



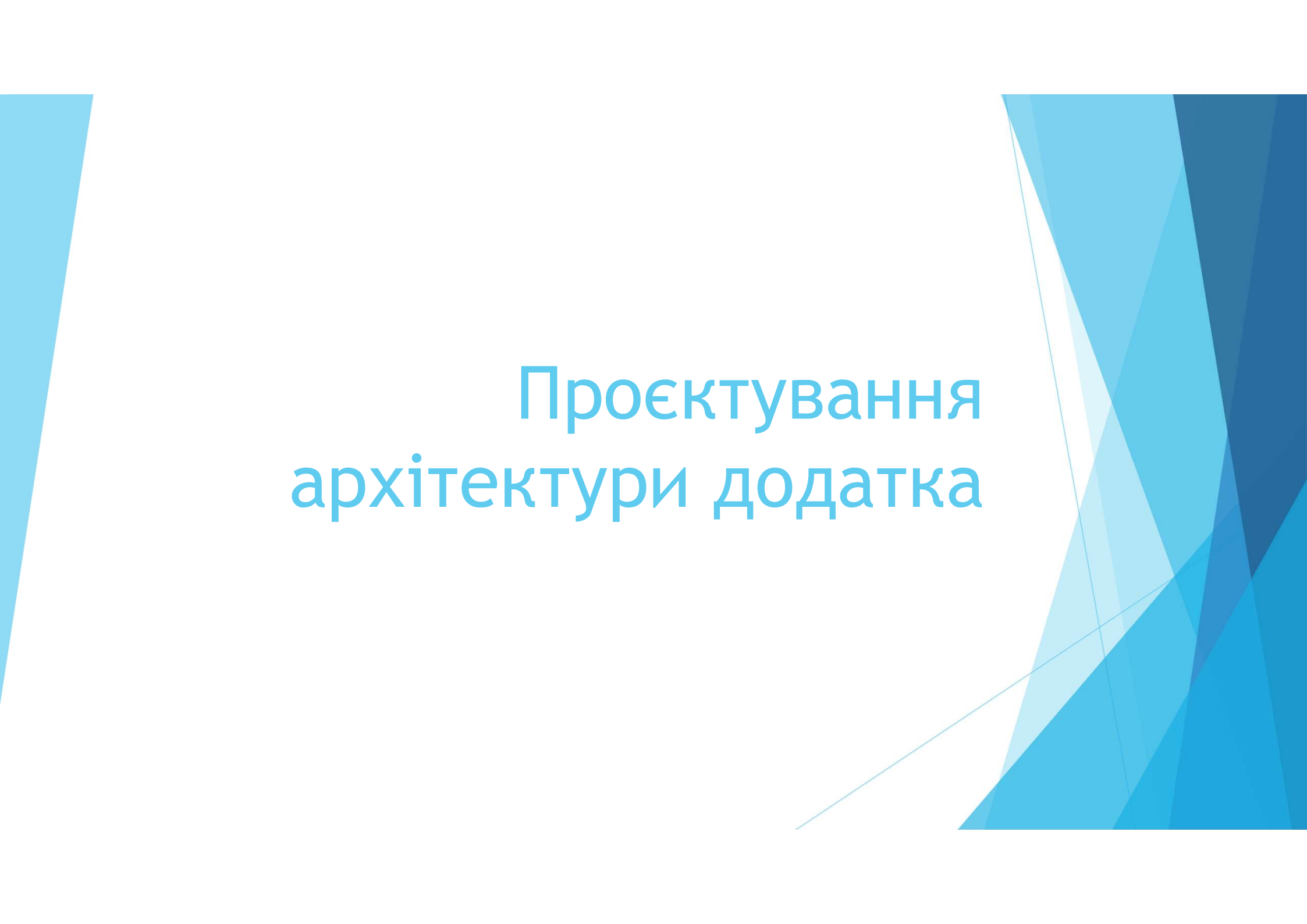
**Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»**

Інформаційні системи

Викладач: к.т.н., доц. Саяпіна Інна Олександрівна

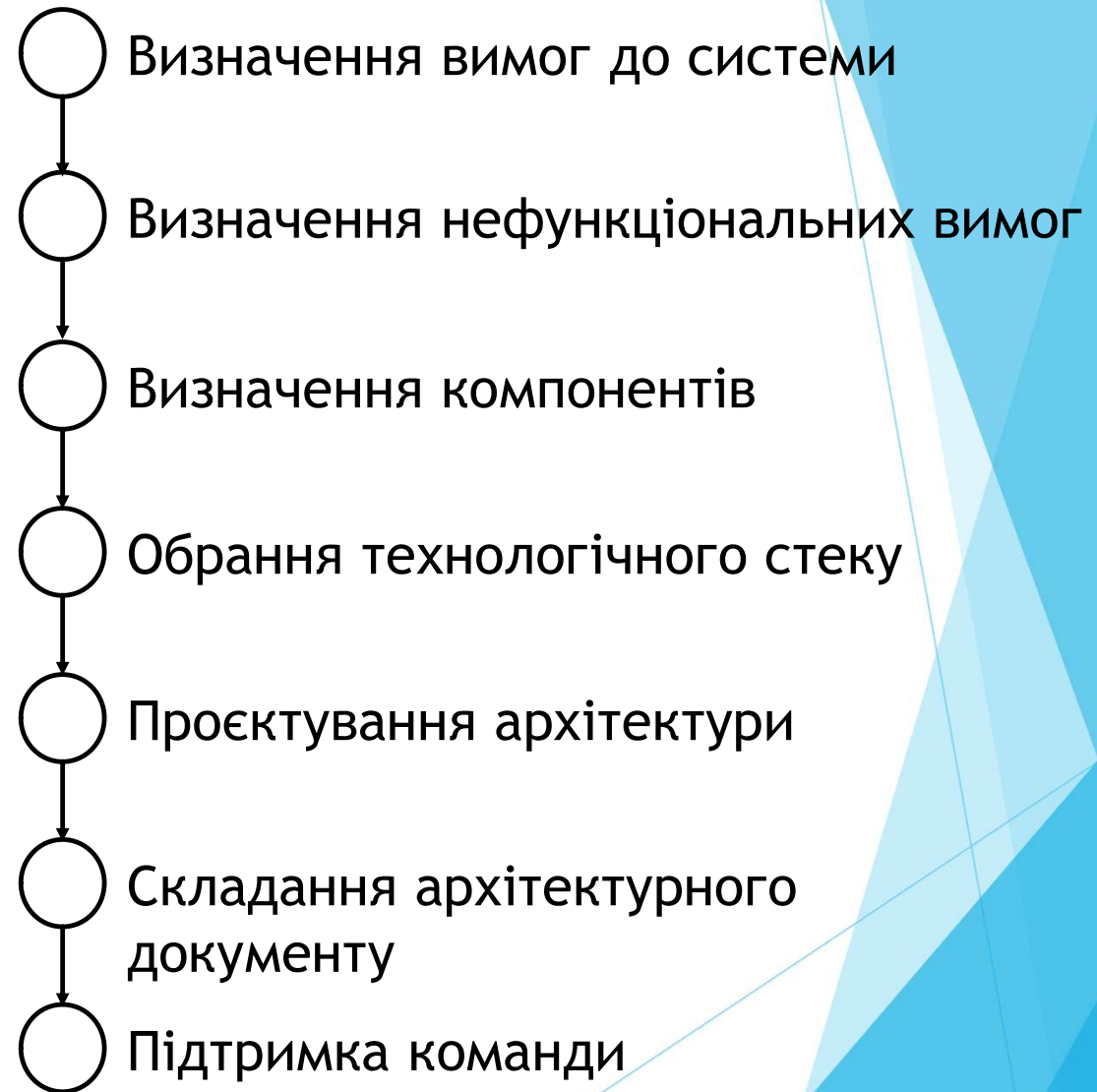
План заняття:

- ▶ Проєктування архітектури інформаційної системи
- ▶ Case#1. Paperbox



Проектування архітектури додатка

Процес архітектури



Визначення вимог до системи

Визначення вимог до системи

Визначення нефункціональних вимог

Визначення компонентів

Обрання технологічного стеку

Проектування архітектури

Складання архітектурного документу

Підтримка команди

➤ Вимоги = Що має робити система

➤ Зазвичай визначається системним аналітиком

○ Визначення вимог до системи

● Визначення нефункціональних вимог

○ Визначення компонентів

○ Обрання технологічного стеку

○ Проєктування архітектури

○ Складання архітектурного документу

○ Підтримка команди

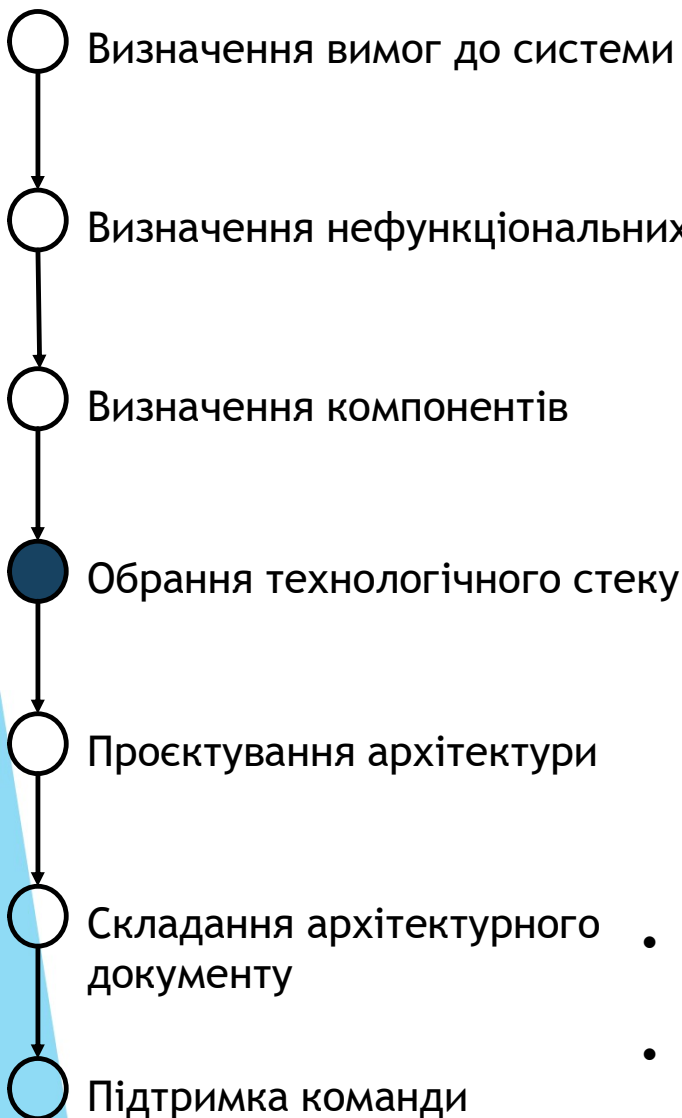
Визначення нефункціональних вимог

- Визначте атрибути технічного рівня та рівня обслуговування
 - тобто, кількість користувачів, обсяги навантажень, продуктивність
- Не завжди відомі клієнту чи аналітику
- Набагато важливіші, ніж звичайні вимоги



Визначення компонентів

- Представлення завдань системи
- Дві цілі:
 - Визначення функціональних можливостей системи
 - Донесення до клієнта свого розуміння системи
- Нетехнічний опис

