Module 2

Week 1 | Day 1 | **Domain Driven Design**

Where we are: **Monolith**

Django Project

Project Application

Templates

Models

Views

URLS

HTML

localhost:3000/project/1

Browser

127.0.0.1:8000/projects/1/

- My tasks
- My projects
- Logout

This is a new project

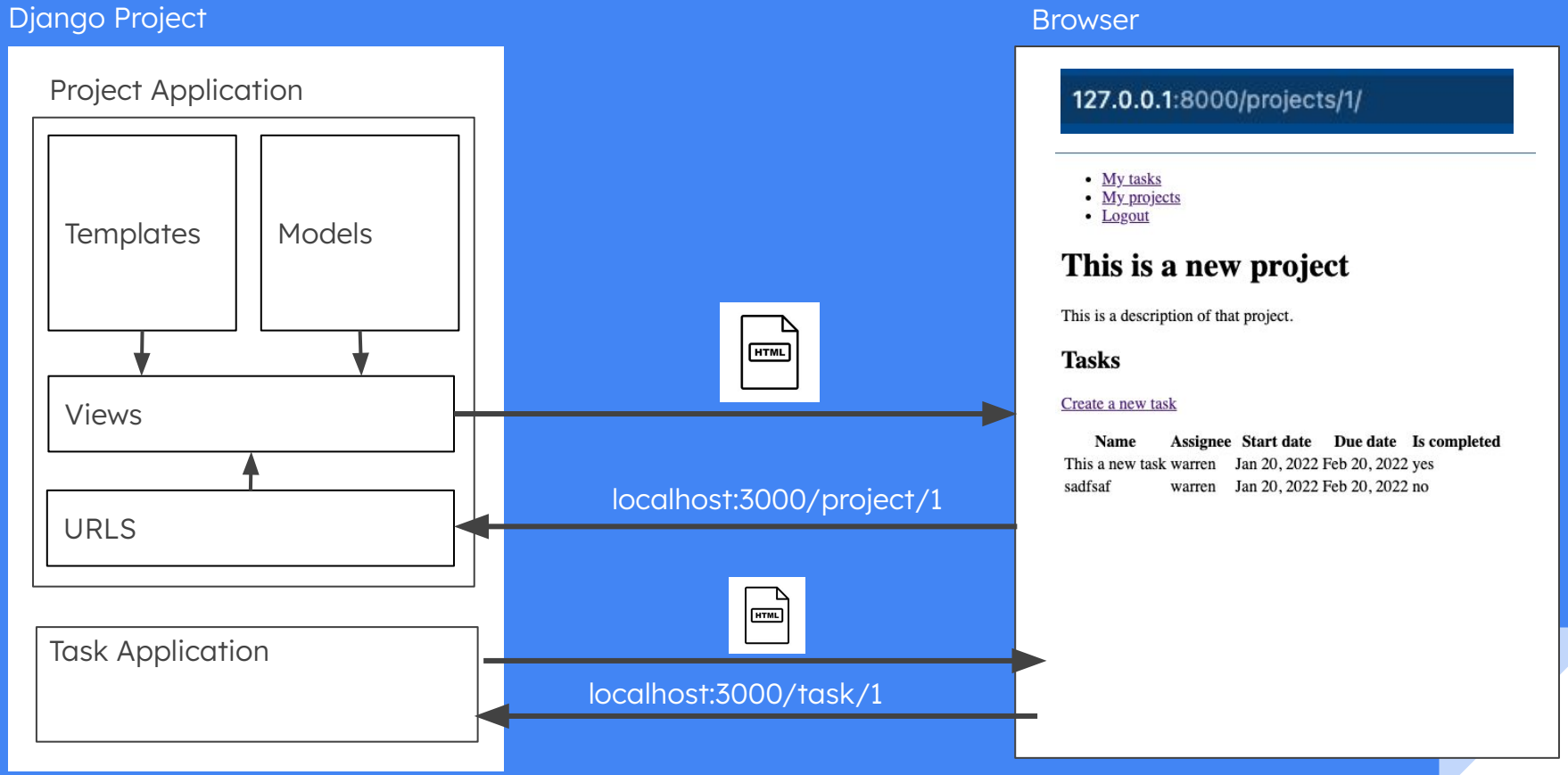
This is a description of that project.

