Case Study

Memi Lavi
www.memilavi.com

Case Study

- Use what we learned in a real-world system
- Go through main steps of the architecture process:
  - Functional Requirements
  - Non-Functional Requirements
  - Mapping the Components
  - Defining Communication

Case Study

- We won't discuss technology stack
  - Not specifically relevant to Microservices
- By the end of this section:

Download the Architecture Diagram!

MyLib

- Library management
- Manages books inventory
- Manages books' borrowing
- Manages customers
- Display notifications (late returns etc.)
- Charges annual fee

MyLib

Requirements

Functional

What the system should do

1. Web Based
2. Manage books inventory
3. Manage books' borrowing
4. Manage customers
5. Display notifications
6. Charge annual fee

Non-Functional

What the system should deal with

MyLib

NFR - What We Ask

1. *"How many expected concurrent users?"*	10
2. *"How many books will be managed?"*	10,000
3. *"How many borrowings in a day?"*	500
4. *"What's the desired SLA?"*	9/5

MyLib

Data Volume

- 1 book record = 1KB
- 10,000 books = 10MB
- 1 borrowing record = 500 bytes
- 500 borrowing / day ->
  - ~182K borrowing / year -> ~91MB / Year

MyLib

Requirements

Functional

What the system should do

1. Web Based
2. Manage books inventory
3. Manage books' borrowing
4. Manage customers
5. Display notifications
6. Charge annual fee

Non-Functional

What the system should deal with

1. 10 Concurrent users
2. 10,000 books
3. 182K borrowing / year
4. ~100MB / year
5. SLA is not very important

Mapping the Components

Microservices attribute #2:

"Organized Around Business Capabilities"

We have well-defined entities, so…

Let's try and design the services around them

Business Entities

Books

Borrowing

Customers

Utilities

Notifications

Payments (External System)

View

Note: For the sake of simplicity, we'll not discuss logging & monitoring

Books Service

- Used to manage the books inventory in the library
- Used by the librarians
- Has a storage (=database)
- Should be synchronous (immediate response)
- Does not depend on other services

Books Service API

- We'll use REST API
  - Because we need immediate response
- Main functionalities:
  - Get book details
  - Add new book
  - Remove book
  - Update book

Books Service API

Functionality	Path	Return Codes
Get book details	`GET /api/v1/book/{bookId}`	`200 OK` `404 Not Found`
Add new book	`POST /api/v1/book`	`200 OK`
Update book	`PUT /api/v1/book/{bookId}`	`200 OK` `404 Not Found`
Remove book	`DELETE /api/v1/book/{bookId}`	`200 Ok` `404 Not Found`

Borrowing Service

- Used to manage the borrowing of books
- Used by the librarians
- Has a storage (=database)
- Should be synchronous (immediate response)
- Does not depend on other services
  - (although refers to the books & customers DB)

Borrowing Service API

- We'll use REST API
  - Because we need immediate response
- Main functionalities:
  - Add a borrowing
  - Add a book return
  - Get borrowing by customer

Borrowing Service API

Functionality	Path	Return Codes
Get borrowing by customer	`GET /api/v1/borrowing/{customerId}`	`200 OK` `404 Not Found`
Add new borrowing	`POST /api/v1/borrowing`	`200 OK`
Add new book return	`POST /api/v1/borrowing`	`200 OK`

Customers Service

- Used to manage the library's customers
- Used by the librarians
- Has a storage (=database)
- Should be synchronous (immediate response)
- Does not depend on other services
    - (although refers to the books & borrowing DB)

Customers Service API

- We'll use REST API
  - Because we need immediate response
- Main functionalities:
  - Add a customer
  - Remove a customer
  - Get customer details
  - Get customer's borrowed books

Customers Service API

Functionality	Path	Return Codes
Get customer details	`GET /api/v1/customer/{customerId}`	`200 OK` `404 Not Found`
Get customer's borrowed books	`GET /api/v1/customer/{customerId}/books`	`200 OK` `404 Not Found`
Add new customer	`POST /api/v1/customer`	`200 OK`
Remove customer	`DELETE /api/v1/customer/{customerId}`	`200 OK`