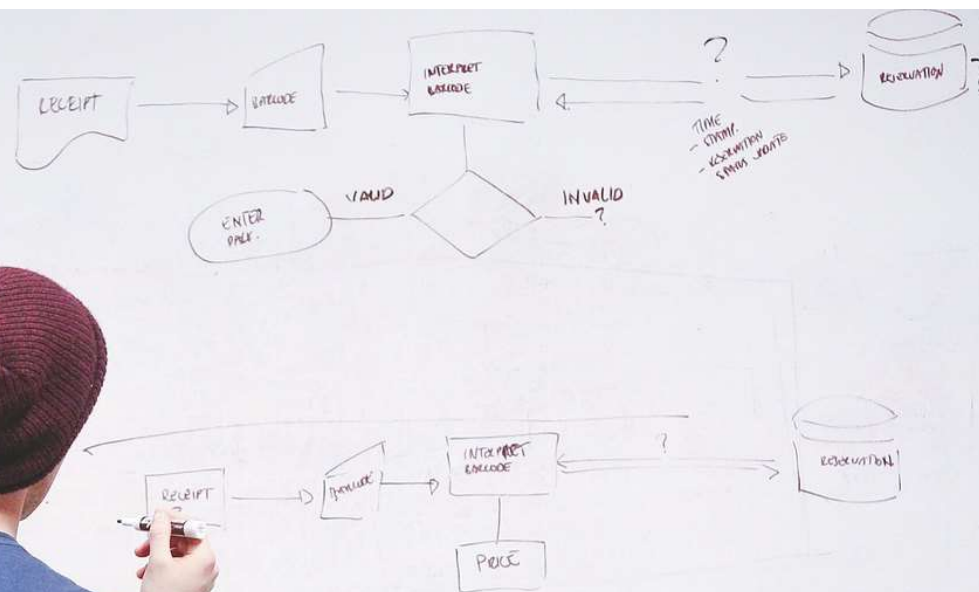
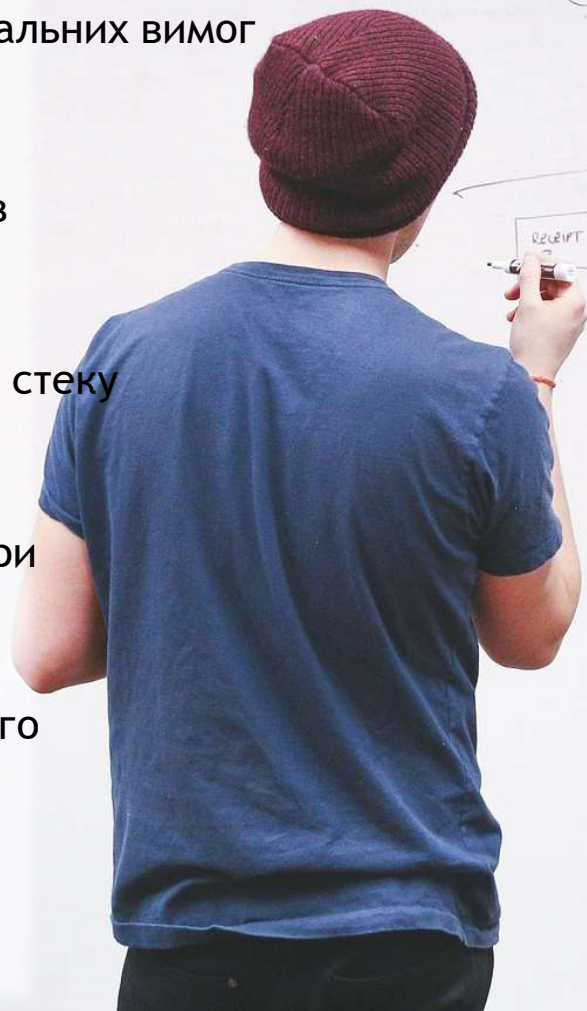


Обрання технологічного стеку



- Зазвичай, обирається платформа для Back End, Front End, Data Store
- Багато факторів, вибирайте з розумом!

- Визначення вимог до системи
- Визначення нефункціональних вимог
- Визначення компонентів
- Обрання технологічного стеку
- Проектування архітектури**
- Складання архітектурного документу
- Підтримка команди



Проектування Архітектури

Визначення вимог до системи

Визначення нефункціональних вимог

Визначення компонентів

Обрання технологічного стеку


Проектування архітектури

Складання архітектурного документу

Підтримка команди

- Описує процес і архітектуру
- Має бути актуальним для всіх учасників

Складання Архітектурного документу



Визначення вимог до системи

Визначення нефункціональних вимог

Визначення компонентів

Обрання технологічного стеку

Проектування архітектури

Складання архітектурного документу

Підтримка команди

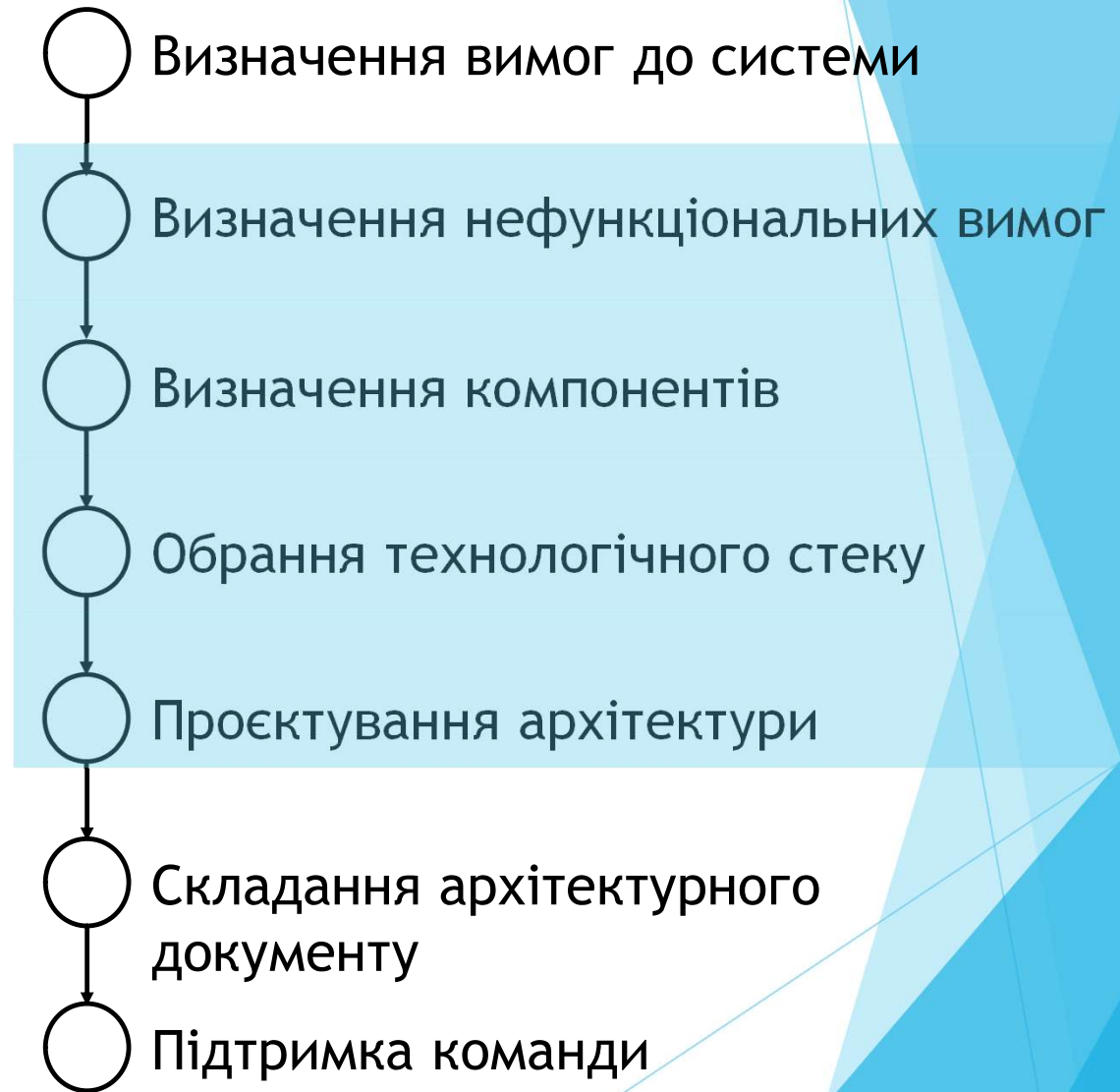
Підтримка Команди

- Архітектура сильно зміниться
- Переконайтеся, що вона залишатиметься
актуальною

Висновок



Процес архітектури



Case #1



Paperbox

- Продає паперові вироби
 - Папір для принтера, конверти тощо.
- Потрібна нова система управління кадрами
- Управління персоналом, зарплати, відпустки, виплати



Вимоги

Функціональні

Що повинна робити

система

1. Веб-інтерфейс
2. Виконувати операції CRUD над співробітниками
3. Керувати зарплатами:
 1. Дозволяти менеджеру подавати зпит змінити зарплату співробітнику
 2. Дозволяти HRy схвалити / відхилити запит
4. Керувати днями відпустки
5. Використовувати зовнішню платіжну систему

Нефункціональні

Як гарно вона це повинна
робити

НФВ – Що ми знаємо

1. Класична інформаційна система
2. Небагато користувачів
3. Не так багато даних
4. Інтерфейс із зовнішньою системою



НФВ – Які є питання

1. *«Скільки очікується одночасних користувачів?»* 10
2. *«Скільки працівників?»* 250
3. *«Що ми знаємо про зовнішню платіжну систему?»*

Платіжна система

- Застаріла система, написана мовою C++
- Розміщується на сервері компанії
- Input – тільки файли
- Файл надходить раз на місяць

Обсяг даних

- 1 співробітник = ~1 Мб даних
- Кожен співробітник має ~10 сканованих документів
(контракт, відгуки тощо)
- 1 сканований документ = ~5 Мб
- Загальний обсяг пам'яті для 1 співробітника = ~51 МБ

Обсяг даних (2)

- За 5 років компанія планує зрости до 500 співробітників
- Загальний обсяг пам'яті: 51 МБ X 500 співробітників = 25,5 ГБ
- Не багато, але:
 - Необхідно враховувати зберігання документів

SLA

4. «Наскільки критична система?»

Не дуже
критична

Вимоги

Функціональні

Що повинна робити

система

1. Веб-інтерфейс
2. Виконувати операції CRUD над співробітниками
3. Керувати зарплатами:
 1. Дозволяти менеджеру подавати зпит змінити зарплату співробітнику
 2. Дозволяти HRy схвалити / відхилити запит
4. Керувати днями відпустки
5. Використовувати зовнішню платіжну систему

Нефункціональні

Як гарно вона це повинна
робити

1. 10 одночасних користувачів
2. Керує 500 користувачами
3. Прогнозований обсяг даних: 25,5 ГБ
 1. Реляційний і неструктурований
4. Не критична
5. Файловий інтерфейс

Компоненти

Виходячи з вимог:

1. Сутності: Працівники, Відпустка, Зарплата
2. Інтерфейс до платіжної системи

Payment
System

Payment
Interface

Відправляє
платіжні дані
в платіжну
систему

Employees
Service

Виконує
операції CRUD
щодо
співробітників

Salary
Service

Процес
затвердження
зарплати

Vacation
Service

Управління
відпустками
співробітників

View
Service

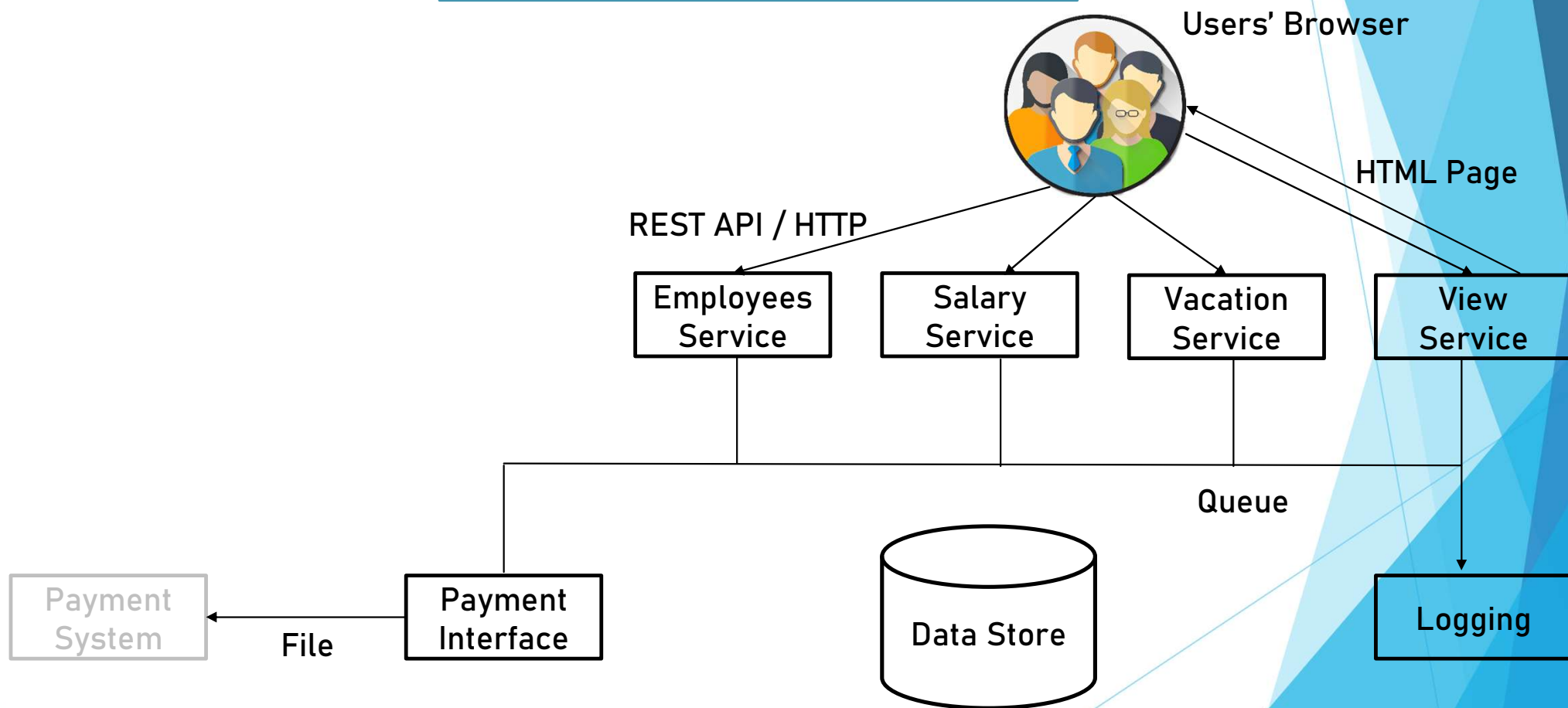
Повертає
статичні файли
в браузер
(HTML, CSS, JS)

