficult to develop due to testing and dependencies.
- Fragment large web service into smaller services.

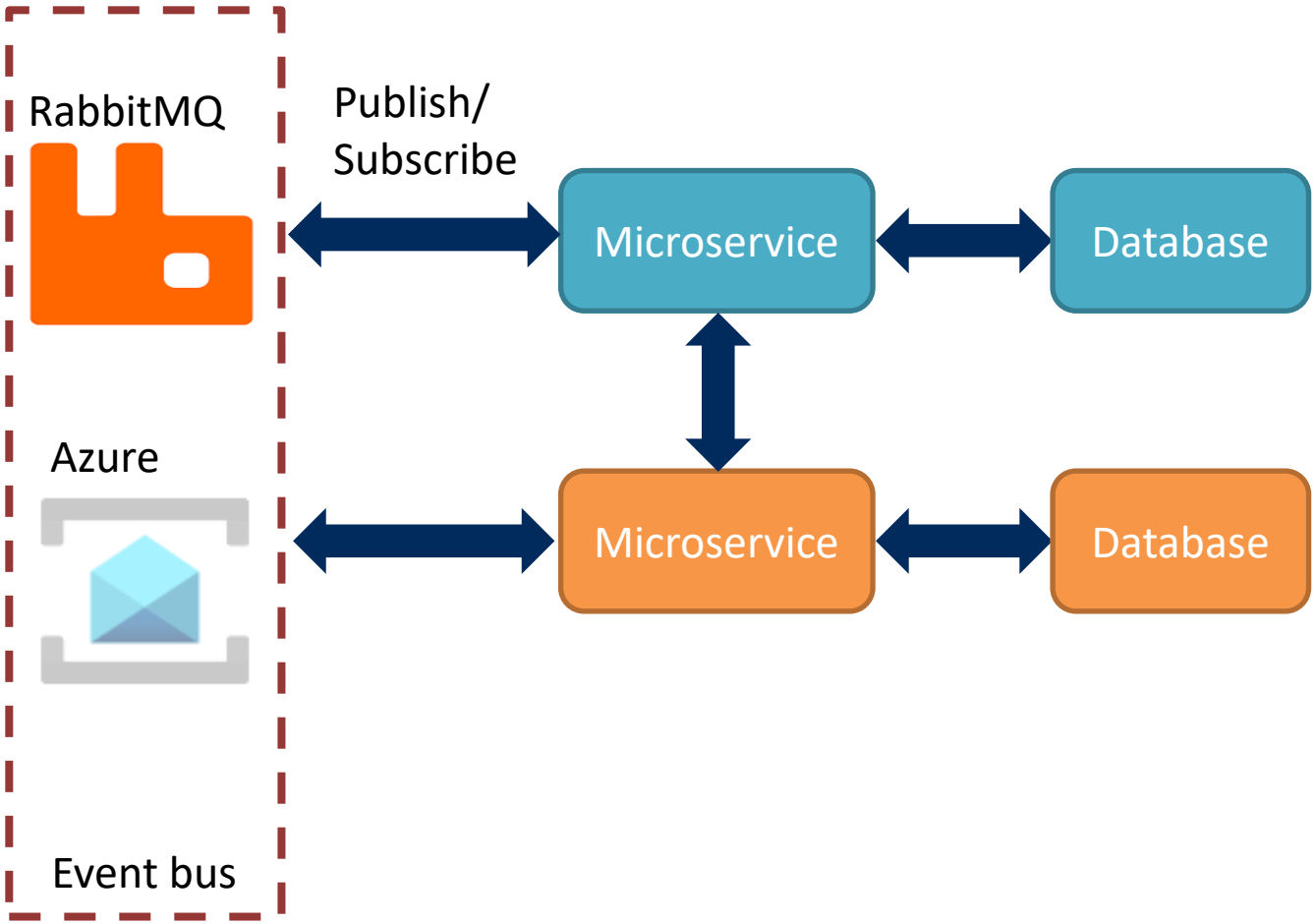
# Monolithic


Mobile application
Requests
Replies
Web browser
Read & write
Application server
Database server

