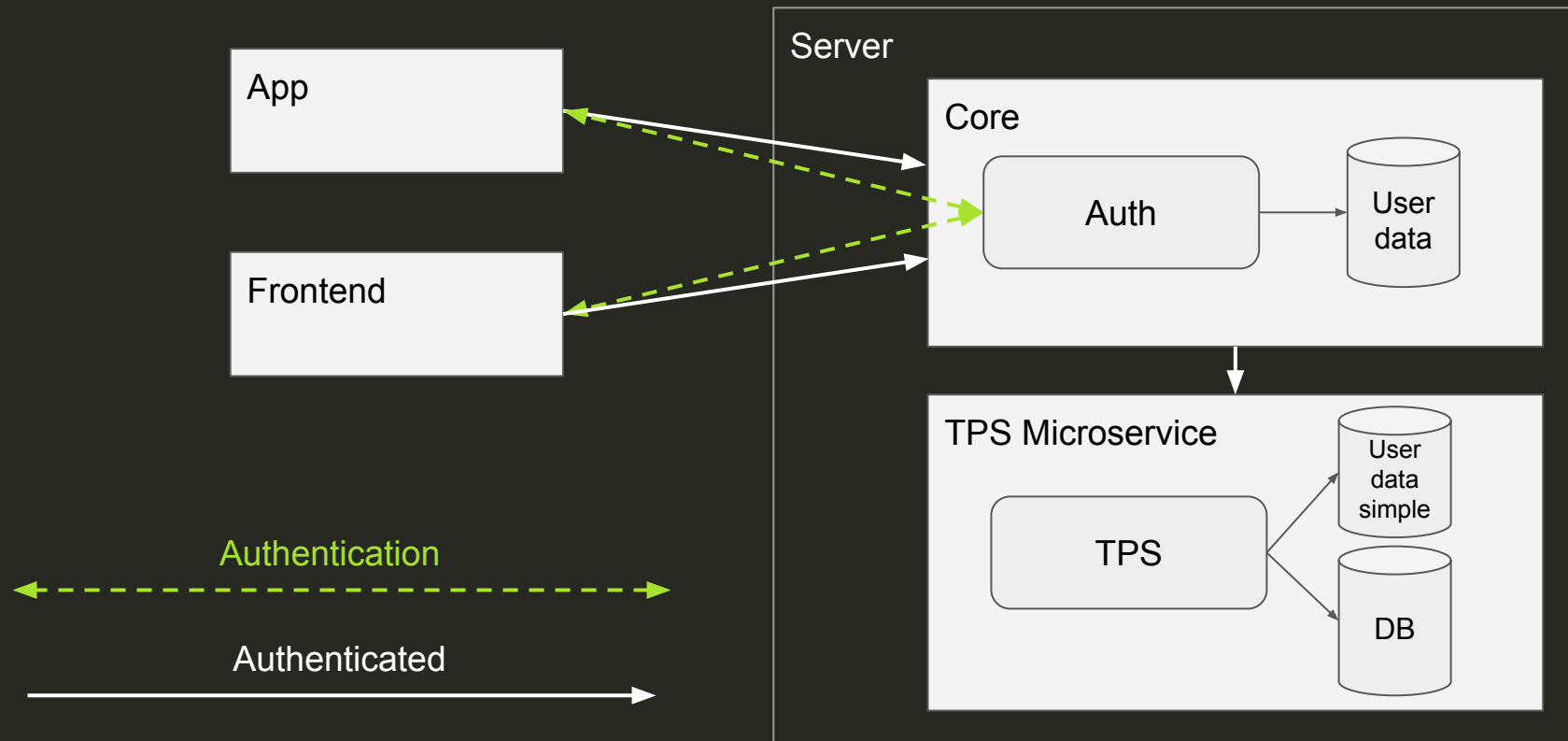


Authentication and identity management in a microservice

delegation of responsibility

Preferred layout:



Definition of a microservice

- Small service decomposed from a monolith
- Isolated and independently deployable
- Stateless and less fragile when changes are introduced
- Single responsibility
- Advice was provided in the [Pregame document](#)