Data Store

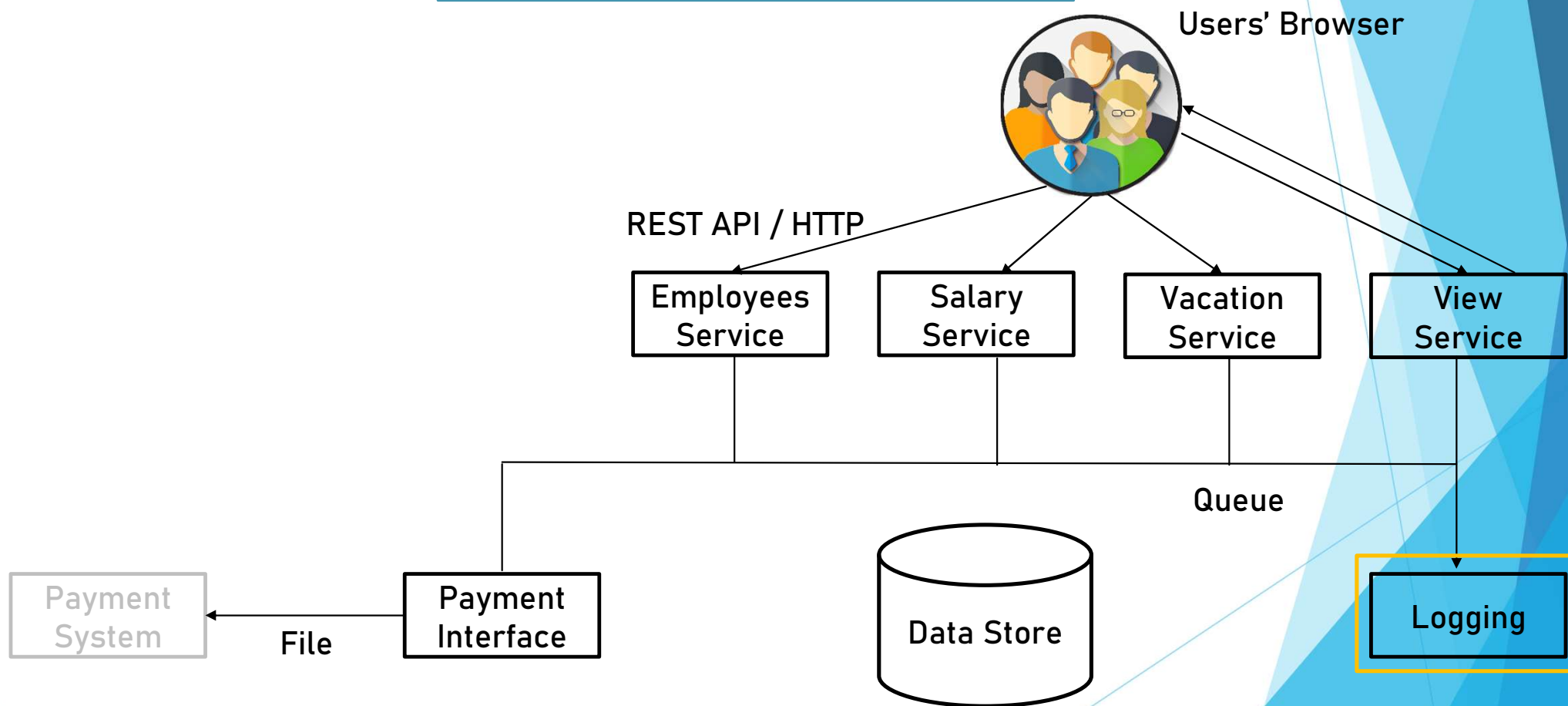
Logging

Питання: одне сховище даних або сховище даних для однієї служби?
А: Дані спільно використовуються між службами, тому єдине сховище даних краще

Обмін повідомленнями



Компоненти



Logging Service

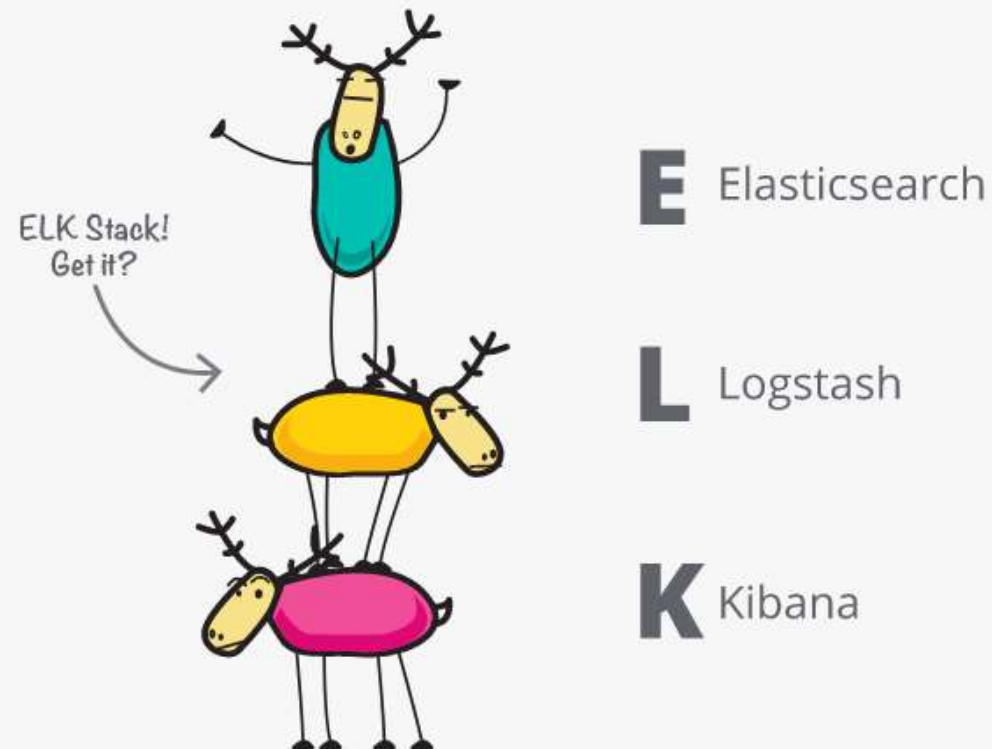
- Дуже важливий
- Його використовують інші сервіси

Logging - Questions

1. Чи існує в компанії механізм логування?
2. Розробити власний або
використовувати зовнішній?

Немає

Logging - Alternative



Logging - Alternative

ELK:

- Потужне сховище даних (**Elastic**)
- Імпорт журналу з багатьох джерел (**Logstash**)
- Чудовий переглядач із можливістю фільтрації
(**Kibana**)

Logging - Alternative

Але:

- Вимагає обслуговування
- Досить складний у встановці та налаштуванні
- Підходить в основному для великих систем з інтенсивним об'ємом даних

НЕ ВАРІАНТ

Logging Service

Кроки:

- Визначитись з типом програми
- Обрати стек технологій
- Спроектувати архітектуру

Тип програми

Що він робить:

- Читає записи логу з черги
- Перевіряє записи
- Зберігає в сховищі даних

Тип програми

Веб-додаток. Веб-програма працює у веб-браузерах і доступ до неї здійснюється через Інтернет. Зазвичай він створюється з використанням таких веб-технологій, як HTML, CSS і JavaScript. Користувачі взаємодіють із програмою через інтерфейс користувача, представлений у браузері. Веб-програми не залежать від платформи, доступ до них можна отримати з різних пристроїв за допомогою браузера та підключення до Інтернету.

Консольний додаток. Консольна програма працює в текстовій консолі або інтерфейсі командного рядка (CLI). Він виконується з терміналу або командного рядка без графічного інтерфейсу користувача (GUI). Консольні програми зазвичай використовуються для завдань системного адміністрування, пакетної обробки або інструментів командного рядка. Зазвичай вони отримують вхідні дані від користувача та забезпечують текстове виведення.

Сервіс/Служба відноситься до фонових процесу або частини функціональності, яка працює безперервно без інтерфейсу користувача. Це може бути окрема служба або частина більшої програми. Служби часто виконують такі завдання, як обробка даних, обробка подій або надання API для взаємодії інших програм. Зазвичай вони працюють у фоновому режимі, незалежно від взаємодії користувача.

Тип програми

Мобільний додаток. Мобільний додаток, як випливає з назви, призначений для роботи на мобільних пристроях, таких як смартфони та планшети. Він розроблений спеціально для мобільних платформ, таких як iOS (iPhone/iPad) або Android.

Мобільні програми мають спеціальний інтерфейс користувача, оптимізований для сенсорної взаємодії та менших екранів.

Вони можуть отримати доступ до функцій пристрою, таких як GPS, камера, акселерометр і push-сповіщення.

Desktop програма. Desktop програма працює на настільному або портативному комп'ютері. Він встановлюється та виконується локально на машині користувача, надаючи спеціальний інтерфейс користувача. Програми для настільних ПК можуть використовувати всі можливості базової операційної системи та апаратного забезпечення. Вони можуть отримувати доступ до локальних файлових систем, взаємодіяти з периферійними пристроями та надавати більш багатий досвід користувача порівняно з веб-програмами чи мобільними додатками.

Тип програми

- Веб-додаток і веб-API ✗
- Мобільний додаток ✗
- Консоль ?
- Сервіс ?
- Desktop додаток ✗

Тип програми

- Веб-додаток і веб-API ✗
- Мобільний додаток ✗
- Консоль ✓
- Сервіс ✓
- Desktop додаток ✗

Технологічний стек

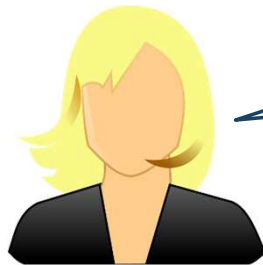
для:

- Коду компонента
- Сховища даних

Технологічний стек

Код повинен:

- Мати доступ до API черги
- Перевіряти дані
- Зберігати дані



Ми знайомі зі стеком Microsoft, тому є експертами в .NET і SQL Server

Технологічний стек



Архітектура

User Interface /
Service Interface

Business Logic

Data Access

Data Store



```
graph TD; A[Архітектура] --- B[User Interface / Service Interface]; B --- C[Business Logic]; C --- D[Data Access]; D --- E[(Data Store)]
```

Архітектура

User Interface /
Service Interface

Business Logic

Data Access

Data Store

```
graph TD; A[Архітектура] --- B[User Interface / Service Interface]; B --- C[Business Logic]; C --- D[Data Access]; D --- E[(Data Store)];
```

