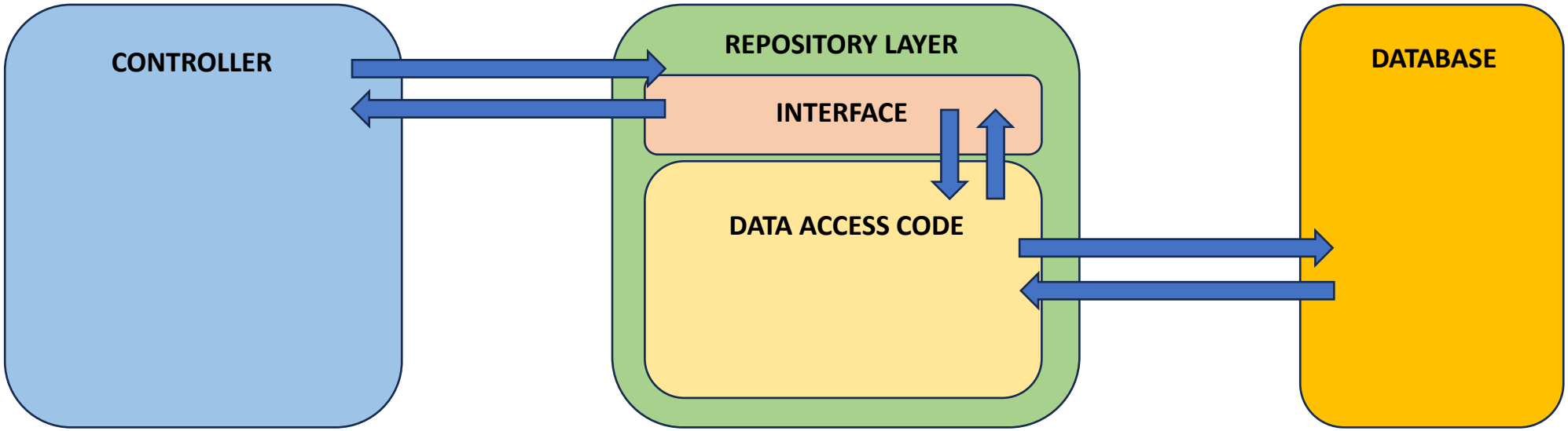
Repository Pattern

This programming design pattern is designed to create a separation (uncoupling) your controller/domain layer from the logic associated with interacting with the database. This not only creates a separation of concerns, but it also makes modifications/replacements of the data source and its associated code, without the controller being aware of it.



Whenever the controller needs data from the database, all it knows is it can make a request via a set of known methods that are given to it by the repository interface.

The controller is unaware of the logic inside the data access code, it only knows what requests it can make to the interface.

The repository layer contains an interface outlining all the allowed request types/methods that are allowed to be interacted with by the controllers.

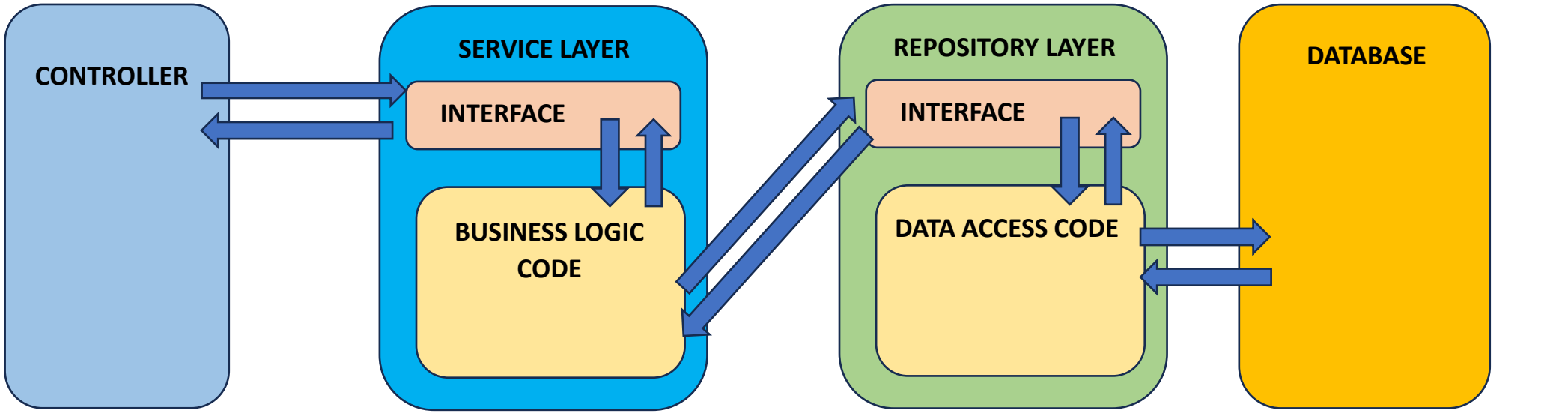
The data access code inherits from the interface and holds all the code related to requesting and finding data in the database. Most of this code will be inside the methods that are outlined by the interface.

Because all interaction to the controller is done through the interface commands, the data access code knows nothing about the controller, only the interface and data store. This means that if we want to replace the database or data access code, so long as the replacement parts still conform with the interface methods, you can add to the system without needing to make any modifications to the controllers.

The database holds all the data and only interacts with the data access code to fulfill requests.

The Repository -Service Pattern

The service pattern is similar to the repository pattern except it puts another layer between the controller and the repository. This service layer manages any specific business logic that needs to be applied between the controller and business data. It still uses interfaces to outine the commands/interations allowed and therefore keeps your code uncoupled and able to change out sections wihtout rewriting code in the adjoining sections.



Whenever the controller needs data from the database, it requests it from the service layer.

The service layer gets the request from the controller and manages any required business logic or rules as the data passes through.

This could be validating the request/data and checking permissions on the way to the database or filtering and formatting the data on the way back to the controller.

When it receives a request form the service layer, the repository manages any database interactions required and returns the required data to the service.

In this pattern the repository only deals with the data requests, any additional logic or code associated with business rules is managed by the service layer.

The database holds all the data and only interacts with the data access code to fulfill requests.