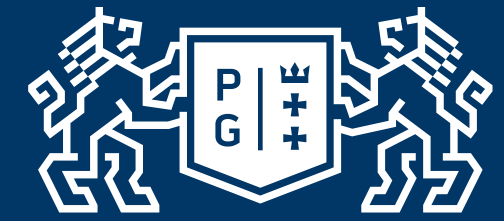


Spring MVC - REST Services

Internet Services Architectures

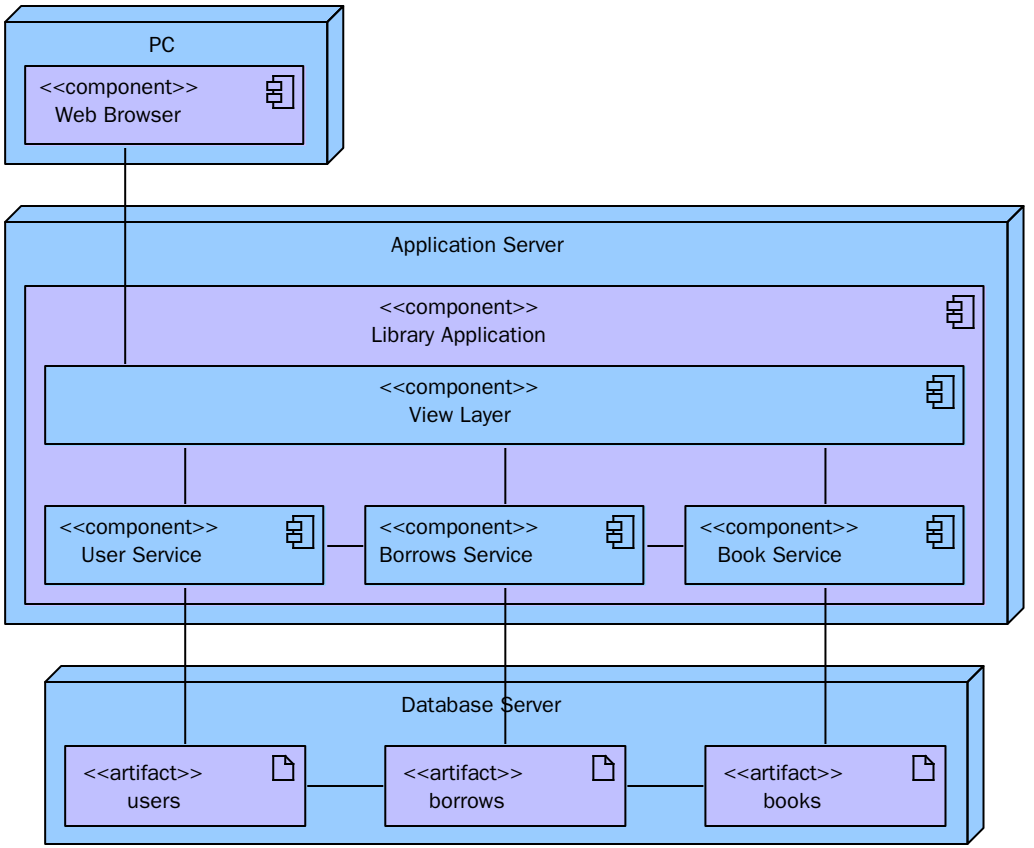
Michał Wójcik

Classic application approach

How applications were (sometimes still are) developed in the past?

POLITECHNIKA GDAŃSKA

Monolithic Application



Monolithic Application

Monolithic application:

- simple to develop small applications (everything in single place),
- all teams work on the same project (maybe different modules), know the same context,
- easy to deploy (e.g.: single war file for Java web applications),
- easy scaling by running multiple copies.