Tasks

Create a new task

Name	Assignee	Start date	Due date	Is completed
This a new task	warren	Jan 20, 2022	Feb 20, 2022	yes
sadfsaf	warren	Jan 20, 2022	Feb 20, 2022	no

Where we are: **Monolith**



Where we are: **Monolith**

Pros:

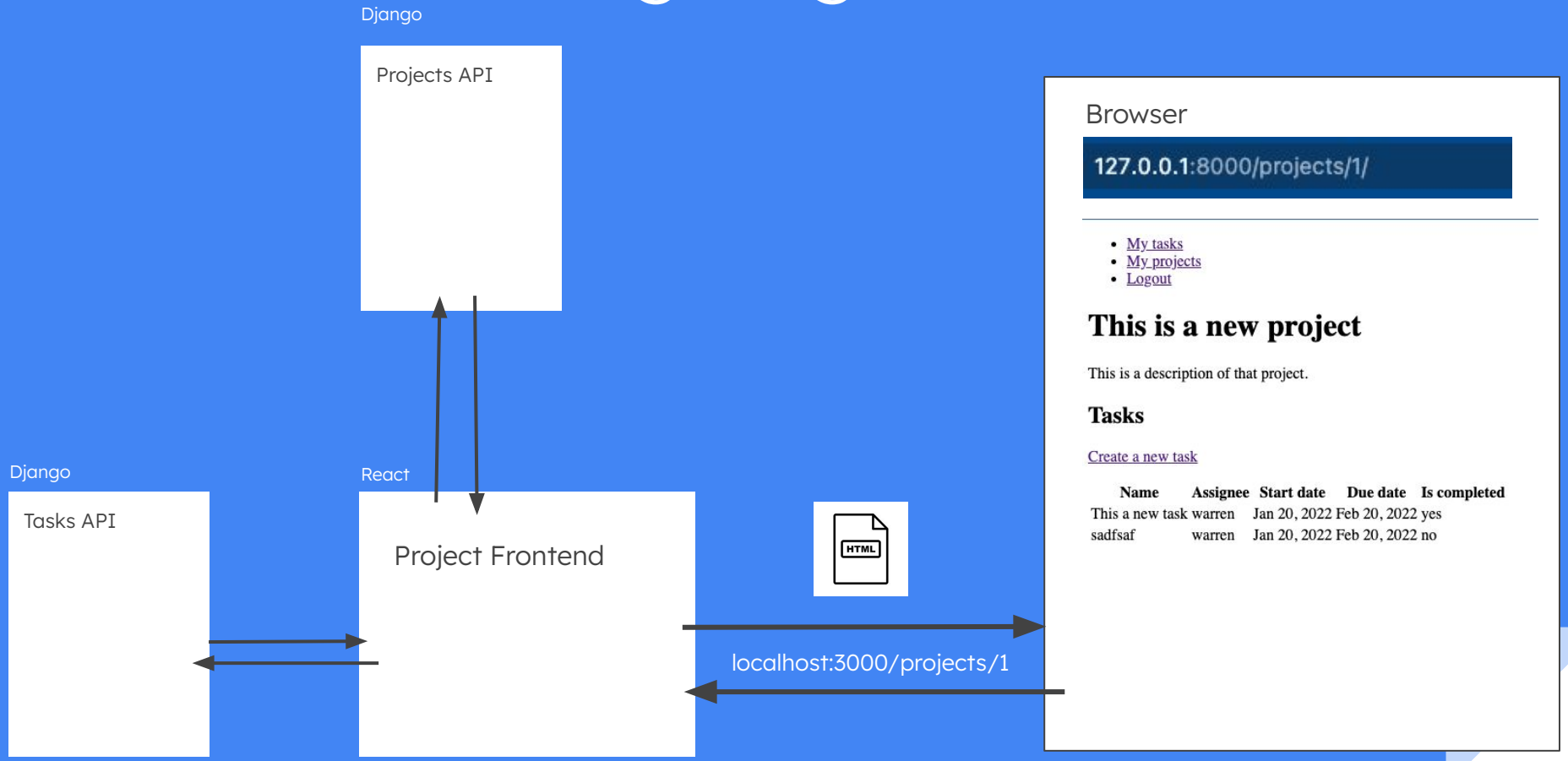
Easy to get up and running
Everything is in one language
Clear process for each feature