Notifications Service

- Used to send notifications to the library's customers
  - Book was returned, New books added, etc.
- Used by the other services
- Asynchronous
- Might experience load (ie. Send notifications to all the customers)

Notifications Service API

- We'll use Queue
  - No immediate response required
  - Distributes load

Payments Service

- Used to send payments instructions to external service
- Used by other services (for example – a new customer joins the library)
- Asynchronous
- High-reliability is a must

Payments Service API

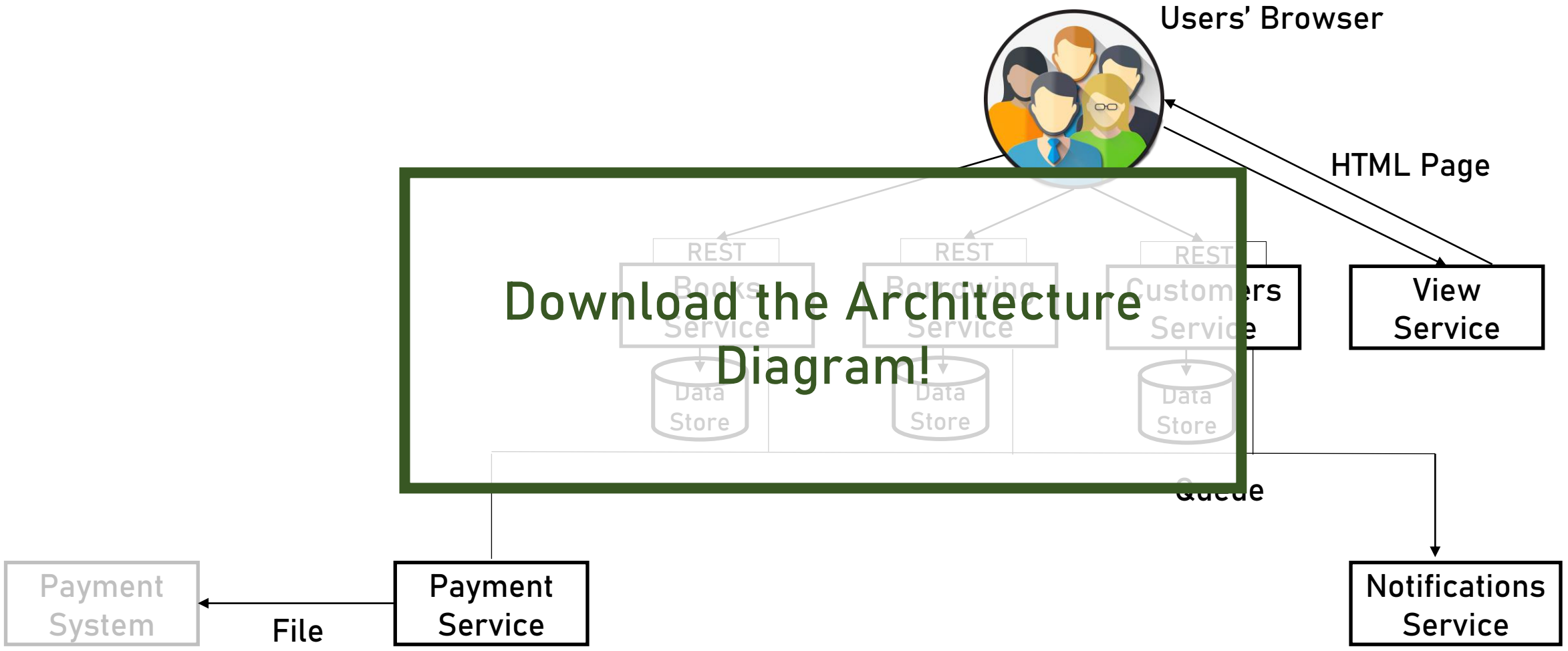
- We'll use Queue
  - No immediate response required
  - High-availability

View Service

- Used to serve static content to the web browser
  - HTML, CSS, JS files
- Very basic
- No API required, works directly with the browser

MyLib

Architecture Diagram



Download the Architecture Diagram!