Problems start when application goes **big**:

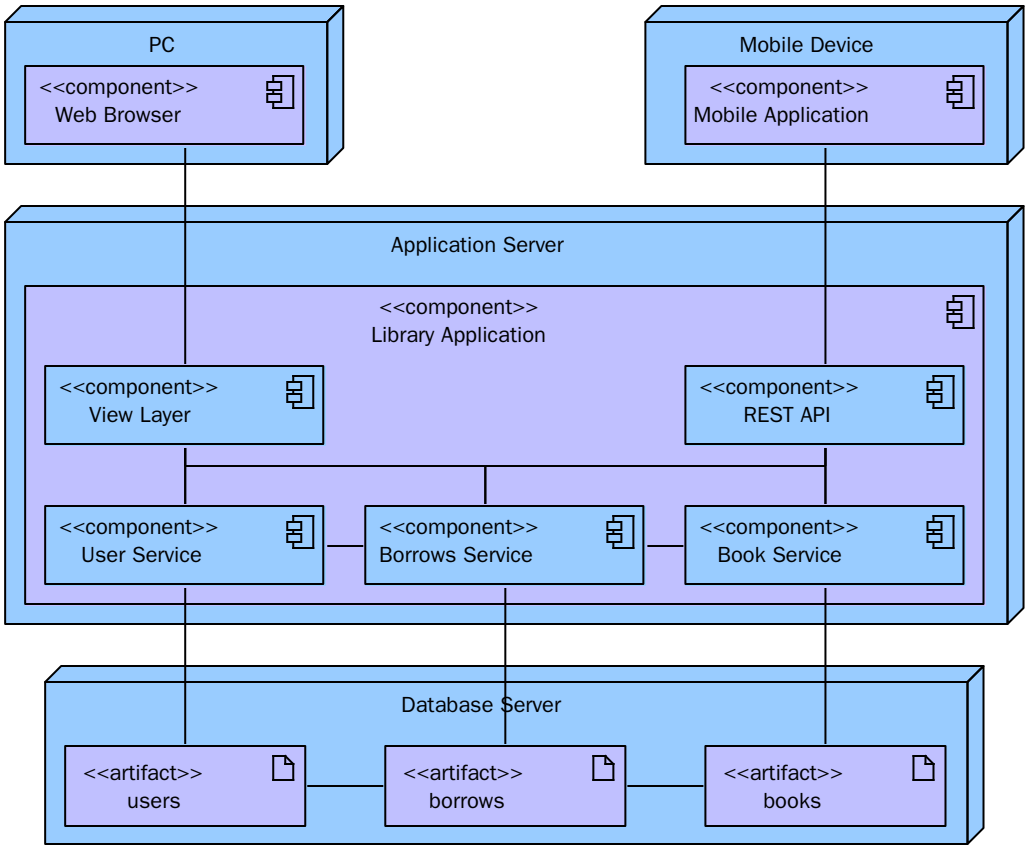
- a lot of code in one project is difficult to understand,
- high cost of introducing new project members,
- it's difficult for teams to work independently,
- the larger code base is the slower IDE is,
- long CI/CD process,
- long-term commitment to technology stack.

Mobile and embedded devices

With increasing usage of devices like smartphones, tables, smart TV etc. there was demand for programming API instead fo HTML web pages.

POLITECHNIKA GDAŃSKA

REST Endpoint



REST Endpoint

Adding **REST endpoint** to **monolith**:

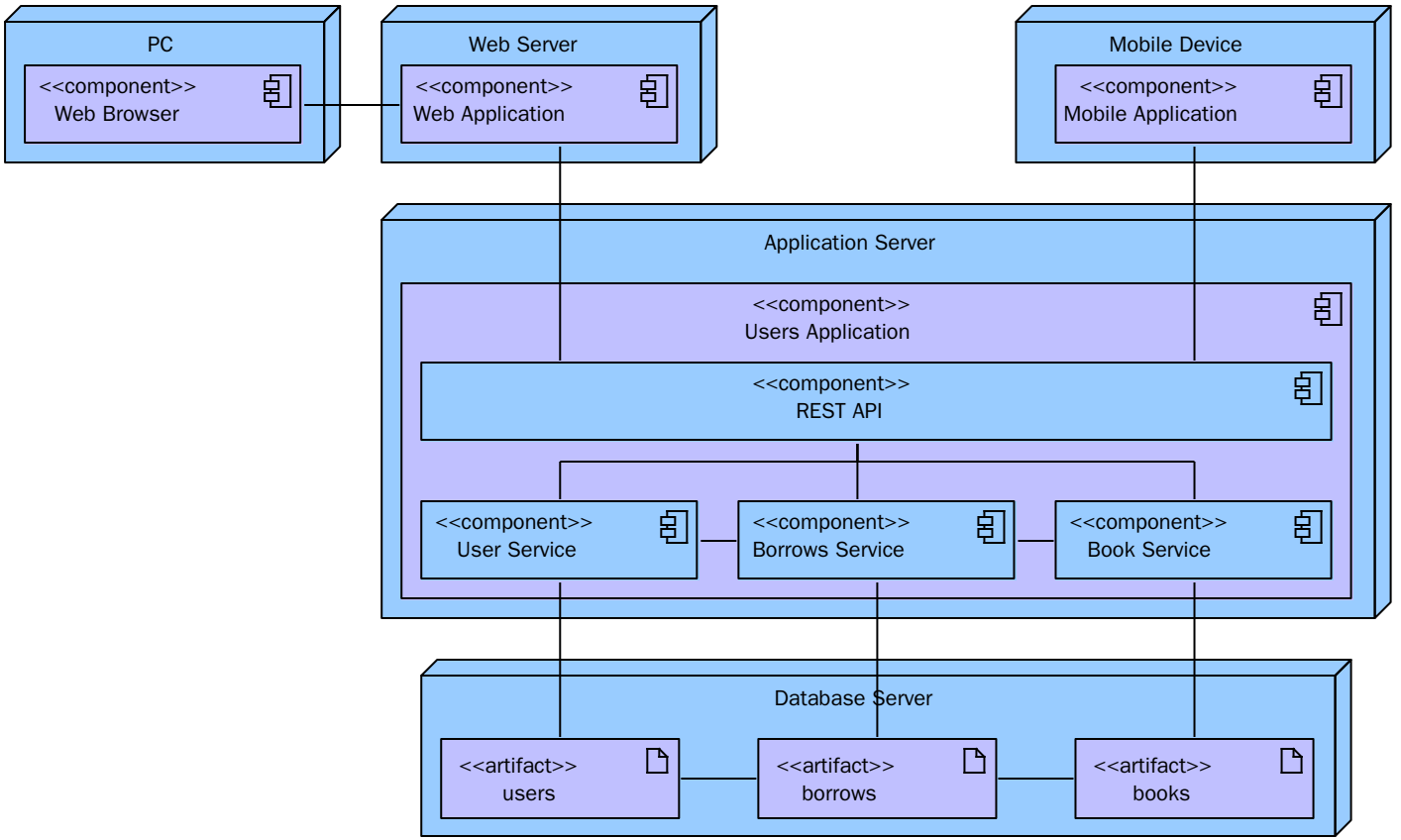
- two access modules in one application must be maintained,
- even more responsibility for project team which must have knowledge about next module,
- separate teams can develop different mobile clients without knowledge of base application implementation (only REST API documentation is required).

Single entry point

If we must maintain REST API, maybe it should be the only entry point?