Aspects of authentication identity management

Name

Explanation

Example

- | | | |
|------------------|---|--|
| - Responsibility | System concerned of authenticating users? | Core, external service or microservice itself |
| - Locality | Where is user data stored? | A single database, all databases |
| - Authorization | How do we identify user and roles? | CompanyId, DaAppInstall ... |
| - Statefulness | How is the state of authentication shared between services? | In the request, in shared or separate sessions |

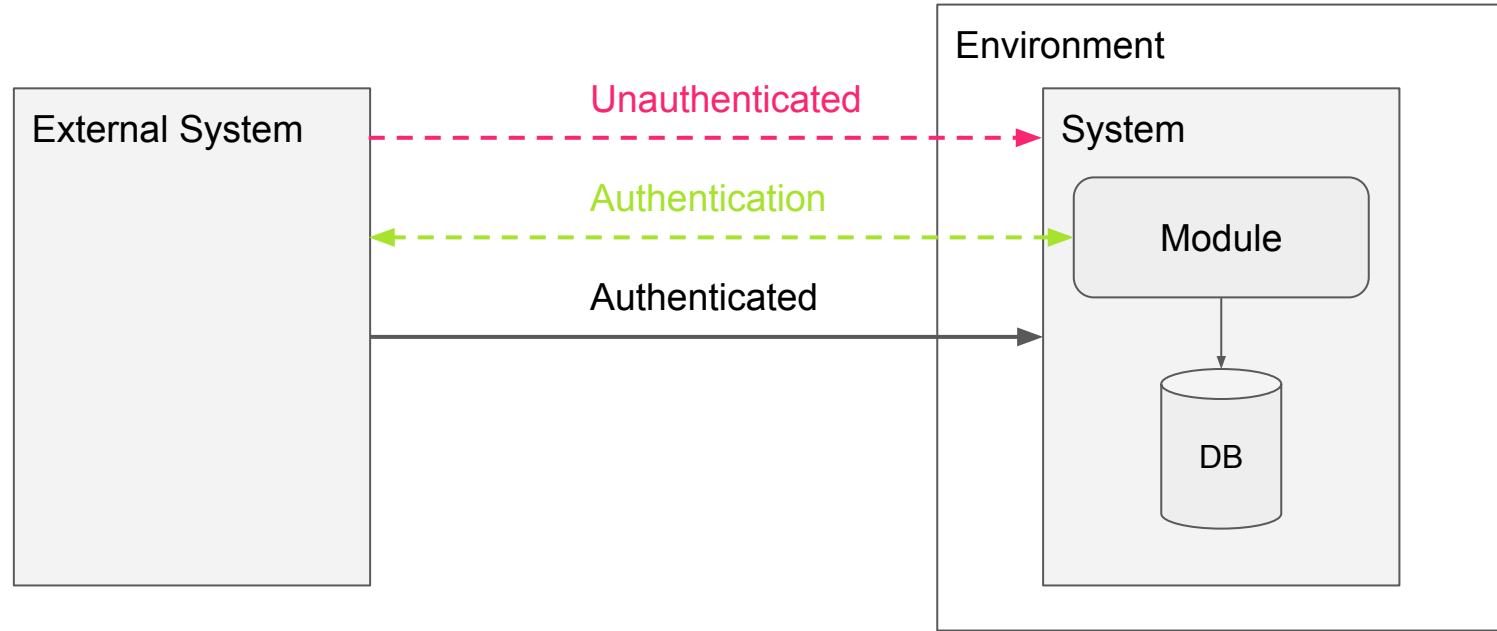
Examples

Here are four examples 1 ... 4 increasing from basic to more extreme implementations.

