Back To Module Home

Architecture of Scalable Applications

0% completed

Different Tiers in Software Architecture

Monolith and Microservices

Conclusion

Mark Module as Completed

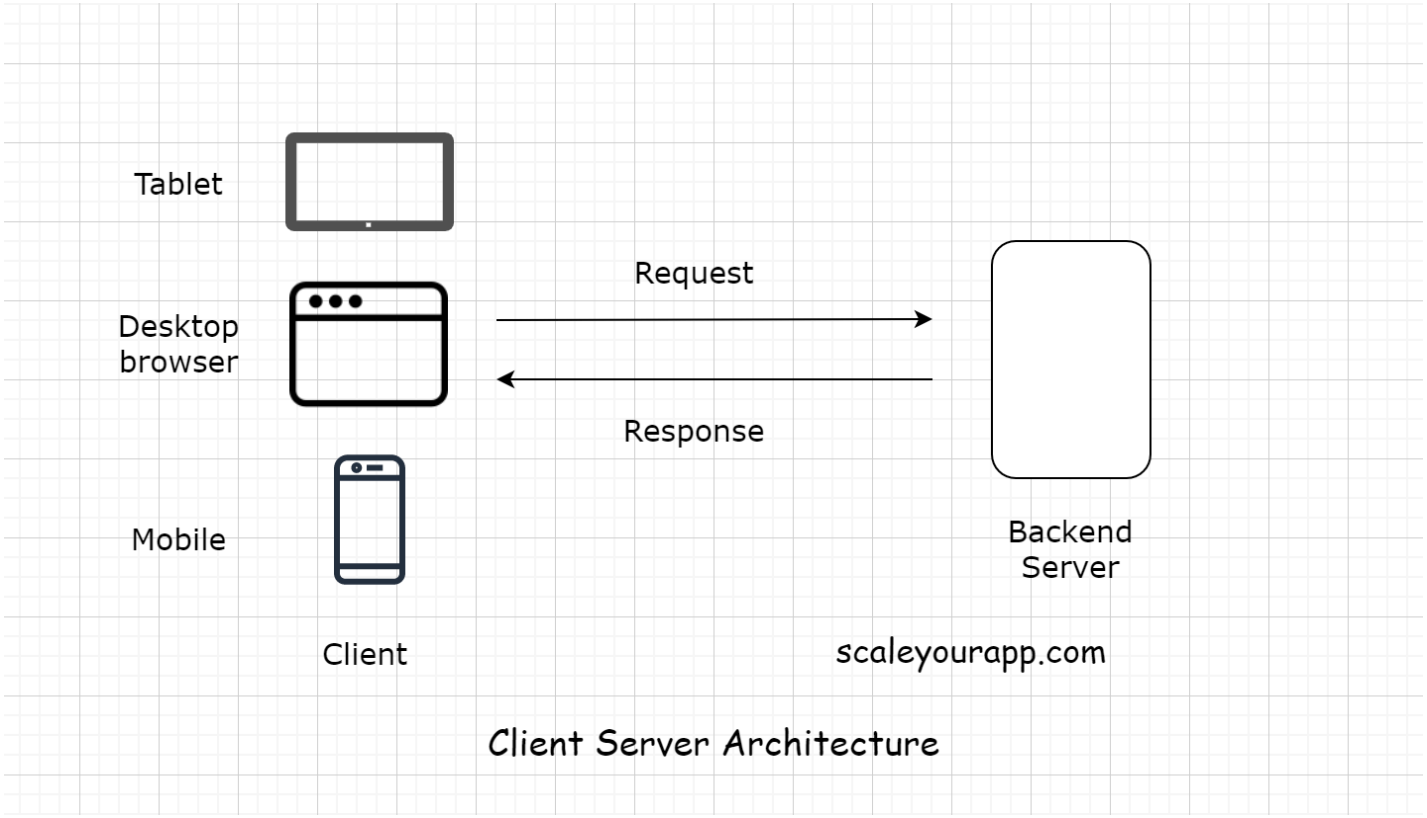
Two-Tier Applications

In this lesson, you will learn about the two-tier applications.

We'll cover the following

Two-tier application#

> A *two-tier* application involves a *client* and a *server*. The *client* contains the *user interface* with the *business logic* in one machine. Meanwhile, the *backend server* includes the *database* running on a different machine. The *database server* is hosted by the business and has control over it.



Why do we need *two-tier* applications? Why not host the *business logic* on a different machine and have control over it too?

Also, again isn't the application code vulnerable to being accessed by a third person?

The need for two-tier applications#

Well, yes! The code is vulnerable. However, there are use cases where *two-tier* applications come in handy, for instance, a *to-do list* app or a similar planner or a productivity app.

In these scenarios, even if the code is accessed by a third person, it won't cause the business much harm. On the contrary, since the *business logic* and the *user interface* reside in the same machine, there are fewer network calls to the *backend server*. This keeps the latency of the application low, proving to be an upside.

Taking a *to-do list* app as an example, the application makes a call to the *database server* only when the user has finished creating their list and wants to persist the data.

Another good example of *two-tier* apps is the browser and mobile app-based games. The game files are pretty heavy, and they only get downloaded on the *client* once when the user uses the application for the first time. And they make the network calls to the backend only to persist the game state.

Additionally, fewer server calls mean less money spent to keep the servers running, which is naturally economical. However, picking this tier type for our service largely depends on our business requirements and the use case. When designing our system, we can choose to keep the *user interface* and *business logic* on the *client* or move the *business logic* to a dedicated *backend server*, making it a *three-tier* application that we will learn in the lesson up-next.

Single-Tier Applications

Next

Three-Tier Applications

Mark as Completed

Report an Issue