Logging Service

Polling

Business Logic

Data Access

Data Store

Ін'єкція залежності
використовуючи

Microsoft.Extensions.DependencyInjection

Використовує Entity
Framework

Опитує чергу кожні
кілька секунд на
предмет записів
журналу

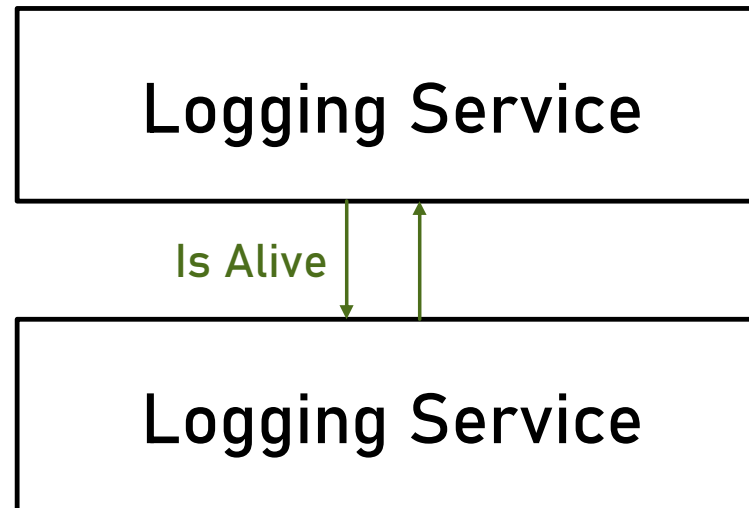
Перевіряє записи

Зберігає записи в
сховищі даних

Резервування Logging Service

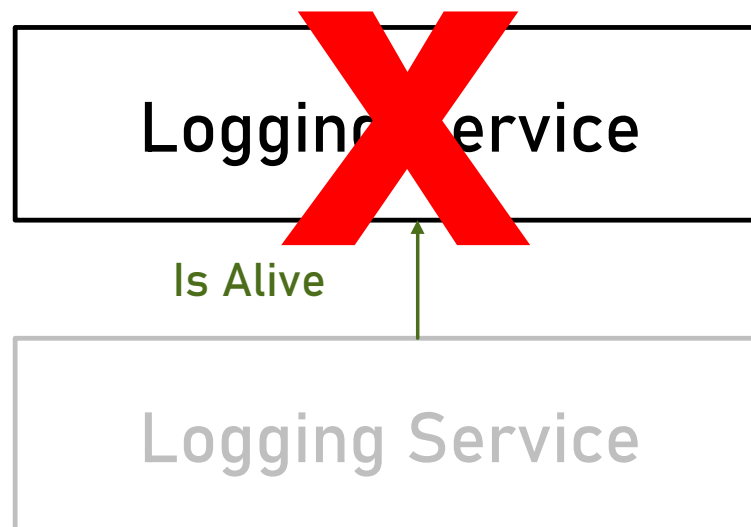
~~Logging Service~~

Резервування Logging Service

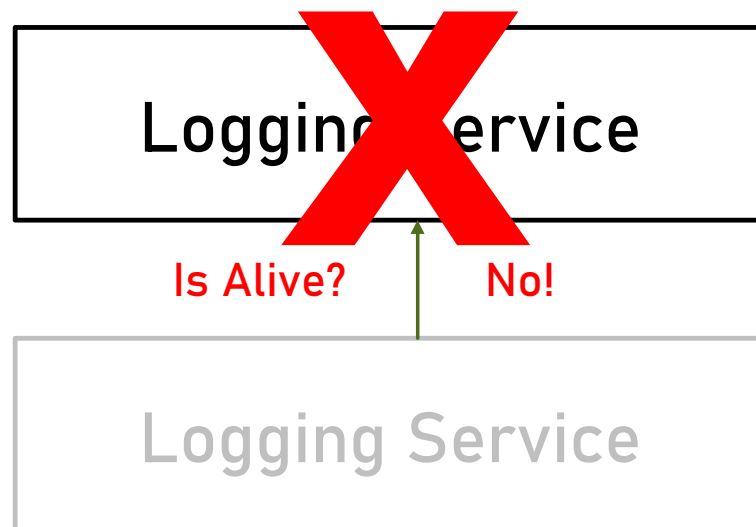


- Активний /
Активний
- Унікає
повторного
читання?

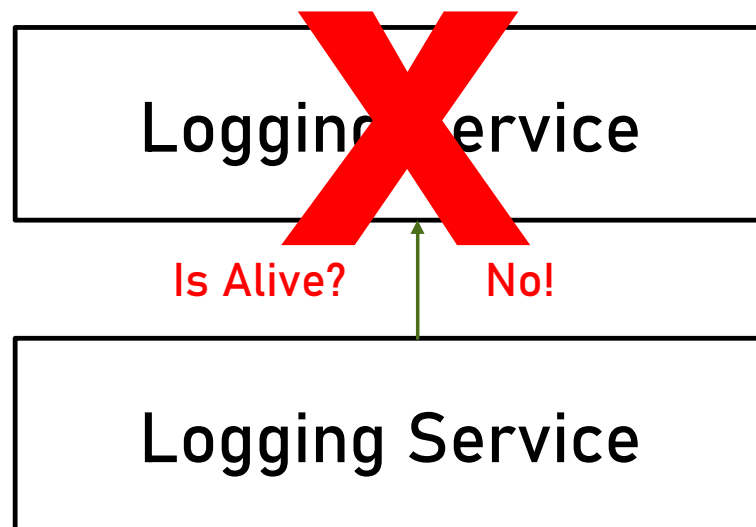
Резервування Logging Service



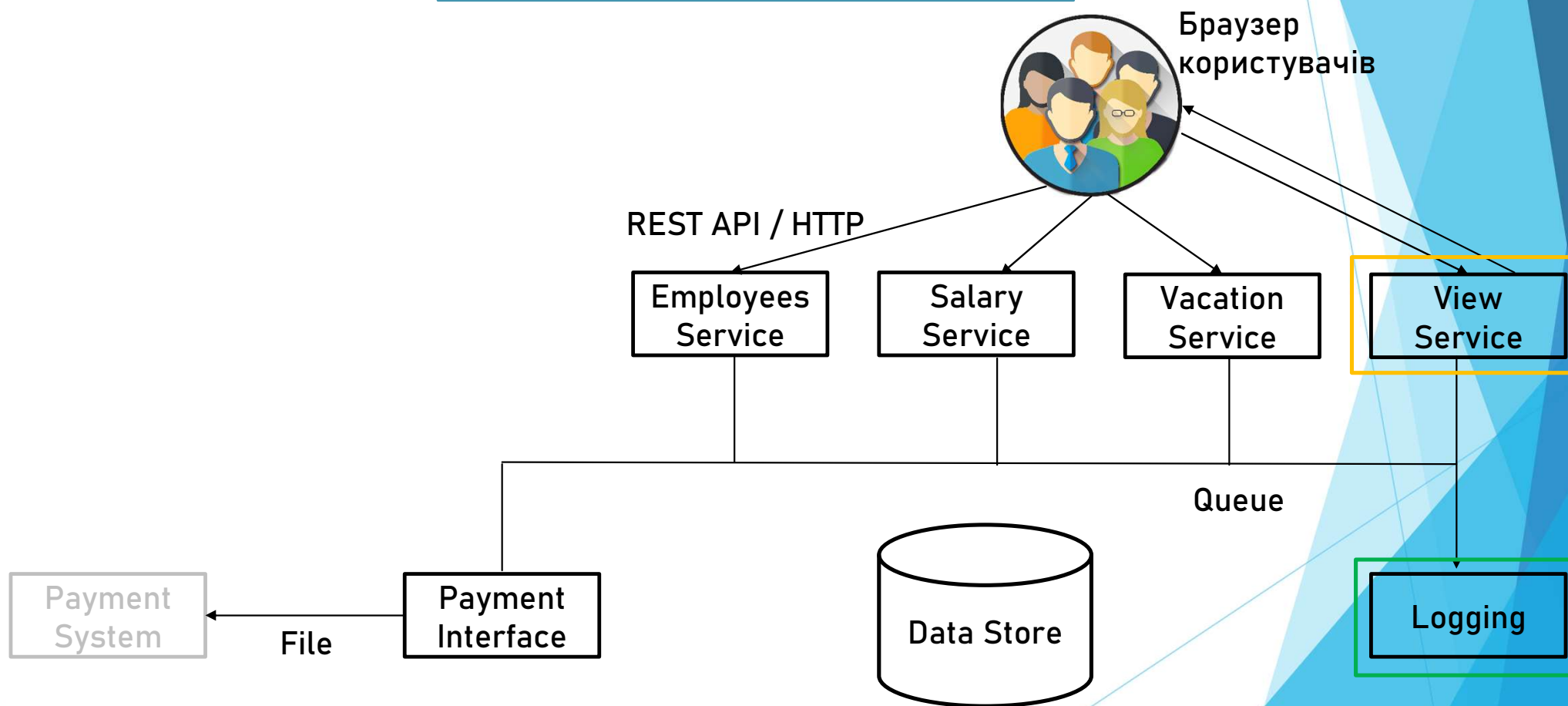
Резервування Logging Service



Резервування Logging Service



Компоненти



View Service

Що він робить:

- Отримує запити від браузерів кінцевих користувачів
- Повертає статичні файли (HTML / CSS / JS)

Тип програми

- Веб-додаток і веб-API ✓
- Мобільний додаток ✗
- Консоль ✗
- Сервіс ✗
- Desktop додаток ✗

Стек технологій

.NET Core чудово підтримує веб-програми

Стек технологій



Архітектура

User Interface /
Service Interface

Business Logic

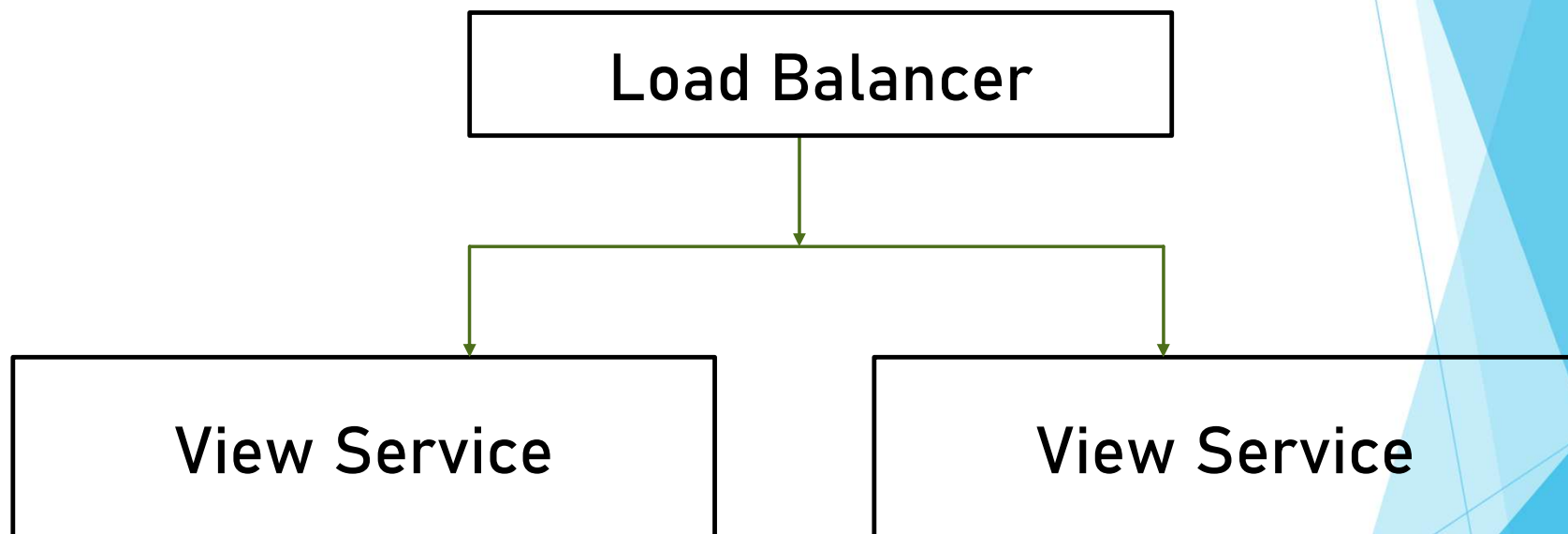
Data Access

Data Store

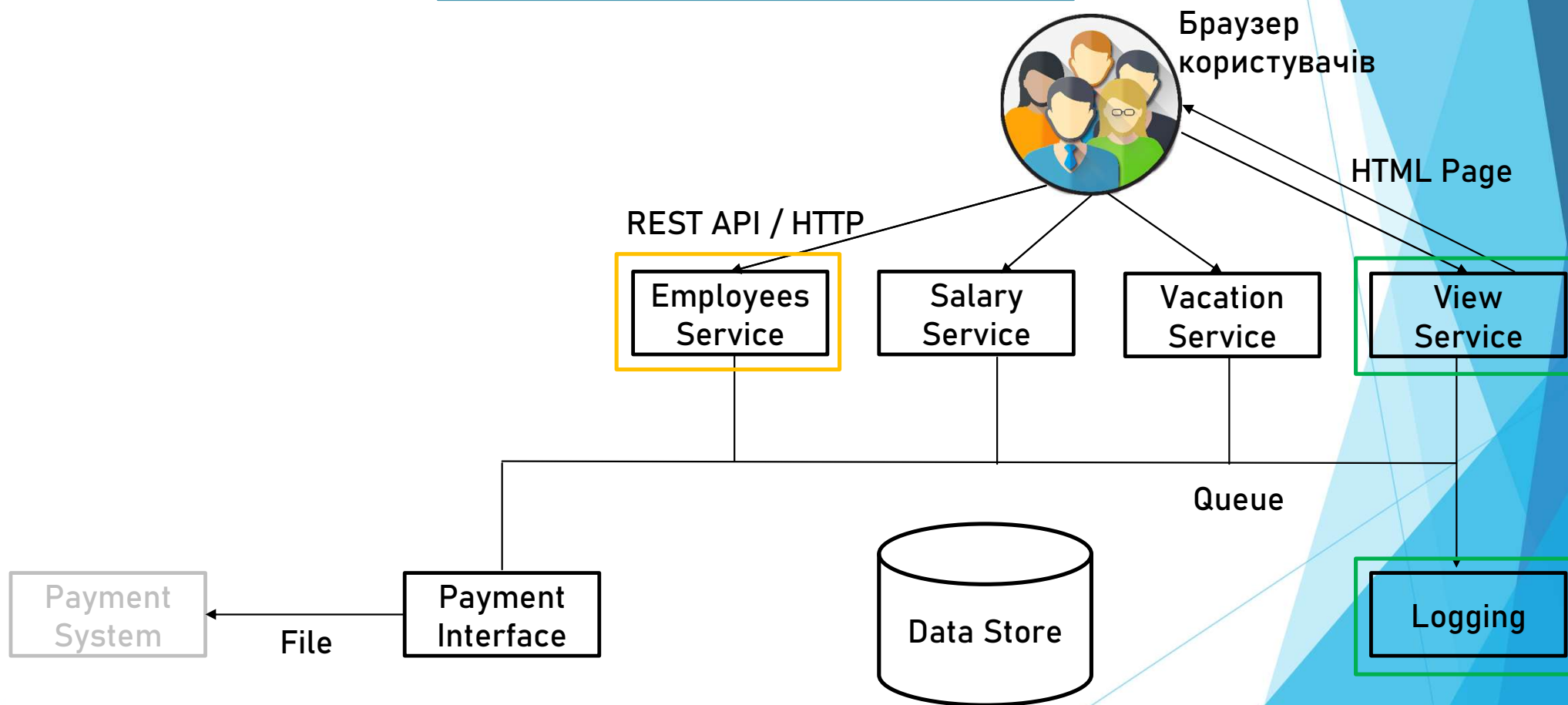
```
graph TD; A[Архітектура] --- B[User Interface / Service Interface]; B --- C[Business Logic]; C --- D[Data Access]; D --- E[(Data Store)];
```



Резервування View Service



Компоненти



Employees Service

Що він робить:

- Дозволяє кінцевим користувачам запитувати дані співробітників
- Дозволяє виконувати дії з даними (CRUD)

Що він не робить:

- Не відображає дані

Тип програми

- Веб-додаток і веб-API ✓
- Мобільний додаток ✗
- Консоль ✗
- Сервіс ✗
- Desktop додаток ✗

Технологічний стек – платформа
для розробників



Технологічний стек – база даних

Дані співробітника
(реляційні)

Документи



?