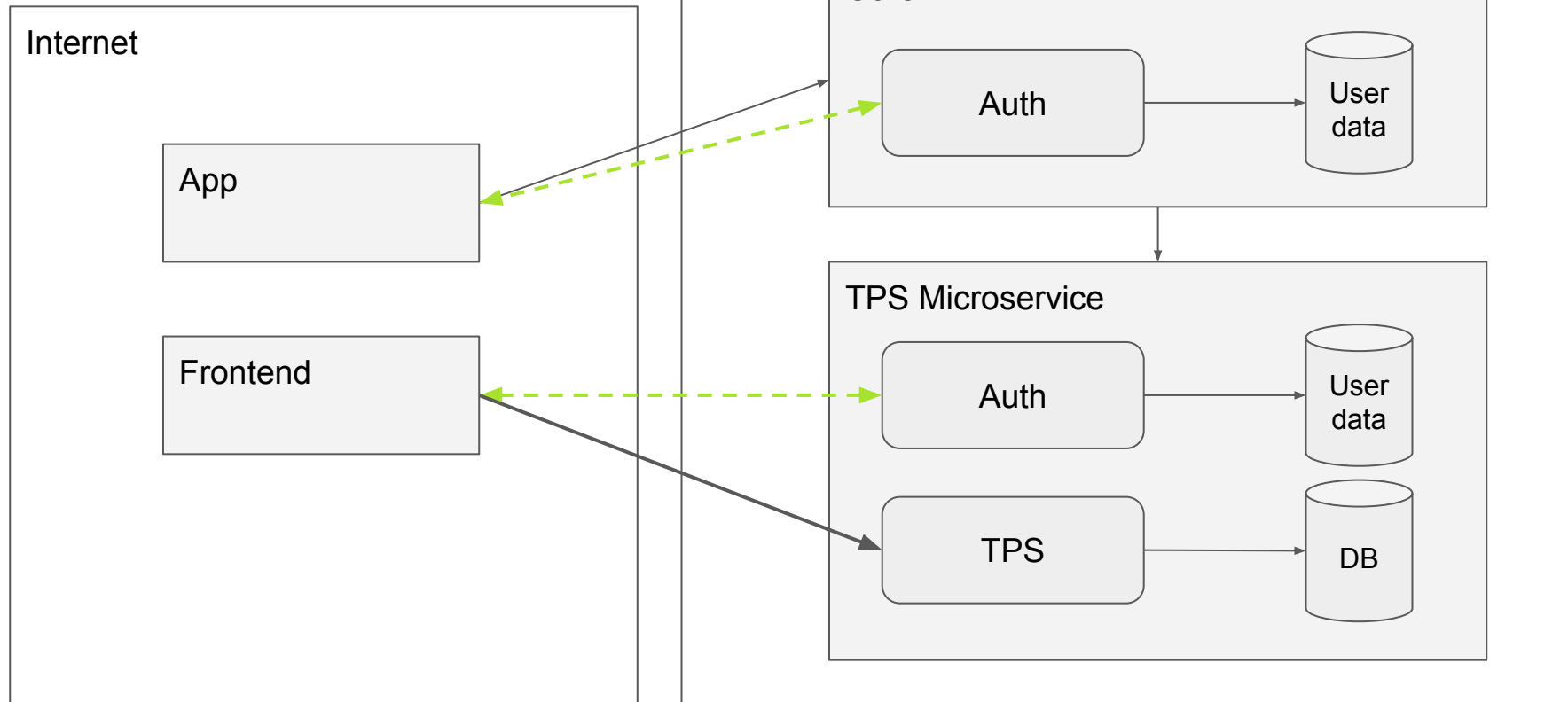
The four aspects are discussed after each figure:

- Responsibility
- Locality
- Authorization
- Statefulness

Symbols used in examples



Ex 1



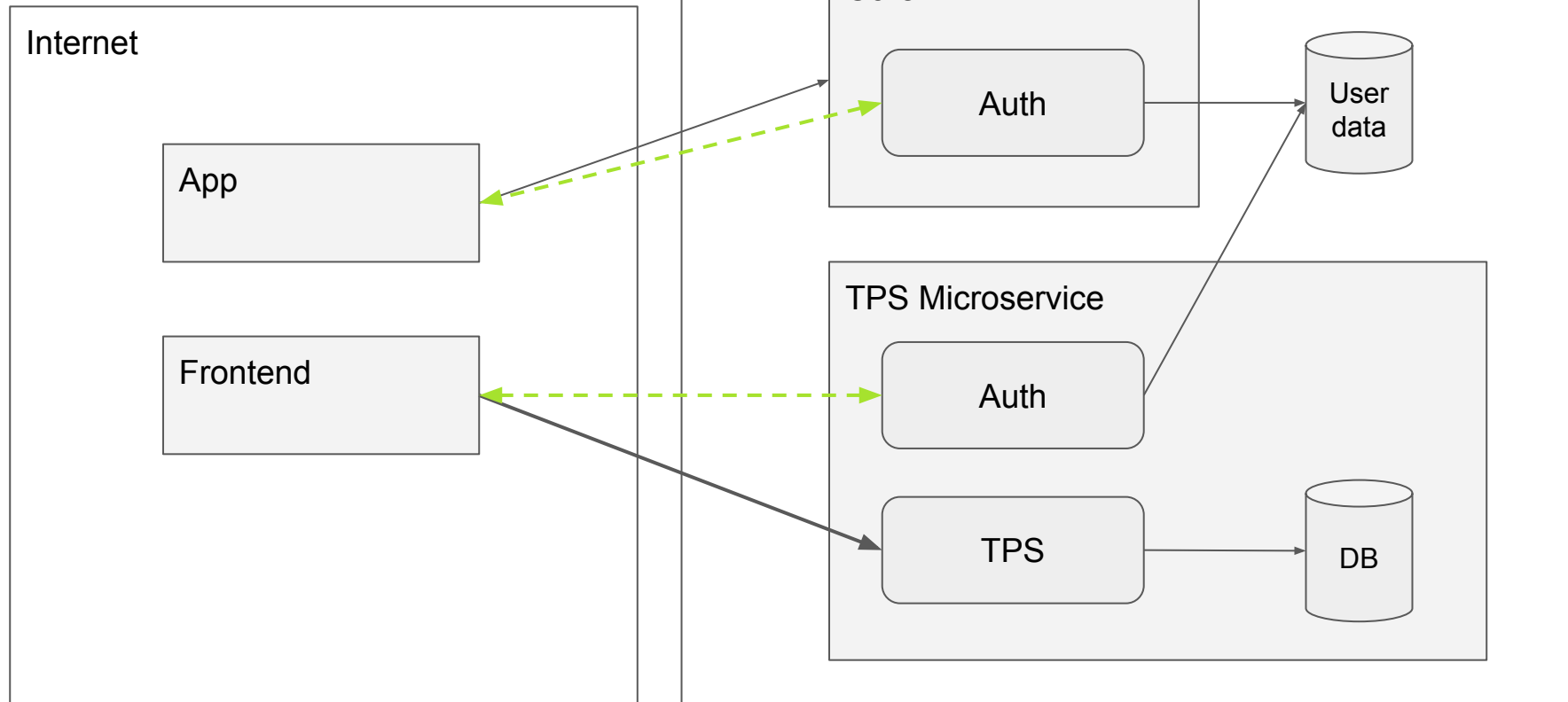
Ex 1 - Aspects

The TPS microservice authenticates its users. It has its own database with user and company data.

Authorization is handled by checking the user data in the database. Sessions are handled by the microservice. So the state resides in the microservice.

The microservice is totally independent, except for the fact that the data that is mutated in other systems must be synchronized in some fashion.