Cons:

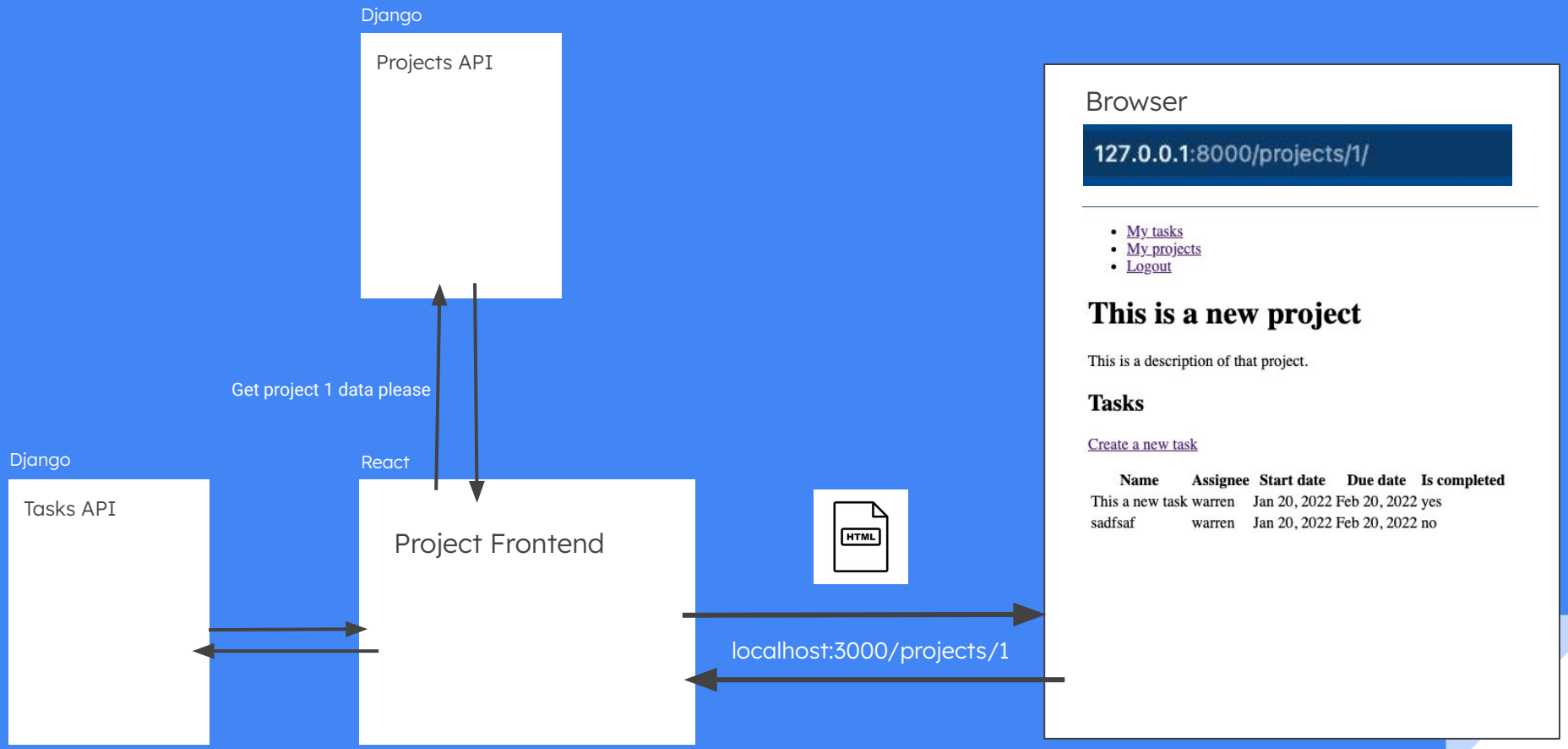
Doesn’t Scale!!!
Language / Platform lockin
Hard to add big new features
Hard for teams to work on it

Big projects
require
max flexibility.