Технологічний стек – база даних

Альтернативи зберігання документів (BLOB).

Relational Database

File System

Object Store

Cloud Storage

Альтернативи зберігання документів (BLOB).

Alternative	Description	Examples	Pros	Cons
Relational Database	Store the document in a specialized column type designed for BLOBs	SQL Server's FILESTREAM, Oracle's BLOB type	Part of the app transaction Part of the DB's backup / DR	Clunky syntax, Limited size
File System	Store the document in a file, and hold a pointer to it in the DB	File System	Unlimited size Easy to execute	Not part of transaction, Unmanageable
Object Store	Use special type of store mechanism that specializes in BLOBs	CEPH	Great scale Unlimited size	Complex setup Dedicated knowledge New product in the mix
Cloud Storage	Store the documents in one of the public cloud storage mechanisms	Azure's Storage Account AWS's S3	Great scale Easy to execute	Requires internet connection Cost

Технологічний стек – база даних

Дані співробітника
(реляційні)

Документи



?

Технологічний стек – база даних

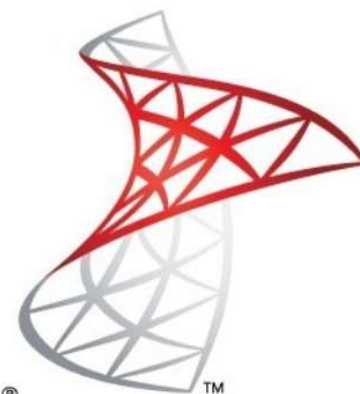
Дані співробітника
(реляційні)



Microsoft®
SQL Server®

Документи

- Документи малі (~1 МБ)
- Вже використовується
- Частина програми



Microsoft®
SQL Server®

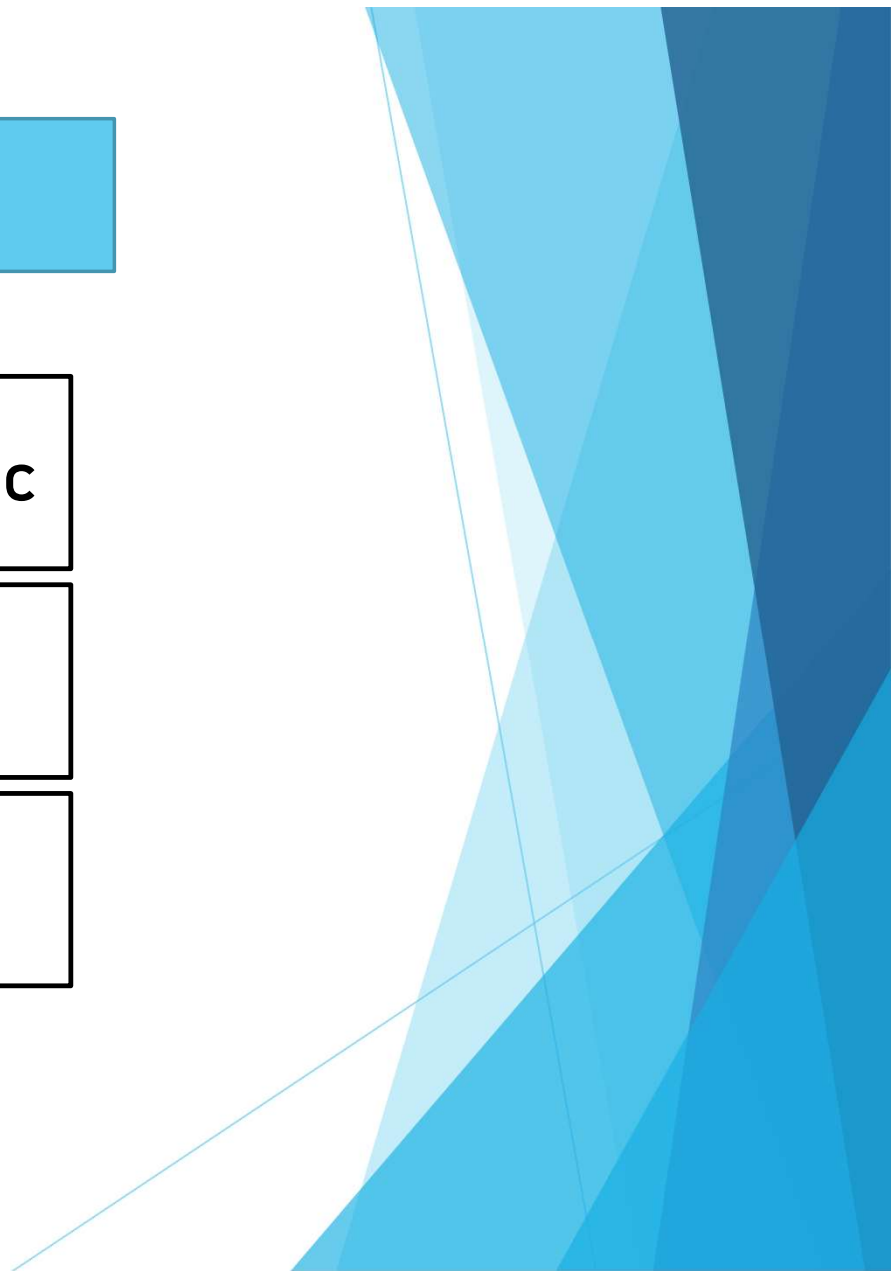
Архітектура

Сервісний інтерфейс

Бізнес-логіка

Доступ до даних

Сховище
даних



API

- Отримання повної інформації про співробітника за ідентифікатором
- Список працівників за параметрами
- Додавання співробітника
- Оновлення інформації про співробітника

**Не фізичне
видалення!**

API – Продовж.

- Додавання документу
- Видалення документу
- Отримання документу
- Отримання документів за параметрами

З: Чи потрібен нам окремий
Сервіс Обробника
документів?

А: Оскільки документи
потрібні лише для
сутності Employee, то
ні.

API

Functionality	Path	Return Codes
Get employee details by ID	GET /api/v1/employees/{id}	200 OK 404 Not Found
List employees by parameters	GET /api/v1/employees?name=...&birthdate=...	200 OK 400 Bad Request
Add employee	POST /api/v1/employees	201 Created 400 Bad Request
Update employee details	PUT /api/v1/employees/{id}	200 OK 400 Bad Request 404 Not Found
Remove employee	DELETE /api/v1/employees/{id}	200 OK 404 Not Found

API

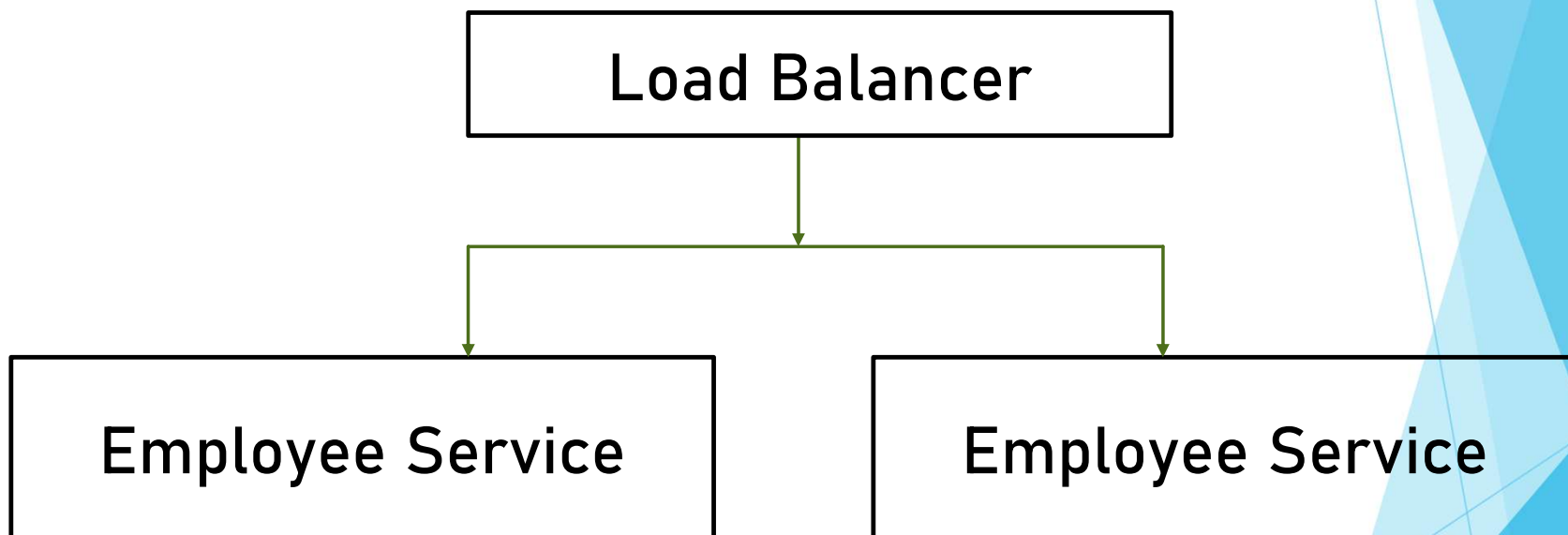
Functionality	Path	Return Codes
Add document	POST /api/v1/employees/{id}/document	201 Created 404 Not Found
Remove document	DELETE /api/v1/employees/{id}/document/{docid}	200 OK 404 Not Found
Get document	GET /api/v1/employees/{id}/document/{docid}	200 OK 404 Not Found
Retrieve documents for employee	GET /api/v1/employees/{id}/documents	200 OK 404 Not Found

