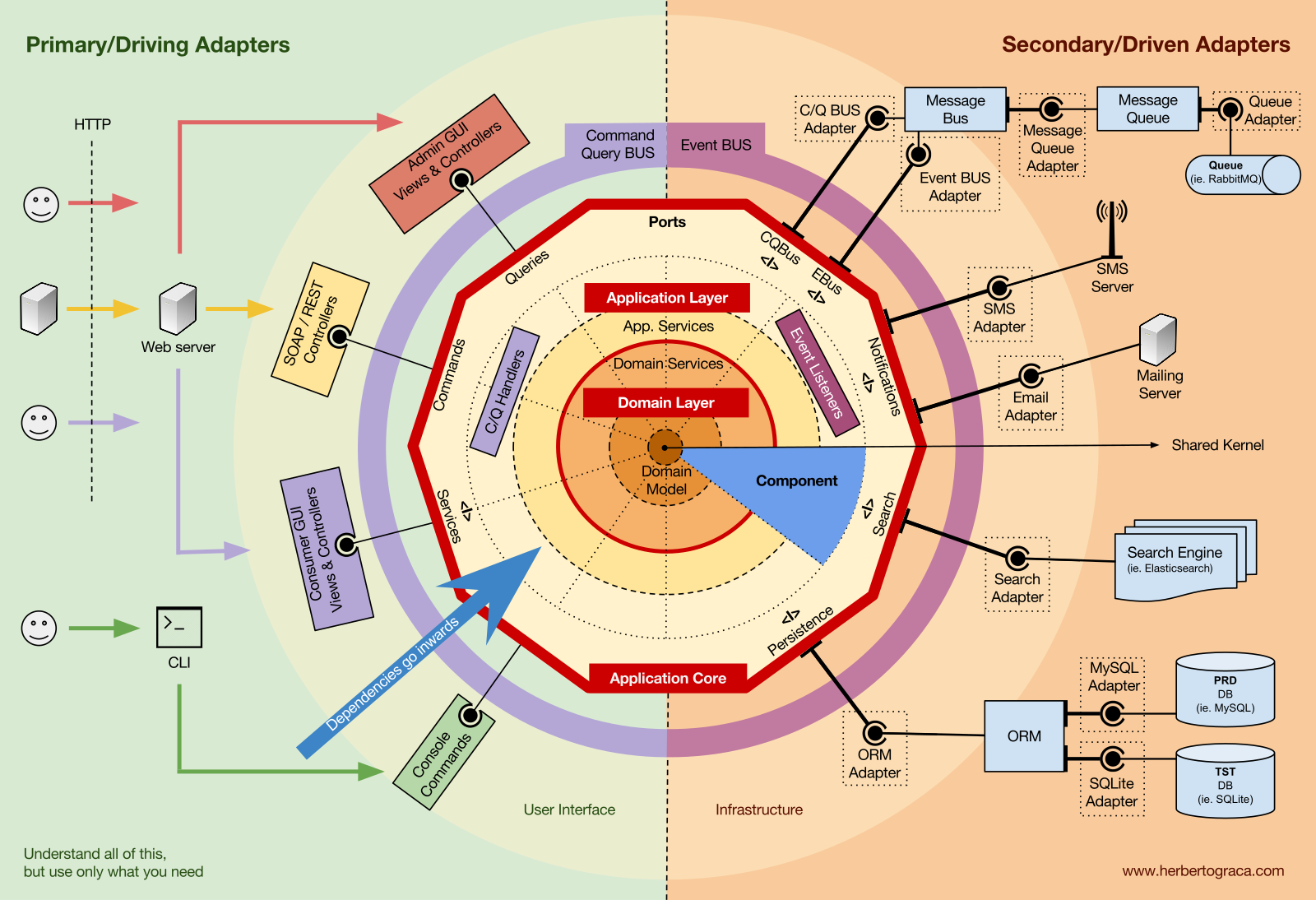
**StarLabs 2022 - Documentation**

**Hexagonal Architecture**

Hexagonal architecture is divided into three parts and defines the strict roles that these parts play within the application.