POLITECHNIKA GDAŃSKA

External Web Application



External Web Application

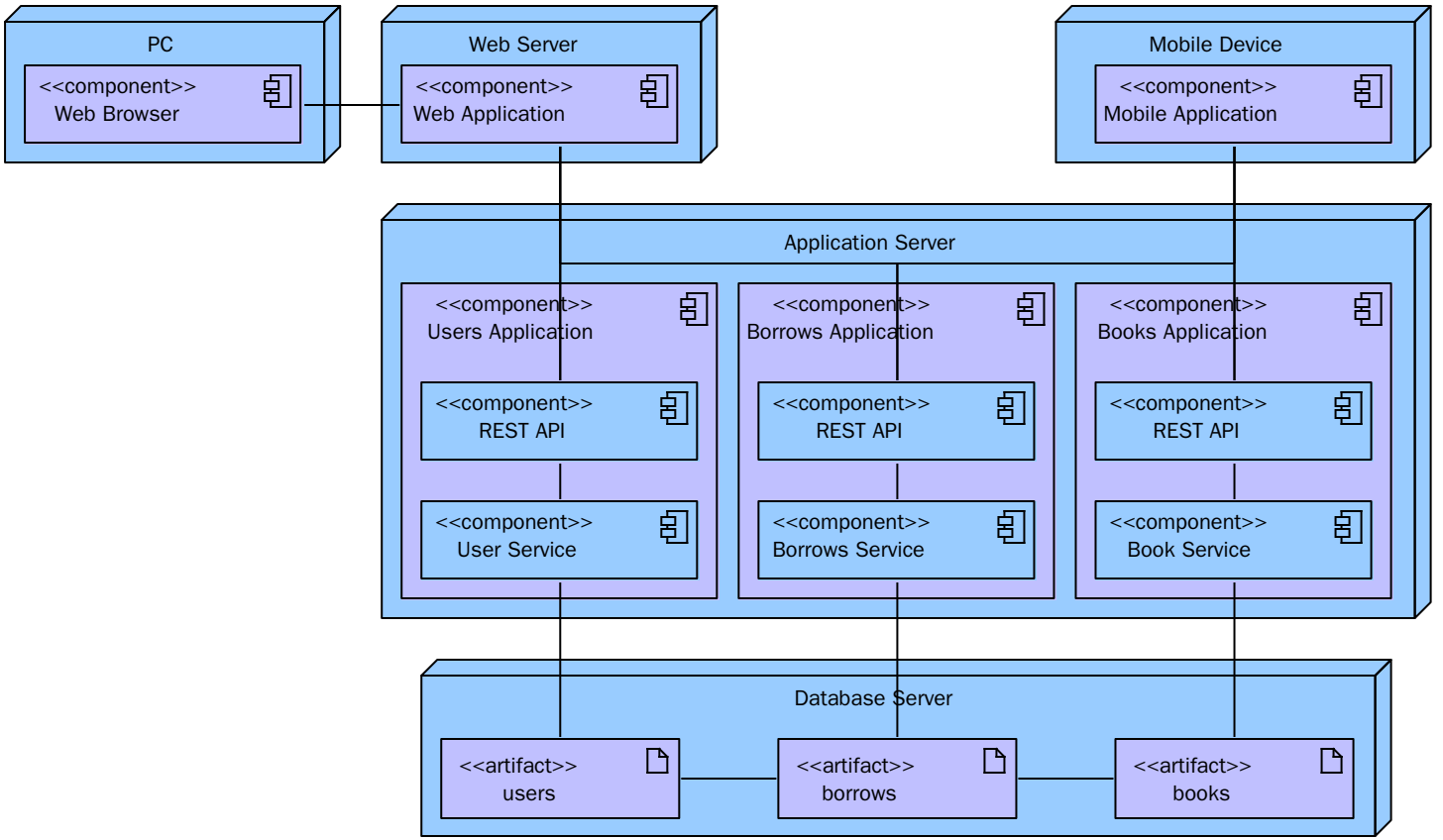
Using **external web application**:

- front-end developers as separated team,
- knowledge about whole application implementation is not required (REST API contract is important),
- back-end developers can focus only on business logic implementation not considering interactions with users.

Decompose monolith

Can we decompose the monolith application?

Microservices on Single Server



Microservices on Single Server

Using **microservices**:

- separated modules (different projects),
- different modules can be developed using different technologies,
- modules can communicate using e.g.: REST endpoints (but not only).

Using **single application server**:

- easy configuration,
- can use communication using non public ports,
- performance issues.

Using **single database**:

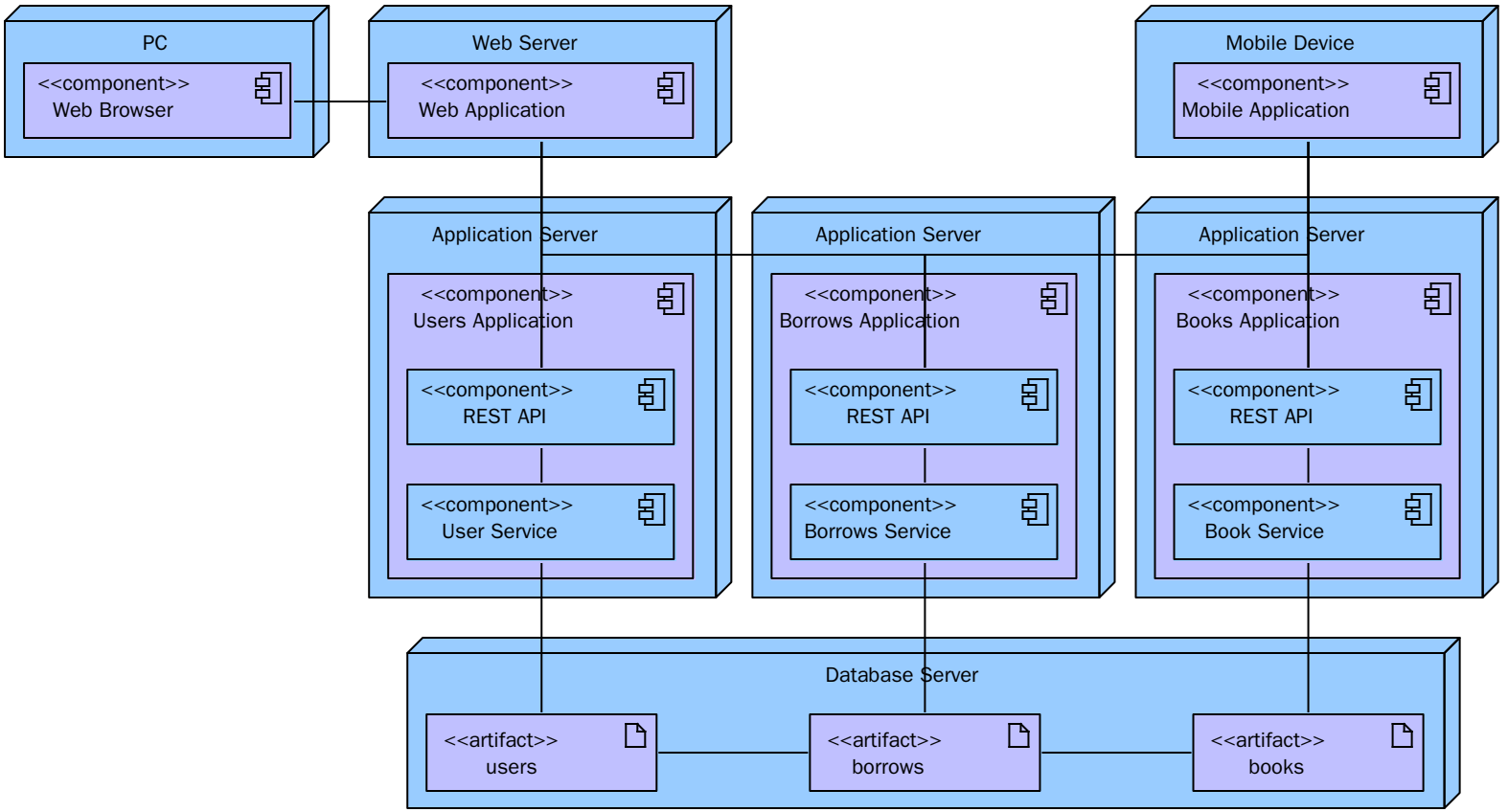
- simple in use ACID (atomicity, consistency, isolation, durability) transactions,
- one database is easy to maintain,
- schema changes must be coordinated between development teams,
- number of modules using the same database can lead to performance problems (e.g.: during long running transaction locks).

Separating modules

If modules are already separated we can move them to different servers.

POLITECHNIKA GDAŃSKA

Separated Application Servers



Separated Application Servers

Using **separated application servers**:

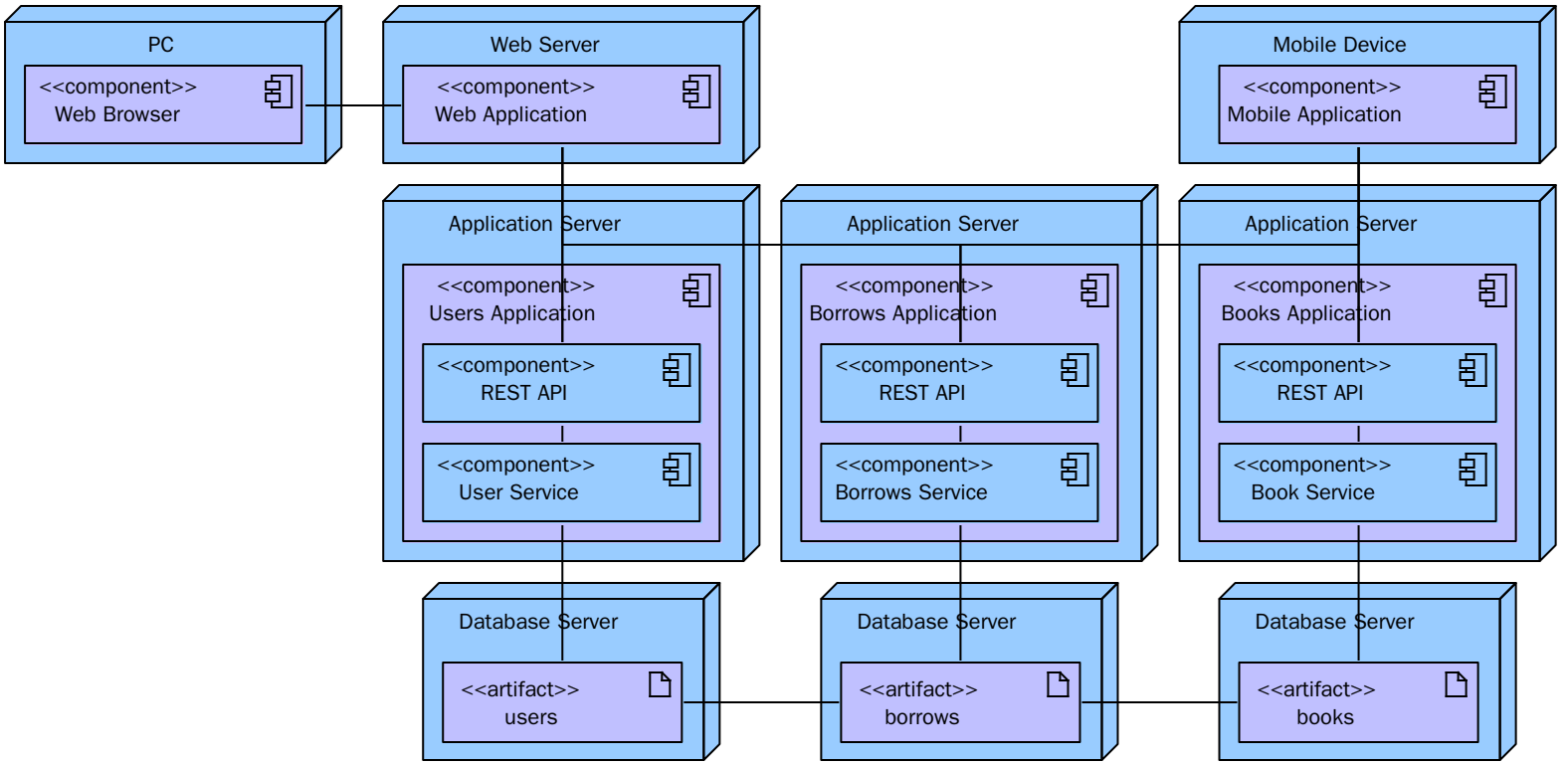
- performance by using multiple machines,
- inside private network non public ports can be used,
- outside private network additional security mechanism can be required.

Separating database servers

If application servers are separated why not database servers?

POLITECHNIKA GDAŃSKA

Separated Databases



Separated Databases

Using **separated database**:

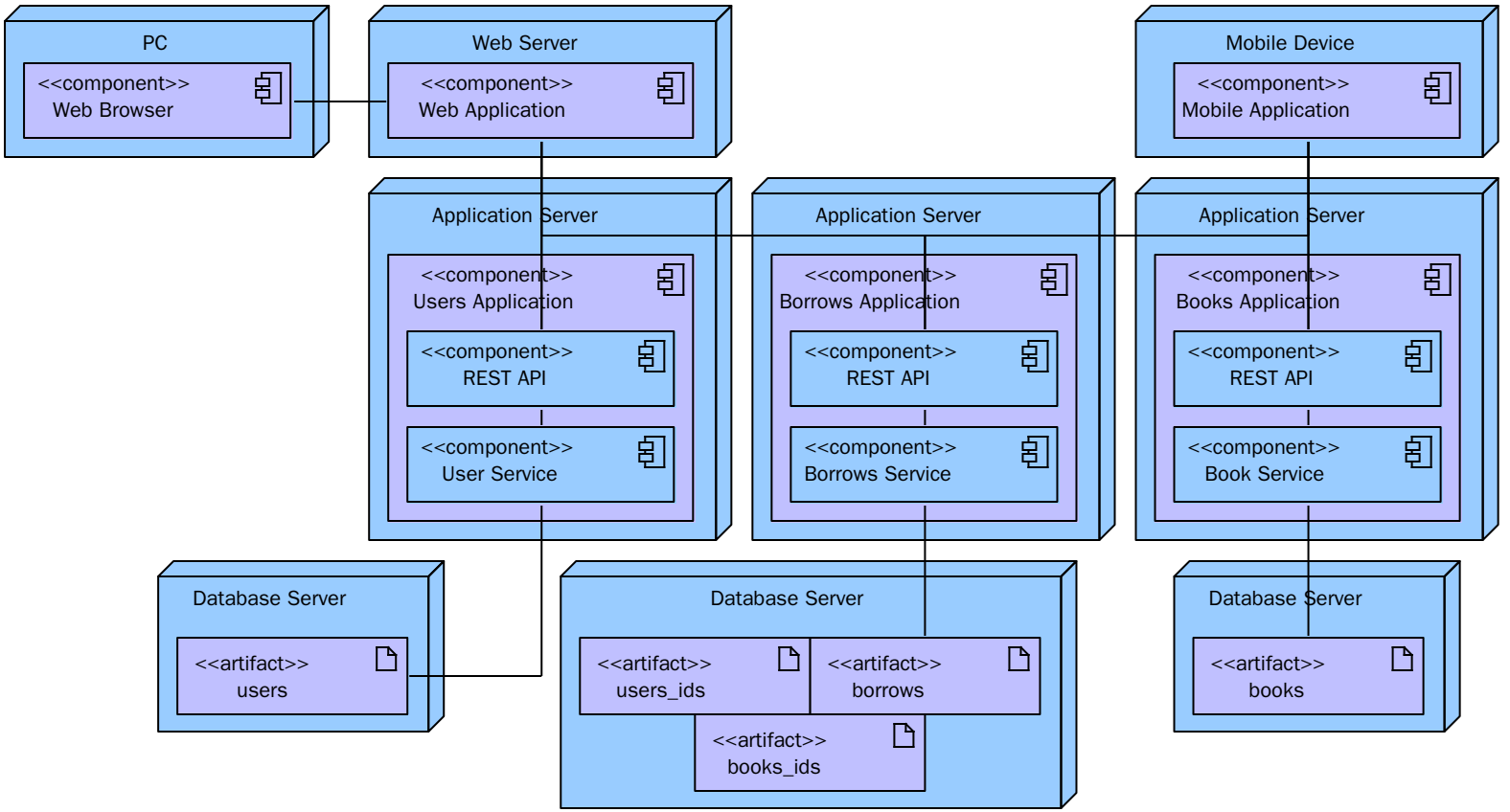
- each service can use different architecture which is best for its purposes (e.g.: SQL, NoSQL, file system storage etc.),
- no ACID transactions or relationships unless we use distributed database with distributed transactions:
  - not always supported by NoSQL databases.

Relationships between data

What about relationships between data stored in different databases?