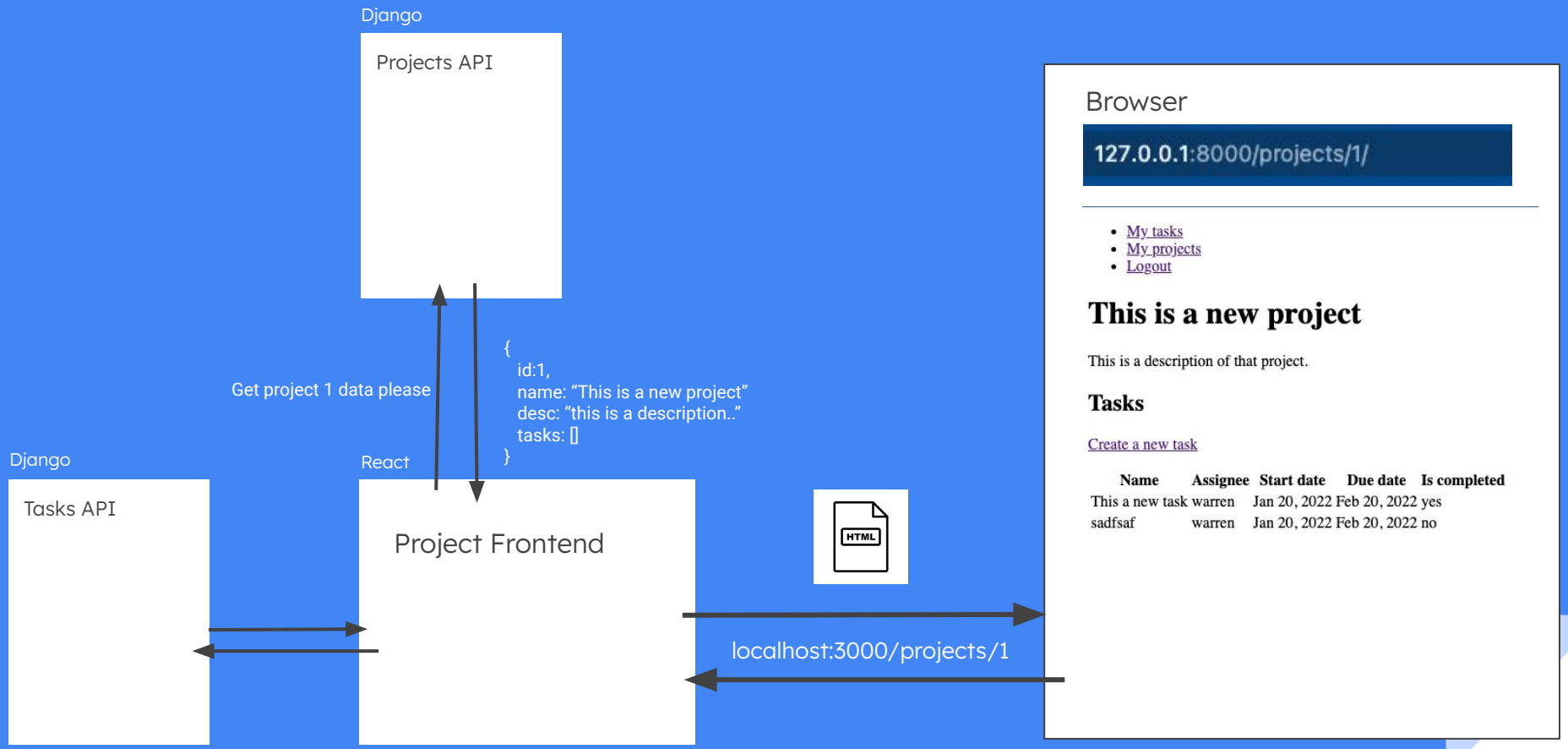
Where we are going: **Microservices**