1. User Interfaces
2. Business Logic
3. Backing services

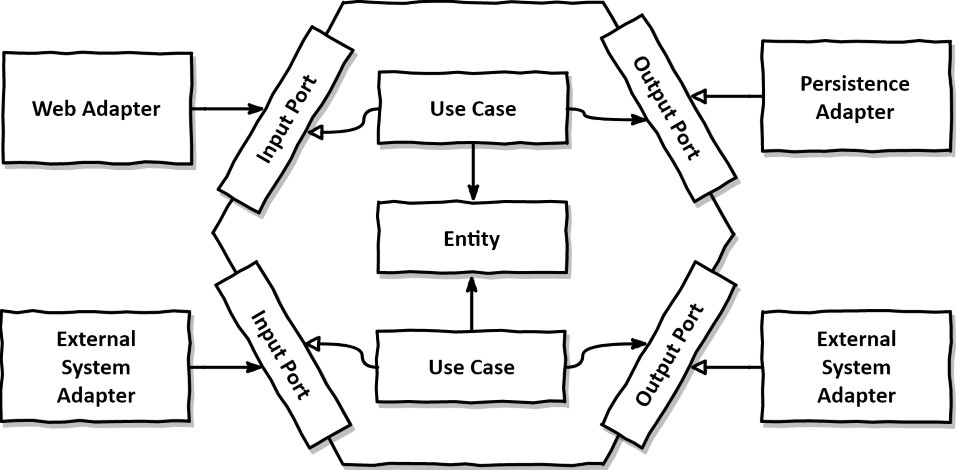


***User Interfaces:***

There can be many user interfaces to a backend - mobile apps, web apps, desktop software, etc. They will all get their resources from the Business logic layer.

***Business Logic:***

It forms the core of the application. It's objective is to cater to the requests of user interfaces. Based on the request, it runs some custom logic, gets the resources needed to fulfil the request and answers back in an agreed upon response format.



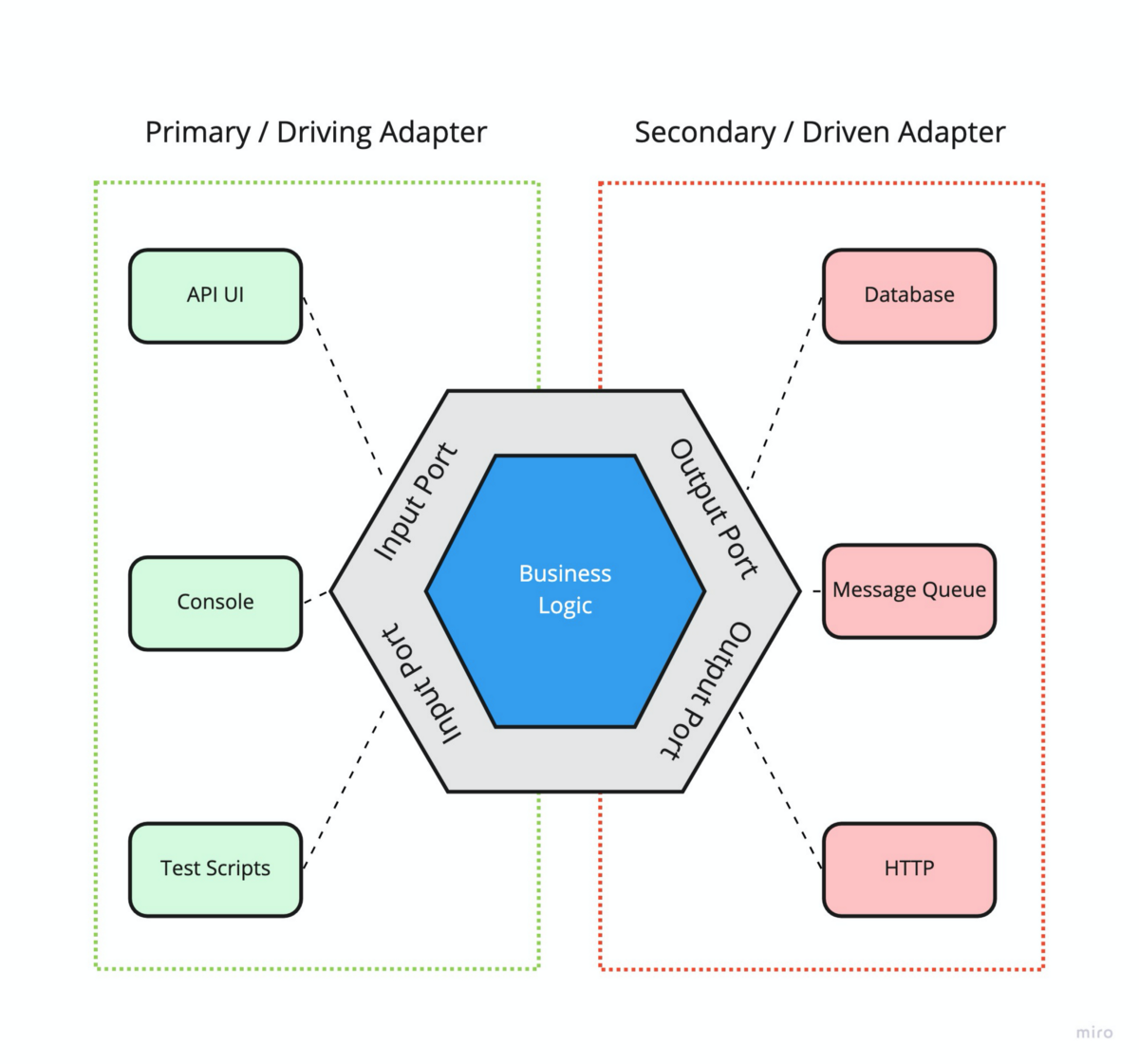
***Backing services***

These are services which support the business logic. They each serve a specific purpose and provide data/services to the application. They interact with the business logic layer and are replaceable as long as the communication contract between the two layers is maintained. A few examples:

1. Data sources
2. Cache aside servers like Redis
3. Notification services
4. Another service like a payment gateway
5. In microservices context, another microservice

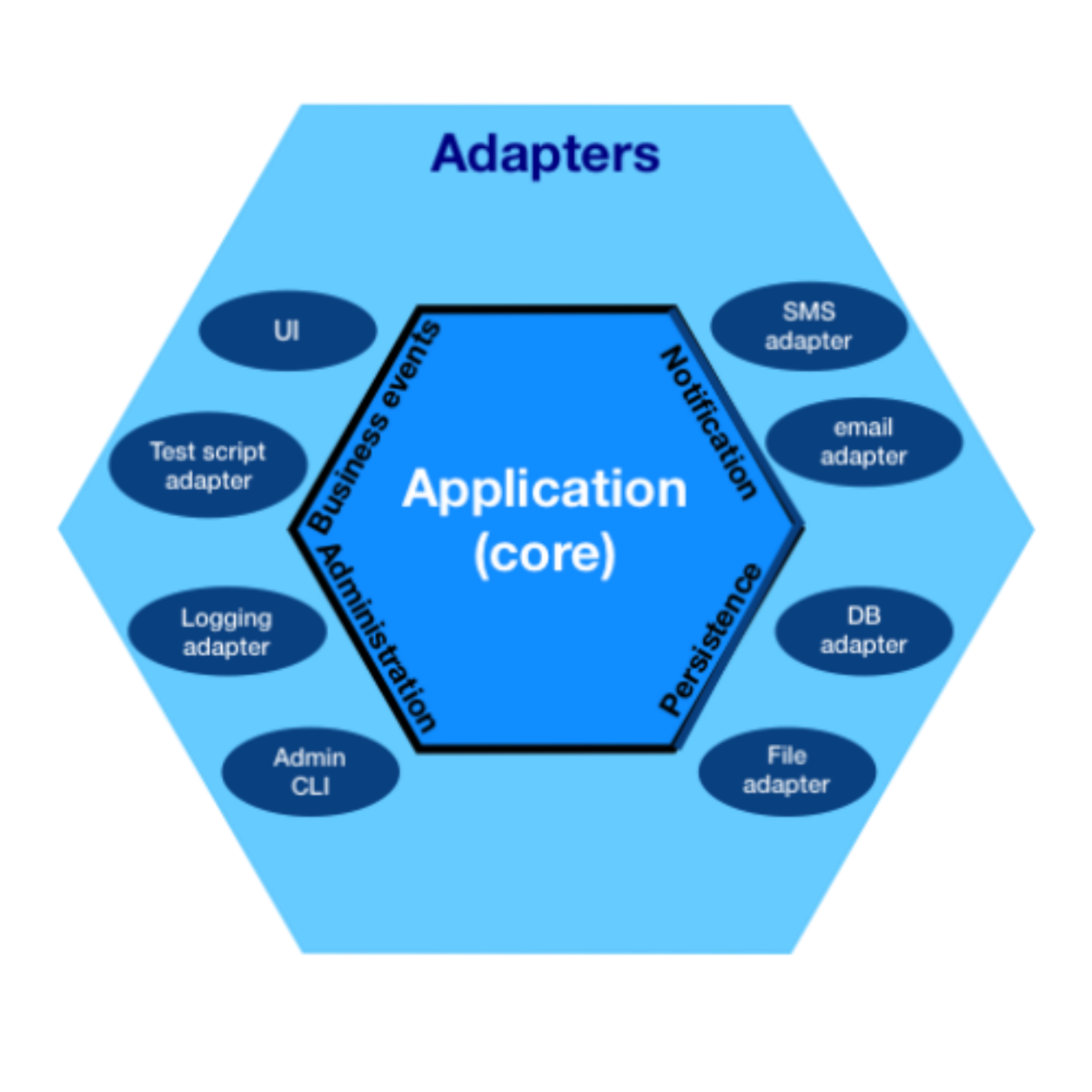
***Ports:***

Ports are what our core application interacts with. Ports stay consistent for the inner application no matter what happens outside them. They are interfaces that the inner components interact with without knowing what’s being plugged into them.



***Adapters:***

Ports are staying consistent but we still want to be able to plug multiple applications to them when needed. These applications could have different needs and may not comply with the interface defined by the ports. This is where out adapters come in. Their purpose is to convert the data provided by the outer applications into a format digestible for the inner application.



***Advantages:***

1. **Swappable components** - as we can see in the database layer. There could also be other services in the same pattern. For ex. I could have notification services and swap between emails and SMS whenever needed.
2. **Separation of business logic** - If implemented well, the hexagonal architecture does not pose a threat to the business rules at the core of the application when outer layers change.
3. **Easier testing across ports** - Testing of the core application can be performed around the ports. If needed, mock resources can be introduced using adapters of their own to make unit testing without databases easier