POLITECHNIKA GDAŃSKA

Duplicate Database Entries



Duplicate Database Entries

Some data may need to be **duplicated**:

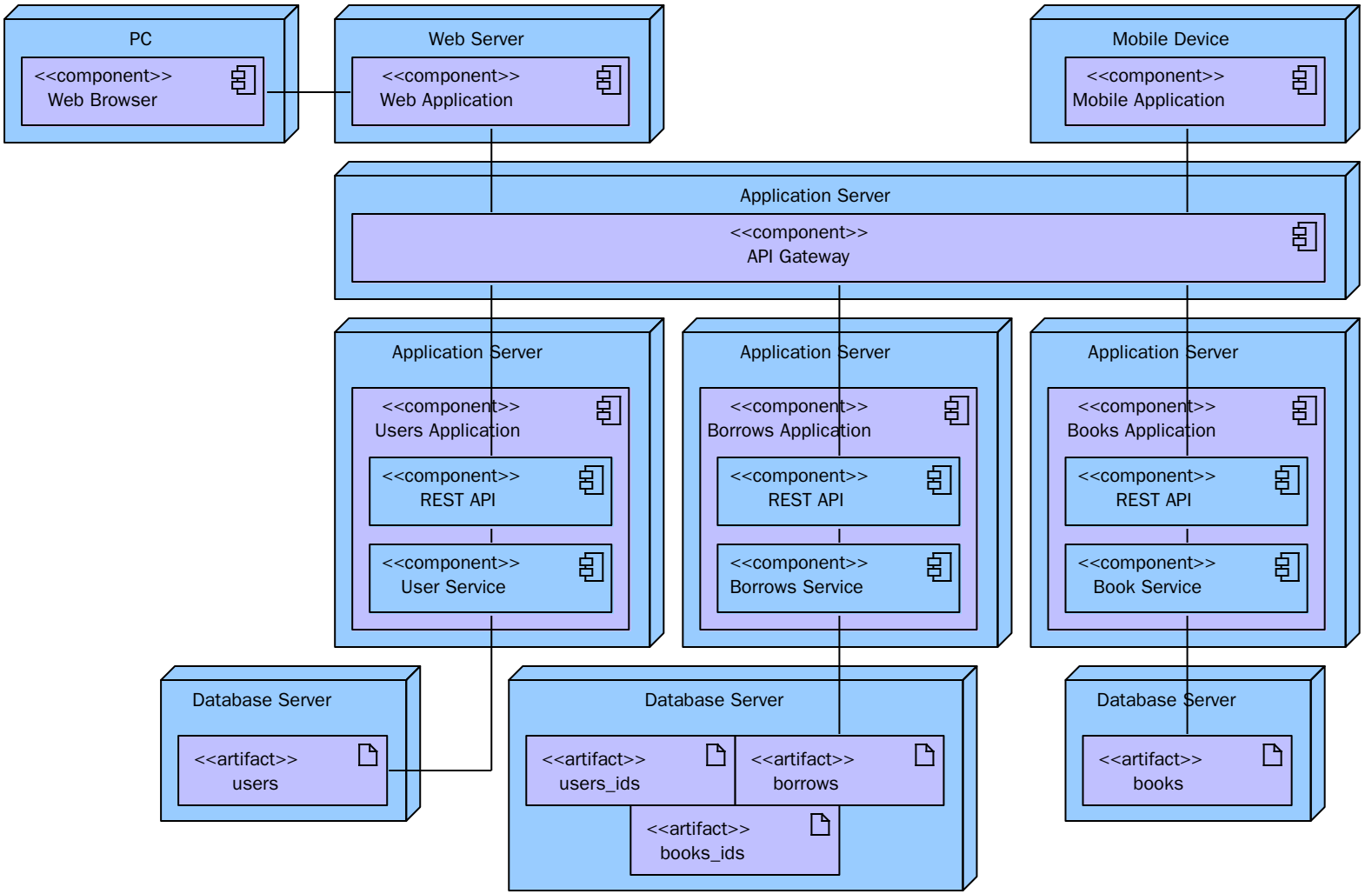
- relationships between objects,
- store only identifiers, not whole data,
- synchronization between modules is required:
  - it can be done by sending synchronization events between modules.

Locale all the services

When we have a number of web services, do we need to remember them all to use?

POLITECHNIKA GDAŃSKA

API Gateway



API Gateway

Using **API Gateway**:

- clients do not need to locate all the services,
- there can be different instances providing API for different clients,
- data from number of services can be joined by the gateway,
- another module to maintain,
- additional communication.

Helpful Sources

- Chris Richardson, *Microservice Architecture*, https://microservices.io/.