Резервування Employee
Service

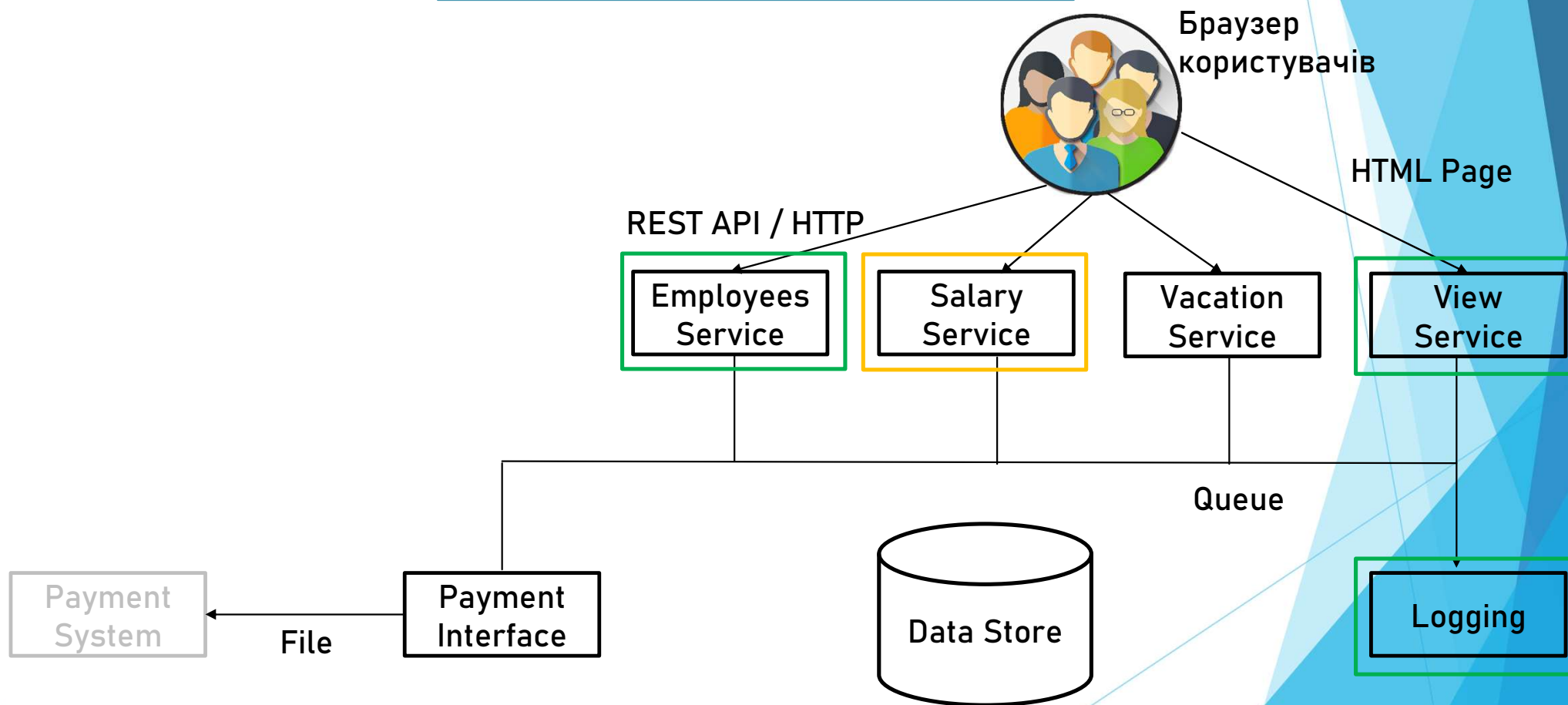
Load Balancer

Employee Service

Employee Service



Компоненти



Salary Service

Що він робить:

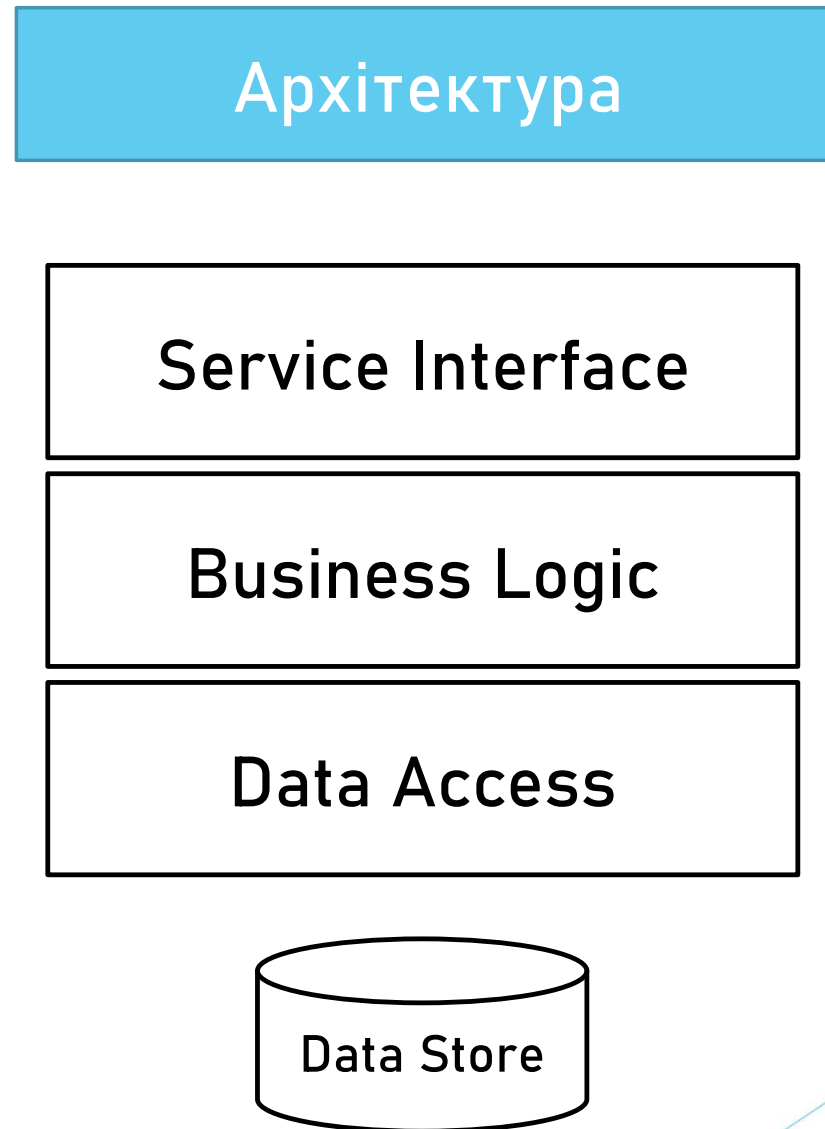
- Дозволяє менеджерам надсилати запит на зміну зарплати співробітника
- Дозволяє HRy схвалити/відхилити запит

Тип програми

- Веб-додаток і веб-API ✓
- Мобільний додаток ✗
- Консоль ✗
- Сервіс ✗
- Настільний додаток ✗

Стек технологій





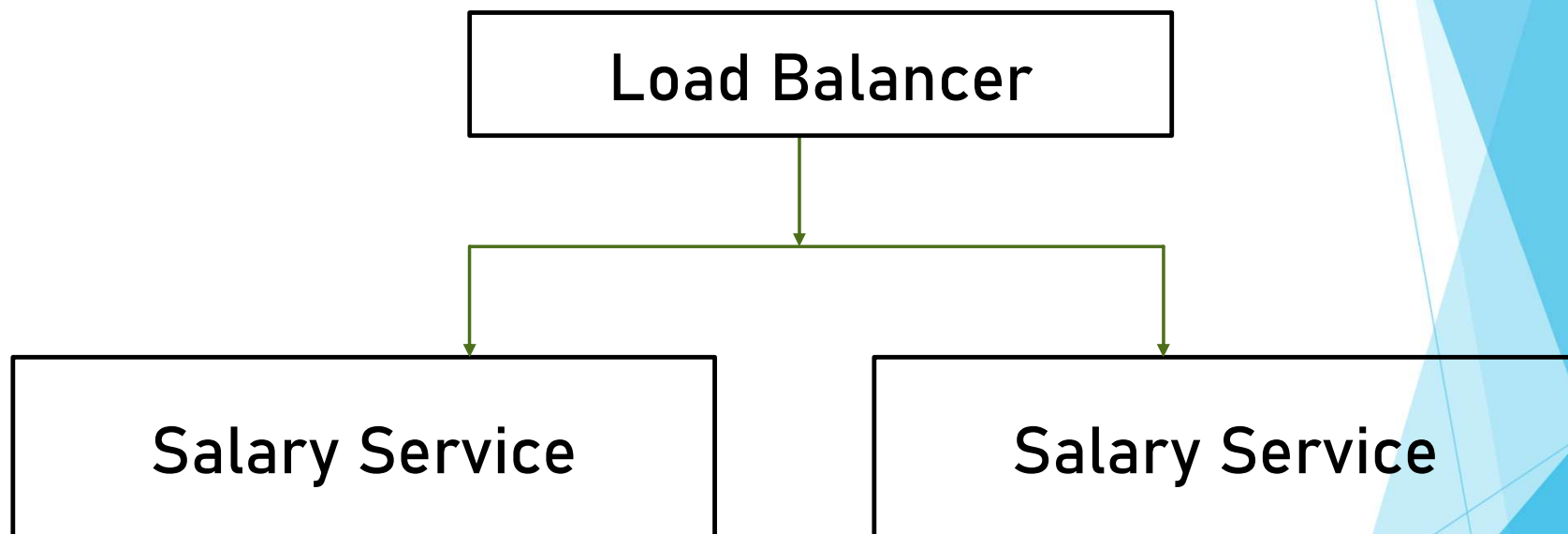
API

- Додавання запиту на зарплату
- Видалення запиту на зарплату
- Отримання запитів на зарплату
- Затвердження запиту на зарплату
- Відхилення запиту на зарплату

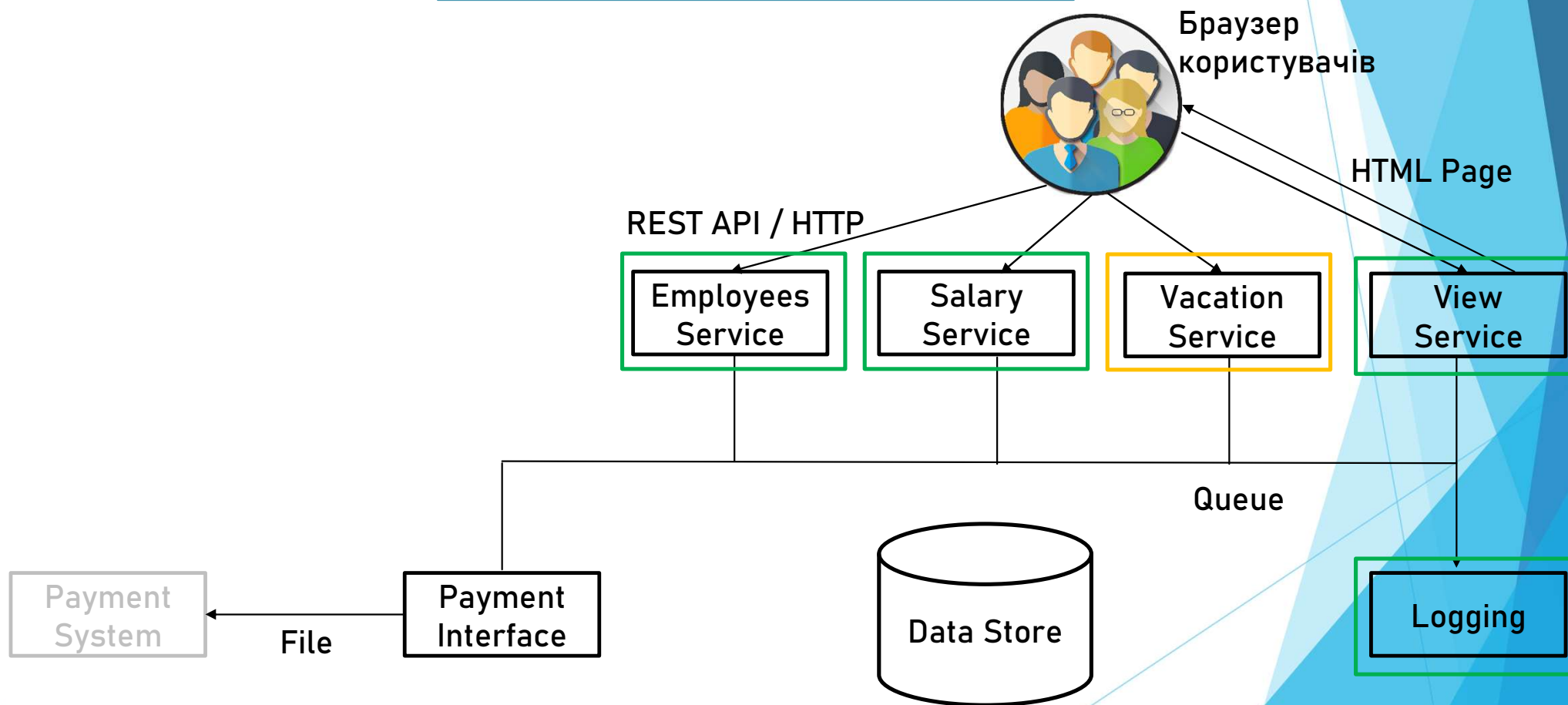
API

Functionality	Path	Return Codes
Додавання запиту на зарплату	POST /api/v1/salaryRequests/	200 OK 400 Bad Request
Видалення запиту на зарплату	DELETE /api/v1/salaryRequests/{id}	200 OK 404 Not Found
Отримання запитів на зарплату	GET /api/v1/salaryRequests	200 OK
Затвердження запиту на зарплату	POST /api/v1/salaryRequests/{id}/approval	200 OK 404 Not Found
Відхилення запиту на зарплату	POST /api/v1/salaryRequests/{id}/rejection	200 OK 404 Not Found