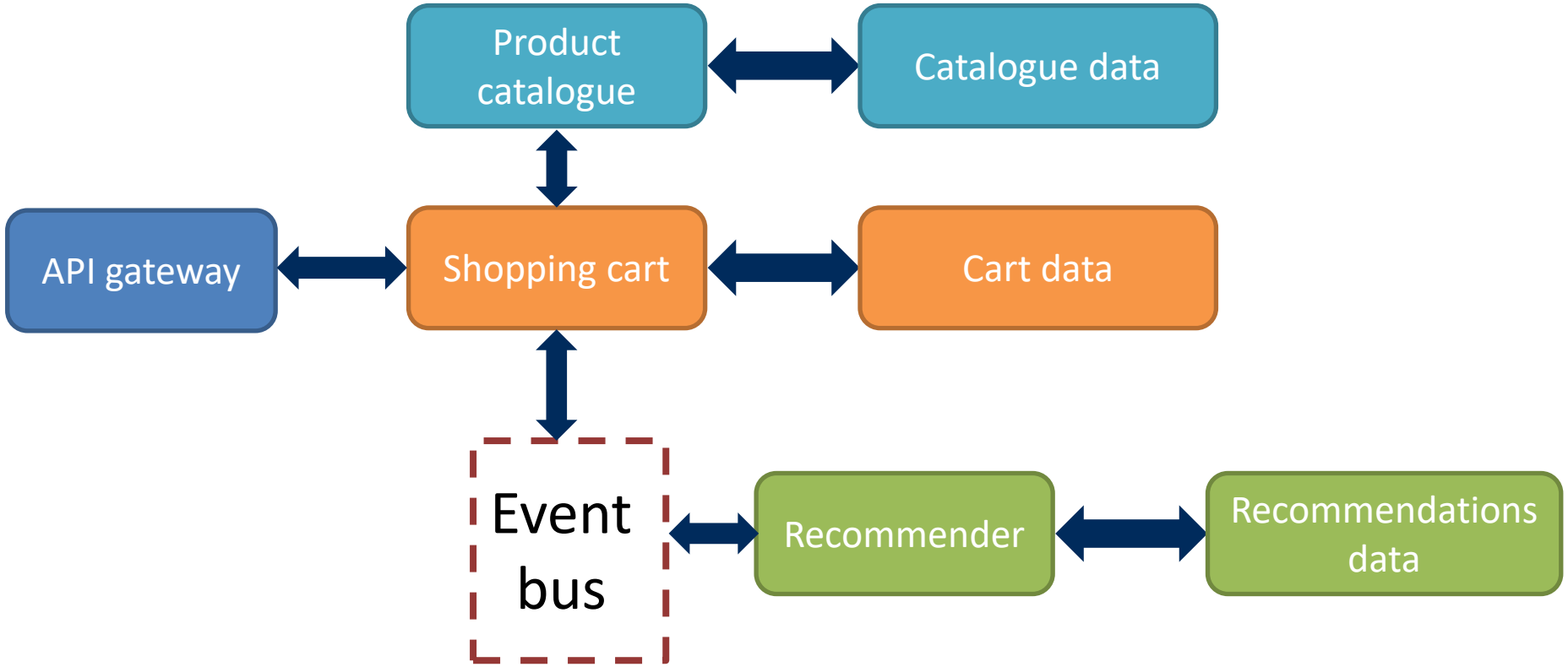
# Microservices



# Microservices: Events & requests



# Microservices: Online shop



# Microservices: Benefits

University of Strathclyde Glasgow

- Easier to maintain.
  - Each microservice is independent.
  - Update one of the microservices.
- Easier to scale.
  - Clone microservices.
- More flexible.
  - Use different implementation (stack/OS) for each service.

# Microservices: Issues

University of Strathclyde Glasgow

- Architecture.
  - Need to decouple into services.
- Integration testing.
- Deployment complexity.
  - Secrets.
  - Resources.
- Latency – connections between microservices.

# Function as a service

University of Strathclyde Glasgow

- Developer provides a function.
- Function is executed on specific event.
- Function can receive data:
  - From HTTP method.
  - From data storage.
- Function can return data:
  - As HTTP return value.
  - To data storage.

# Function as a service: Benefits

University of Strathclyde Glasgow

- A subset of serverless resources.
- Reduces development complexity.
- One action per function.
  - Lightweight.
  - Cost per execution.

# Master and slave(s)

University of Strathclyde Glasgow

- Distribute load across services.
  - Services may be servers that are used by clients.
- Need to ensure service consistency and resilience.
  - State changes through master service.
  - Read requests through master or slave.
  - Slave can become master if master goes offline.