Ex 2

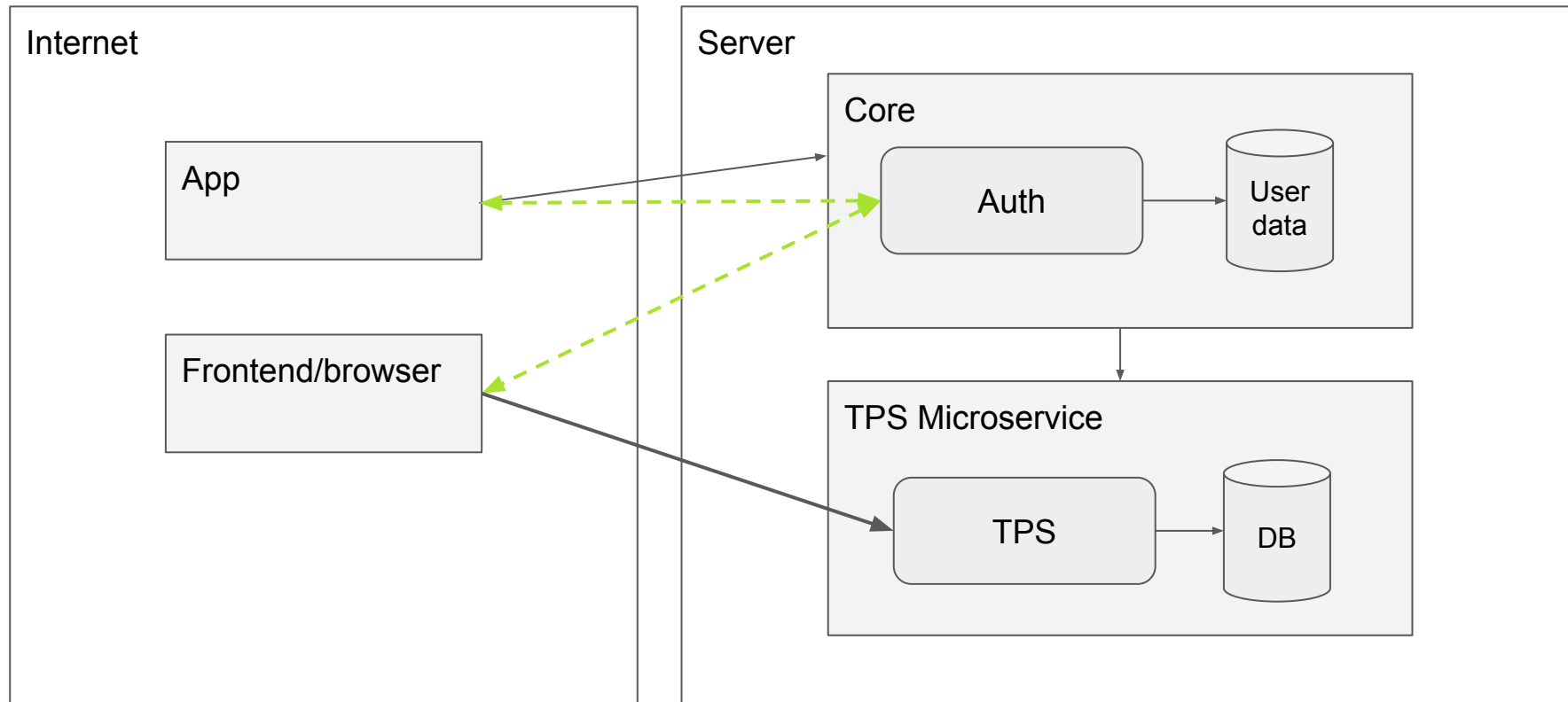


Ex 2 - Aspects

Like example 1, authentication is handled by the microservice. Only this time it connects directly with a database that stores user information.

Depending on how sessions are handled, the state can be shared amongst systems. But when other systems are required to make use of the microservice, more and more sources that contain the state of users need to be shared with the microservice.