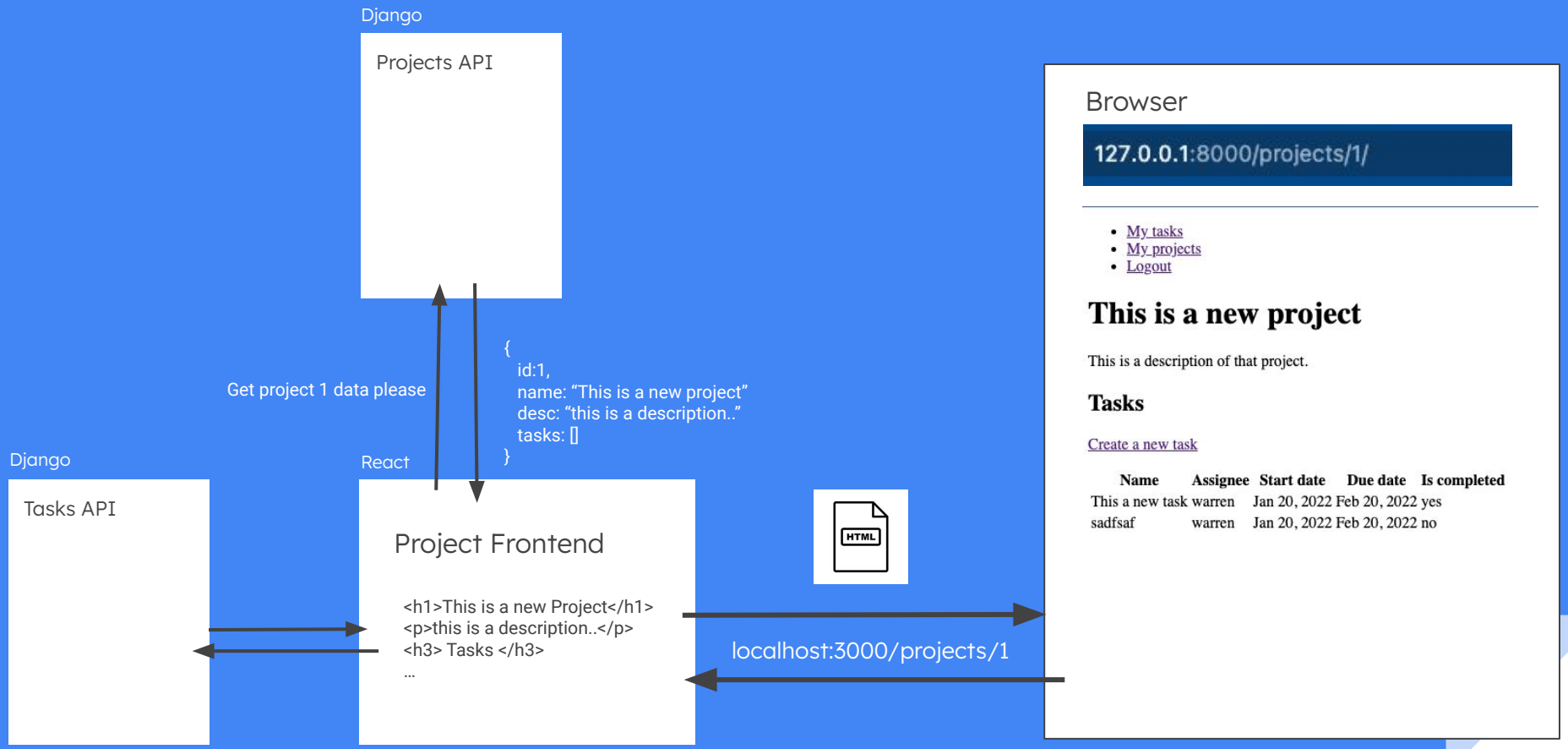
Where we are going: **Microservices**