Резервування Salary Service



Компоненти



Vacation Service

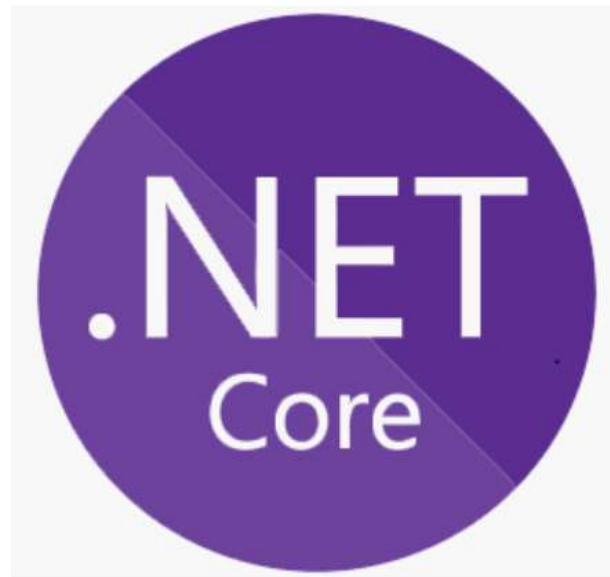
Що він робить:

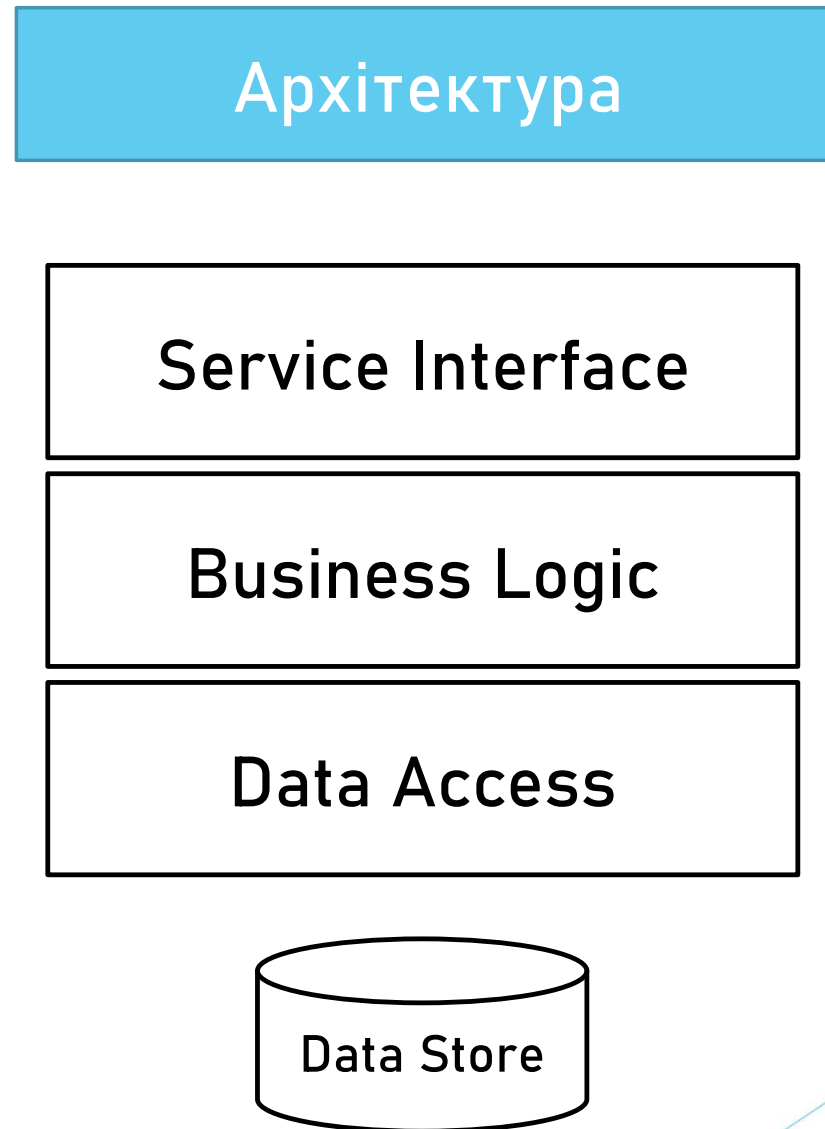
- Дозволяє працівникам керувати своїми днями відпустки
- Дозволяє відділу кадрів встановлювати дні відпустки для співробітників

Тип програми

- Веб-додаток і веб-API ✓
- Мобільний додаток ✗
- Консоль ✗
- Сервіс ✗
- Настільний додаток ✗

Стек технологій





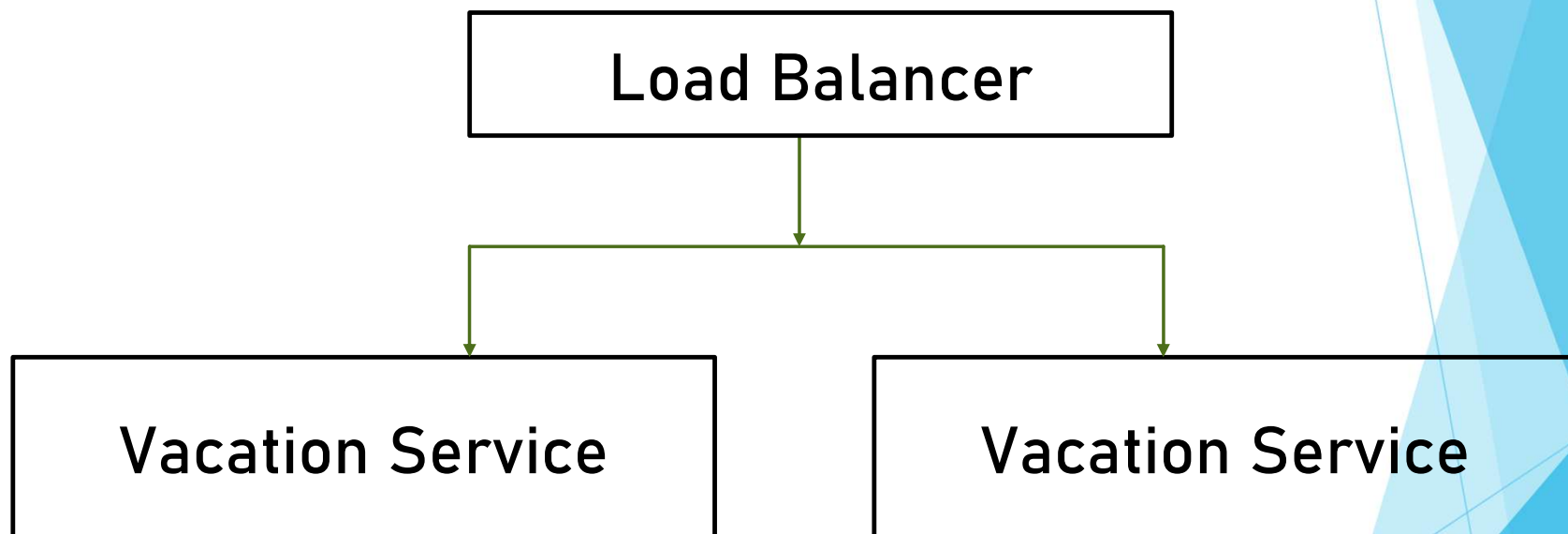
API

- Встановлення доступних днів відпустки (HRом)
- Отримання доступних днів відпустки
- Скорочення відпустки (працівниками)

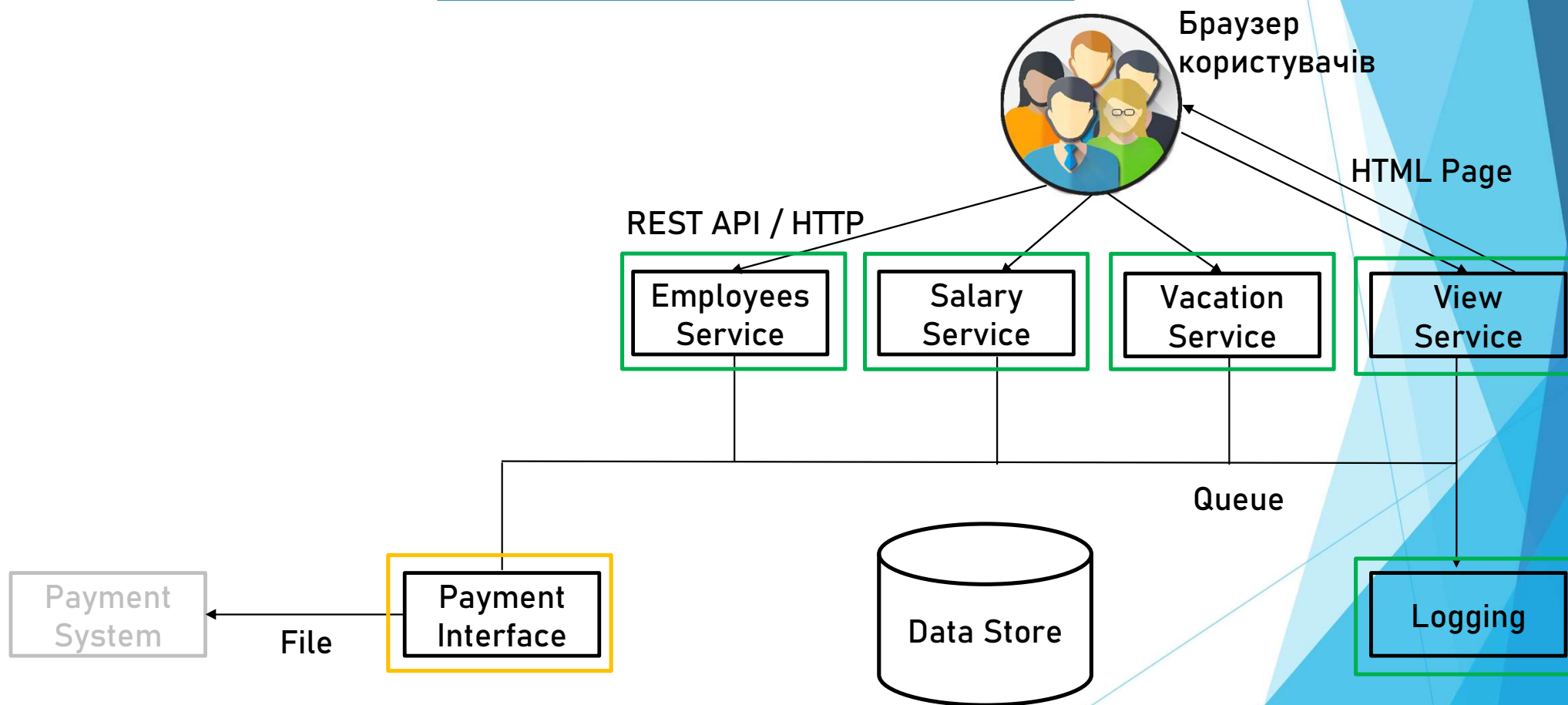
API

Functionality	Path	Return Codes
Встановлення доступних днів відпустки	PUT /api/v1/vacations/{empid}	200 OK 404 Not Found
Отримання доступних днів відпустки	GET /api/v1/vacations/{empid}	200 OK 404 Not Found
Скорочення днів відпустки	POST /api/v1/vacations/{empid}/reduction	200 OK

Резервування Vacation Service



Компоненти



Payment Interface

Що він робить:

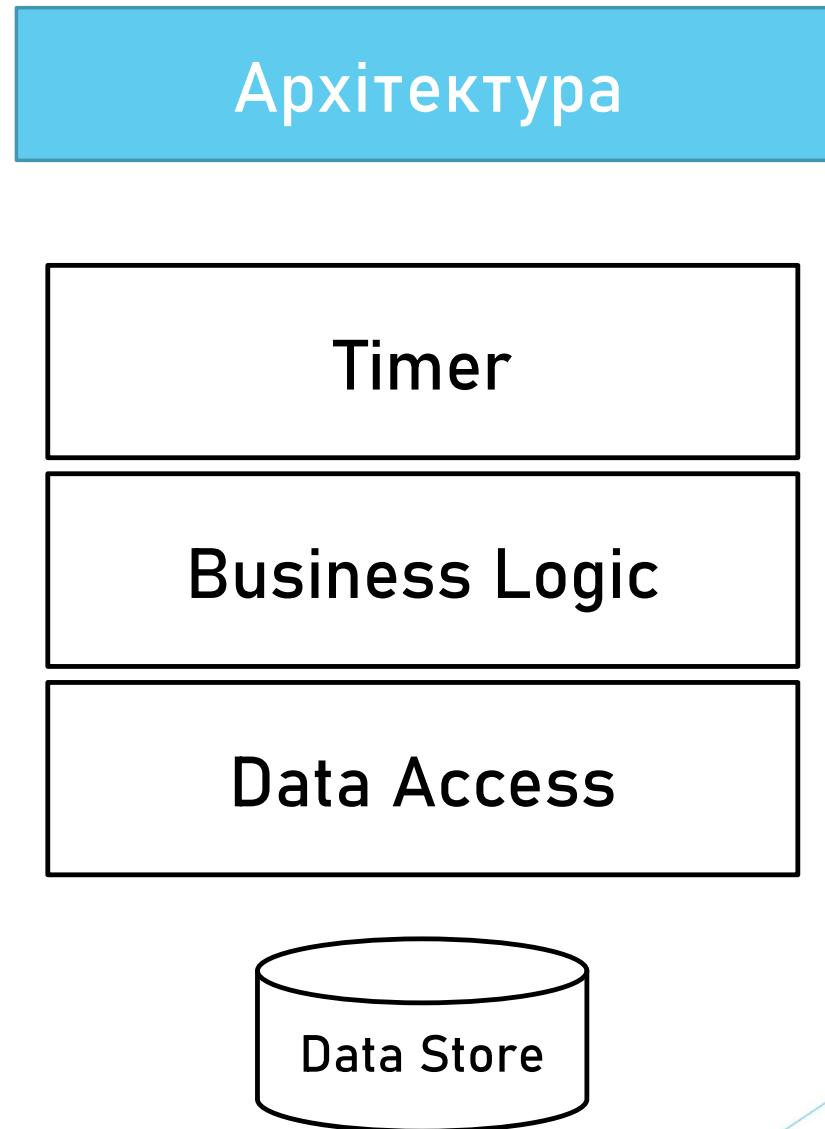
- Раз на місяць запитує базу даних щодо зарплати
- Передає платіжні дані в зовнішню платіжну систему