# Peer-to-peer

University of Strathclyde Glasgow

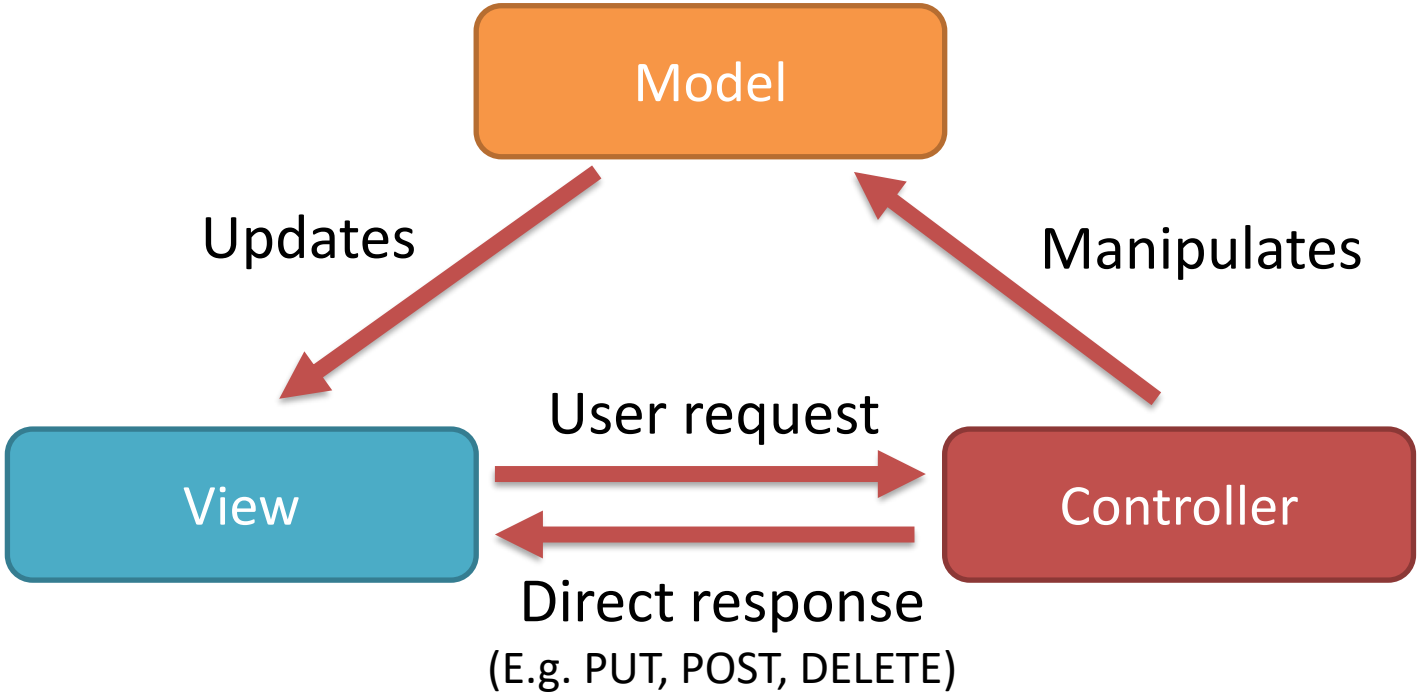
- Decentralised network.
  - Services are clients and servers.
  - Resilient due to lack of central servers.
  - Assume trust within network.
  - Need to implement logging and security mechanisms.

# Peer-to-peer: Applications

University of Strathclyde Glasgow

- Monitoring and control.
  - Internet of things devices – mesh network.
  - Network and service security is a big issue.
    - Spoof and replay attacks.
    - Hybrid server and peer-to-peer may be needed.
- Other uses.
  - Messaging and communication.
  - File sharing – associated with software piracy.

# Model View Controller (MVC)




S. Burbeck, “Applications programming in smalltalk-80 (tm): How to use model-view-controller (mvc).”, Smalltalk-80 v2 5 (1992): 1-11.

# Model View Controller (MVC)

University of Strathclyde Glasgow

- User views data through view.
  - View enables user to raise request to controller.
- Controller receives request and acts as needed.
  - Call other features or functions.
  - Manipulate data as needed.
  - Return view with associated data.
- Model describes the data model associated with the view.
  - May include some functionality within class definitions.

# MVC applications

University of Strathclyde Glasgow

- Web application.
  - More rigid than layered architecture.
    - Interactions between layers as specified by MVC.
  - Need routing description between address and controller.
  - Tends to be slower than non-MVC.
  - .NET (C#) MVC – database, controller and views.
    - Server-side rendering with Razor to create HTML.
    - Modern applications return JSON, rather than HTML.