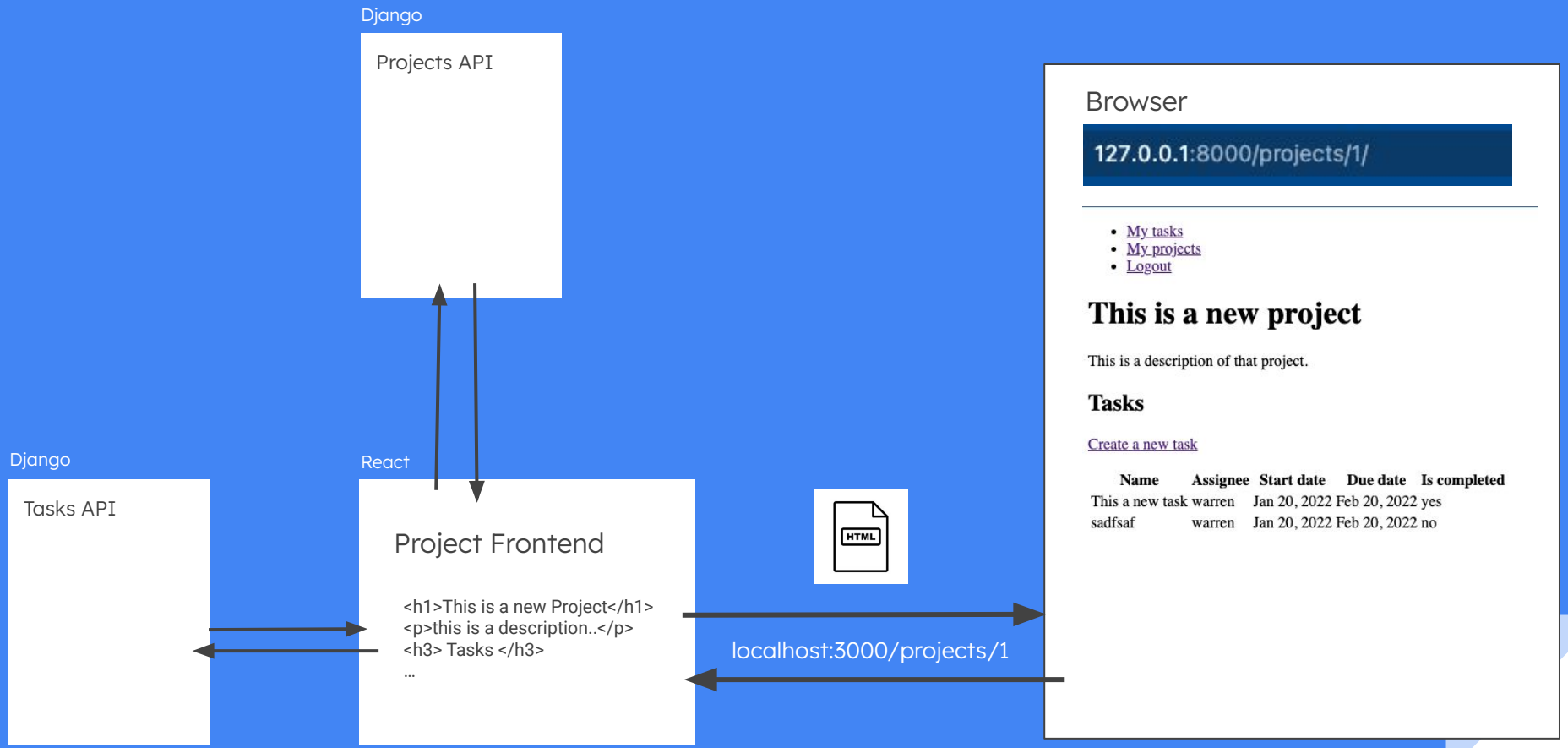
Where we are going: **Microservices**