Тип програми

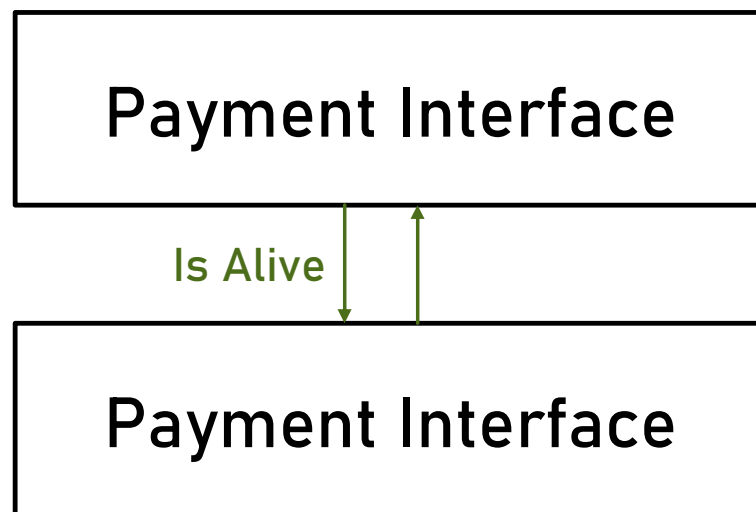
- Веб-додаток і веб-API ✗
- Мобільний додаток ✗
- Консоль ✗
- Сервіс ✓
- Настільний додаток ✗

Стек технологій

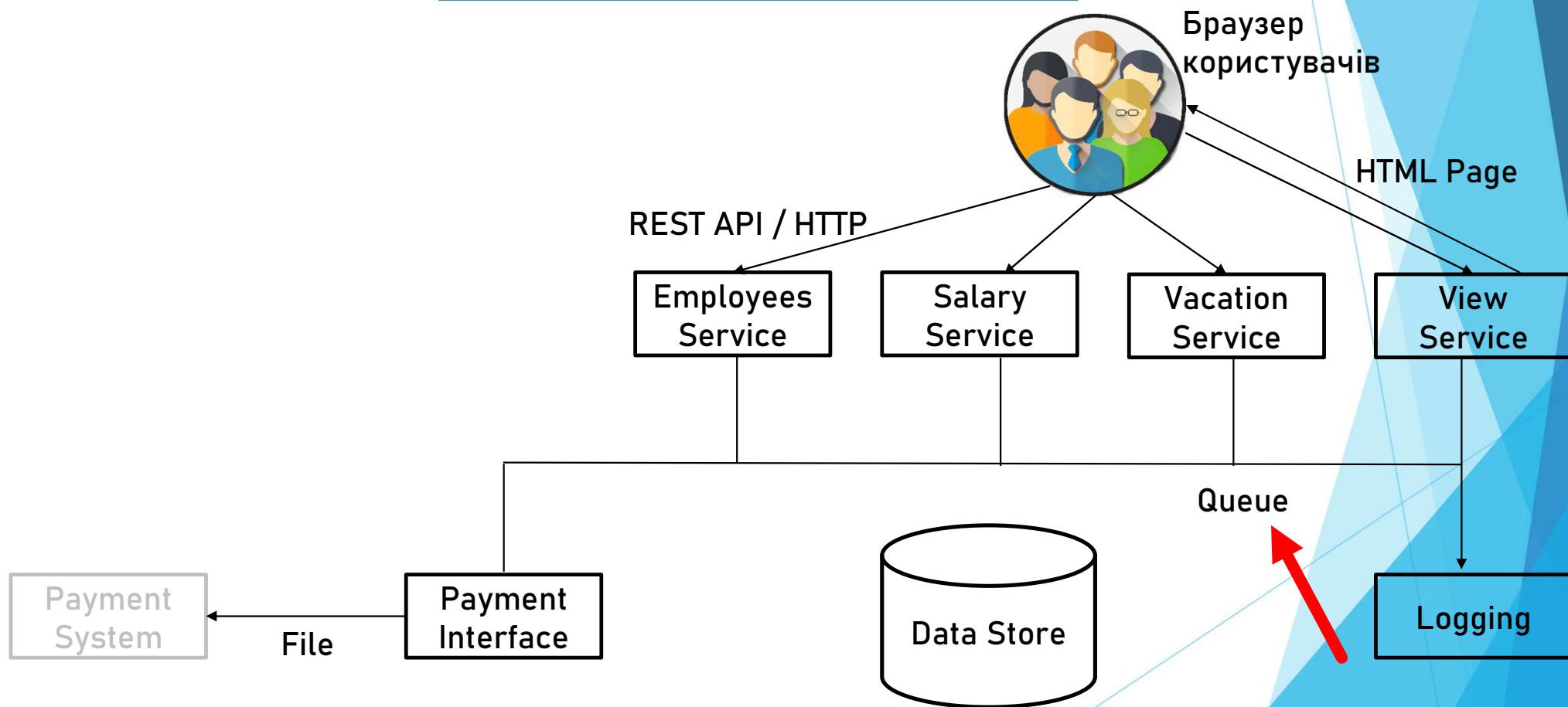




Резервування Payment Interface



Обмін повідомленнями



Технологічний стек – черга

Альтернативи черзі:

~~Самостійна розробка~~



RabbitMQ



Kafka

Alternative	Description	Pros
Rabbit MQ	General purpose message-broker engine	Easy to setup Easy to use
Apache Kafka	Stream processing platform	Perfect for data intensive scenarios

Технологічний стек – черга

Альтернативи черзі:

Самостійна розробка



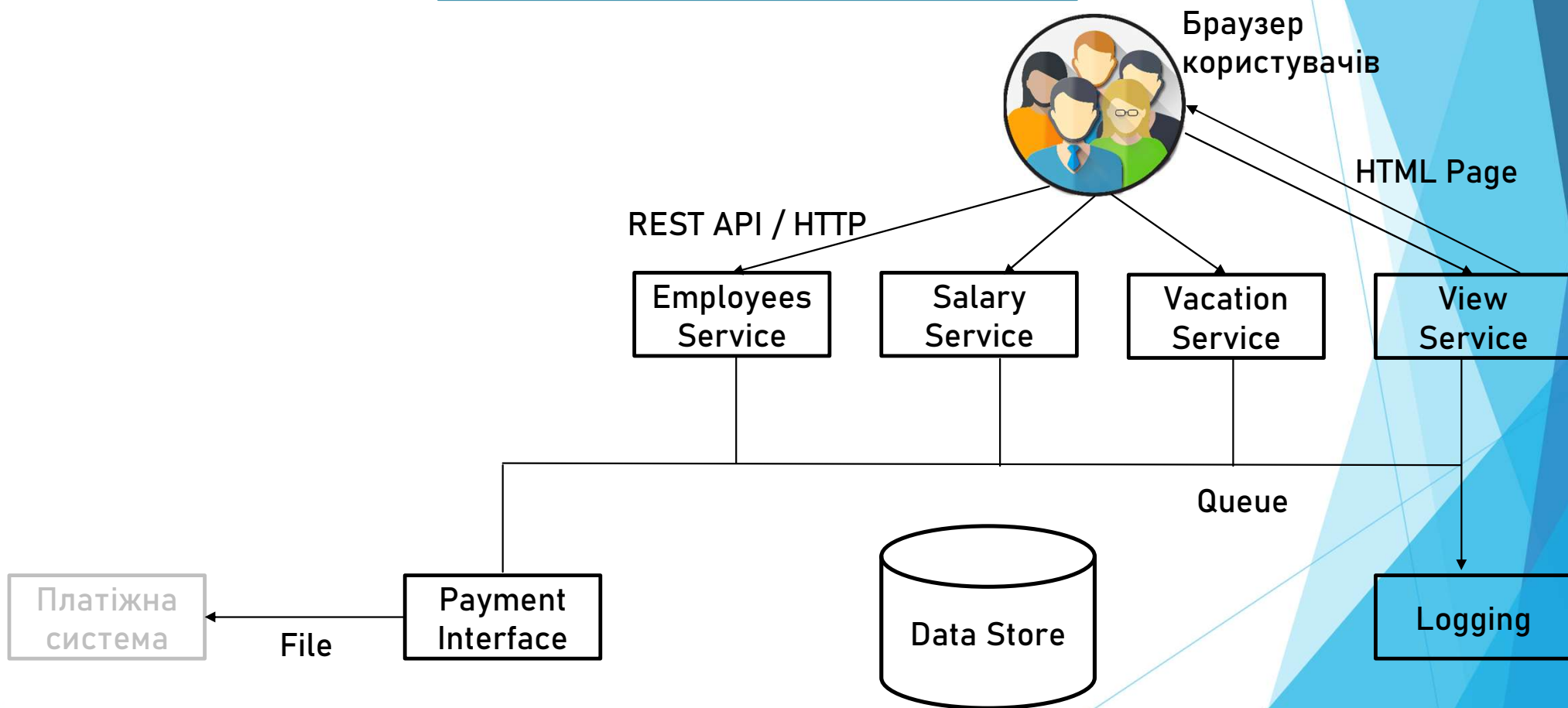
RabbitMQ



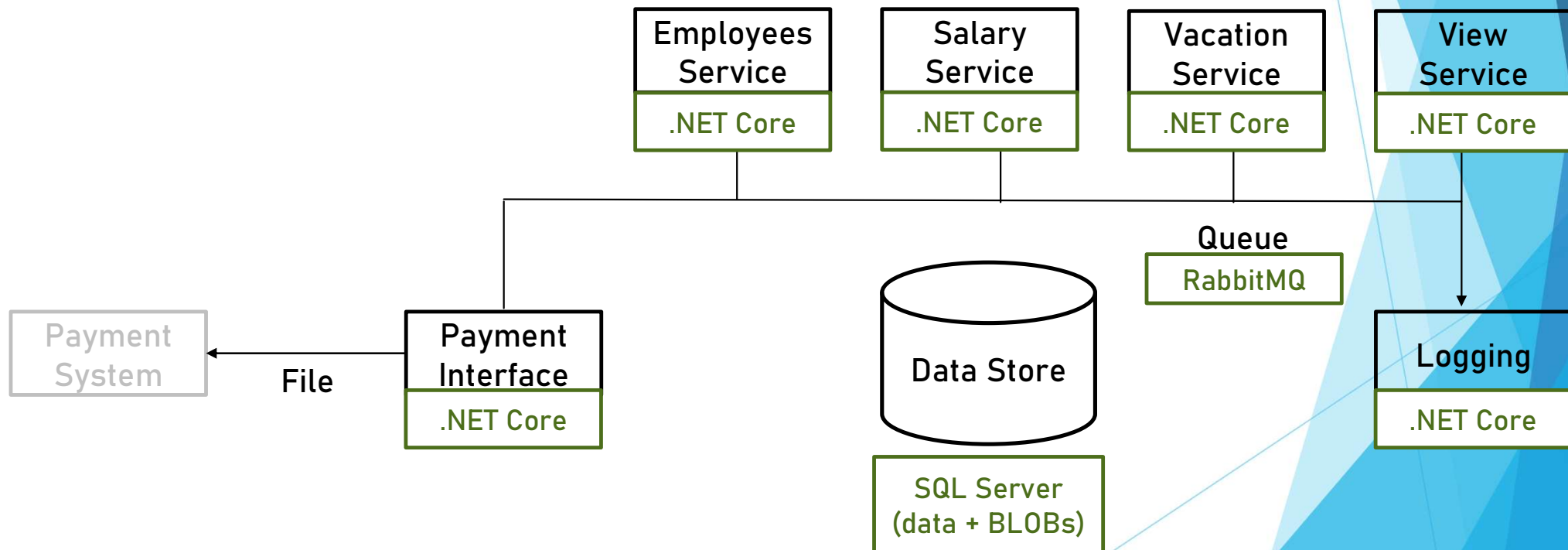
Kafka

- Потік даних не задіяний
- Простий у використанні

Логічна схема



Технічна схема



Фізична схема

