SOLID - example

Single responsibility - Email sender vs email builder. Each class has one responsibility and changes to process should not bleed into the others.

Open/Closed - Open to extend closed to modification. Our system generates different types of reports. We call the mechanisms that fulfil the request fore reports 'providers'. Each one requires common logic for logging and callbacks for completion so we created a ProviderBase. The base encapsulates the common functionality. So any changes to logging or callbacks only effect the base objects responsibilities and not the children.

Liskovs - substitution - providers again (being able to swap different providers depending on need)

Interface Segregation - I don't really have an example of interface segregation just that it's wise to keep contracts small and pointed. A class should not have to depend on an interface with signatures it has no use for. Obviously there is a balance because we want to keep cohesion.

Dependency Inversion - This one is everywhere. I personal swapped an entire application into DI and IoC.

Observable v promise

* Observables are data over time. Promise only has single values

async/blocking –

http actions – post, put, get, delete, patch, options, header

api experience

* Renter, landlord, partners, pes partners, ui

CORS - Prevents malicious website in another tab from executing the content served up by your website’s server

Continues integration – Developers commit code at regular intervals throughout the day that is verified by automated builds allowing teams to detect problems early

Continues deployment – Any committed code that passes automated testing is automatically released into production environment. Should have in place robust automation testing and production monitoring tools/teams..

***Synchronous*** *–* Thread will complete the action either by success or failure.

***Blocking***– Thread will wait on an action until success or failure.

***Asynchronous*** *–* Another thread (logical or physical) will complete the action or inform it is ready using a callback.

***Non-Blocking***– Thread will not wait on to complete the action.