Ex 3



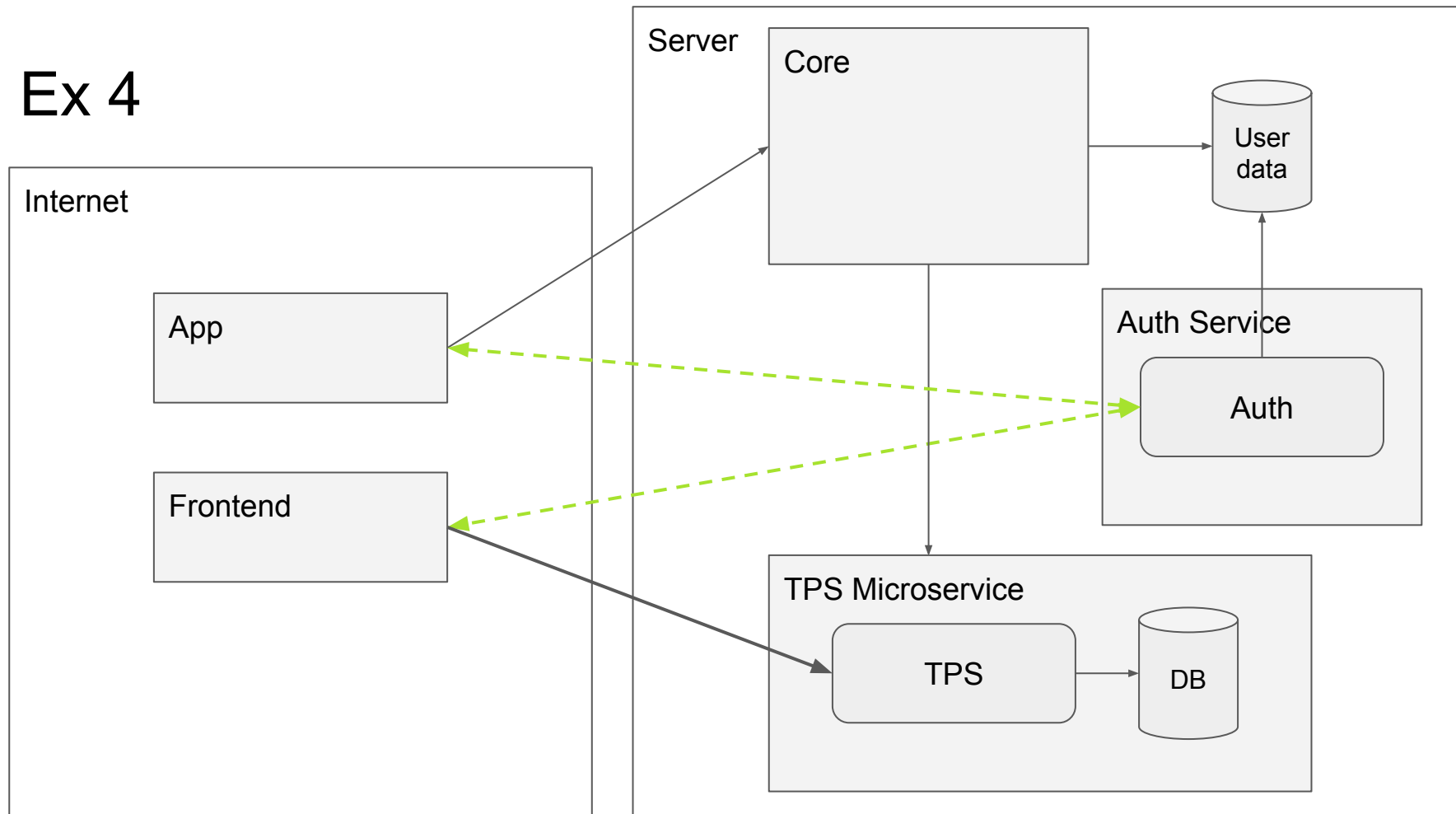
Ex 3 - Aspects

In example 3, the Core system is the only system able to provide authentication tokens. This token must be used to transfer authorization and identity information to the microservice in a stateless manner, because the microservice has no concept of the state of authentication.

A JWT can be used to transfer state in this case.

If in the future, more systems have to make use of the microservice, they depend on the core system anyhow.

Ex 4



Ex 4 - Aspects

In this example, future systems don't depend on the core system. But the core system does depend on the Authentication service in order to make use of other microservices.

Things to keep in mind with JWT

- Managing invalidation of JWT's
 - Invalidating individual tokens conditionally
 - <https://stormpath.com/blog/token-auth-spa>
- Keeping payloads up-to-date when data changes
 - User switches from company, therefore the frontend must act!