# MVC applications

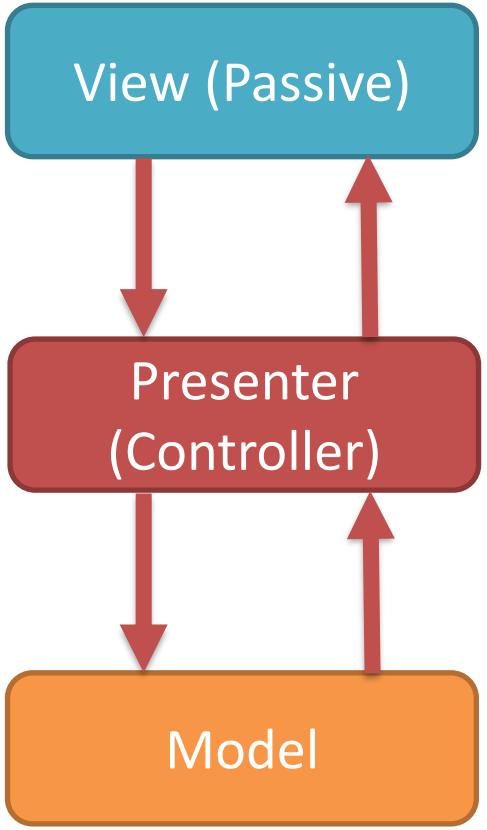
University of Strathclyde Glasgow

- Mobile applications
  - Flutter MVC.
  - iOS mobile development.

# Model View Presenter (MVP)

University of Strathclyde Glasgow

- Generalised form of MVC.
  - Presenter interprets events initiated by user.
  - Presenter provides business logic.
  - All business logic is in Presenter.
- Application:
  - User interface programming.
  - Desktop or mobile.




M. Potel, “MVP: Model-View-Presenter the Taligent programming model for C++ and Java.”, Taligent Inc. 20 (1996).

# Model View ViewModel (MVVM)

University of Strathclyde Glasgow

- Developed by Microsoft for desktop applications.
  - Can be used to develop Android applications too.

https://learn.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm

https://www.digitalocean.com/community/tutorials/android-mvvm-design-pattern

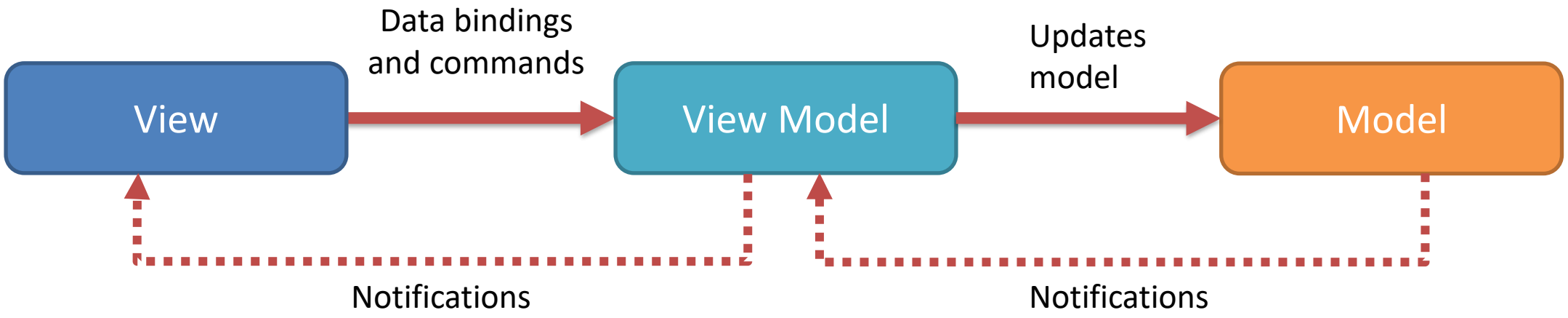
# Model View ViewModel (MVVM)

University of Strathclyde Glasgow

- User interacts with View.
  - Defines layout of GUI elements.
- The View elements are connected to ViewModel.
  - View is bound to ViewModel functions.
- ViewModel is able to access View.
- Active code is in ViewModel and Model.
  - Unit test without needing to test the View.

https://learn.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm

# Model View ViewModel (MVVM)




https://learn.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm

# Model View ViewModel (MVVM)

University of Strathclyde Glasgow

- Windows Presentation Foundation (WPF).
  - XAML markup for appearance.
    - XML-based markup language specific to WPF.
  - C# to implement features.

https://learn.microsoft.com/en-us/dotnet/desktop/wpf/overview/

# Combining patterns

University of Strathclyde Glasgow

- Can use more than one architecture design pattern.
  - E.g. MVC web application with persistency layers.

# Conclusions

University of Strathclyde Glasgow

- Need to capture data model early.
  - Data complexity affects architecture choices.
- Appropriate architecture aids development.
  - Easier to work together with other developers.
  - Easier to test and expand.
- Bad architecture choice can be costly.
  - Harder to change during development.
  - May cause project failure.

University of Strathclyde Glasgow