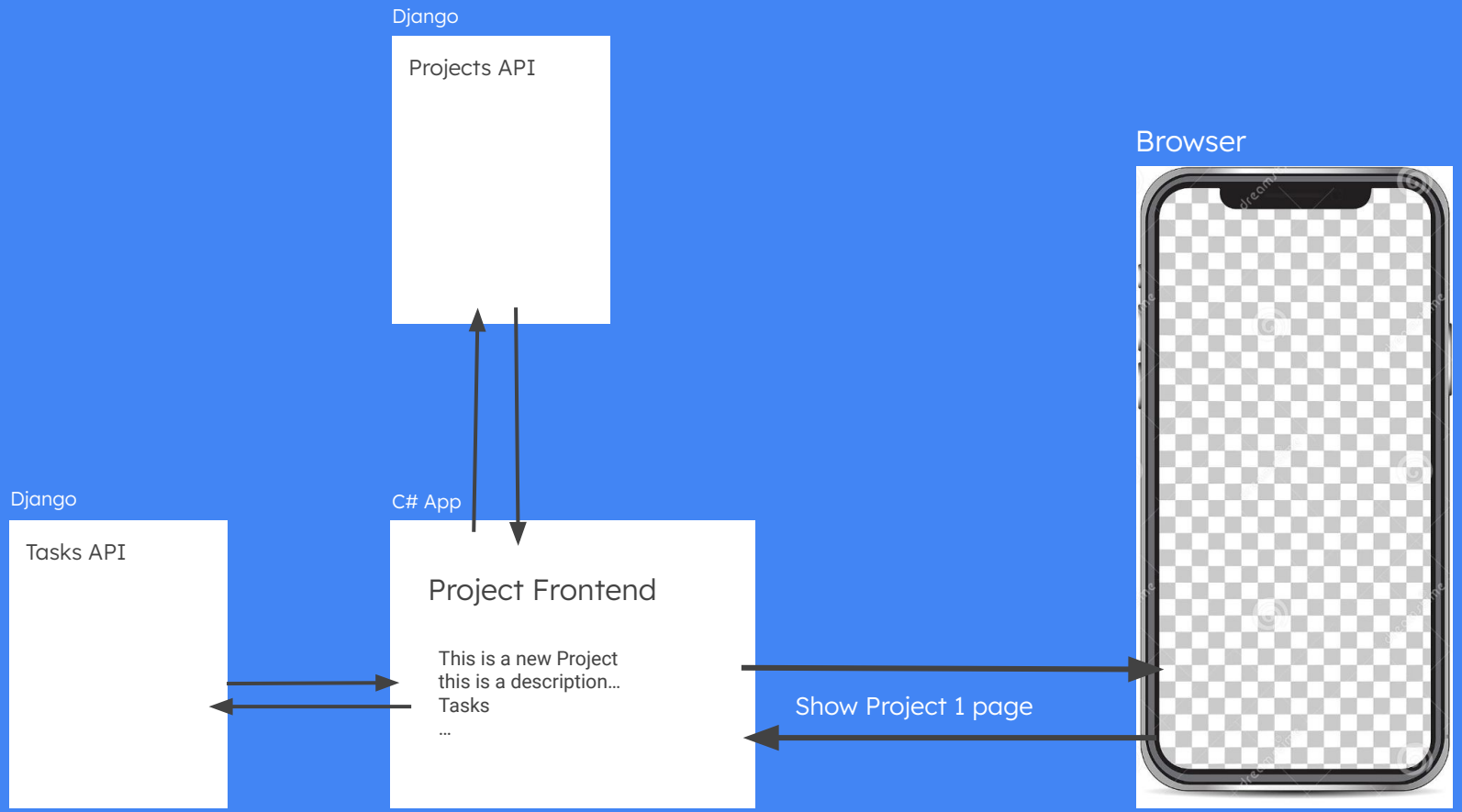
Where we are going: **Microservices**



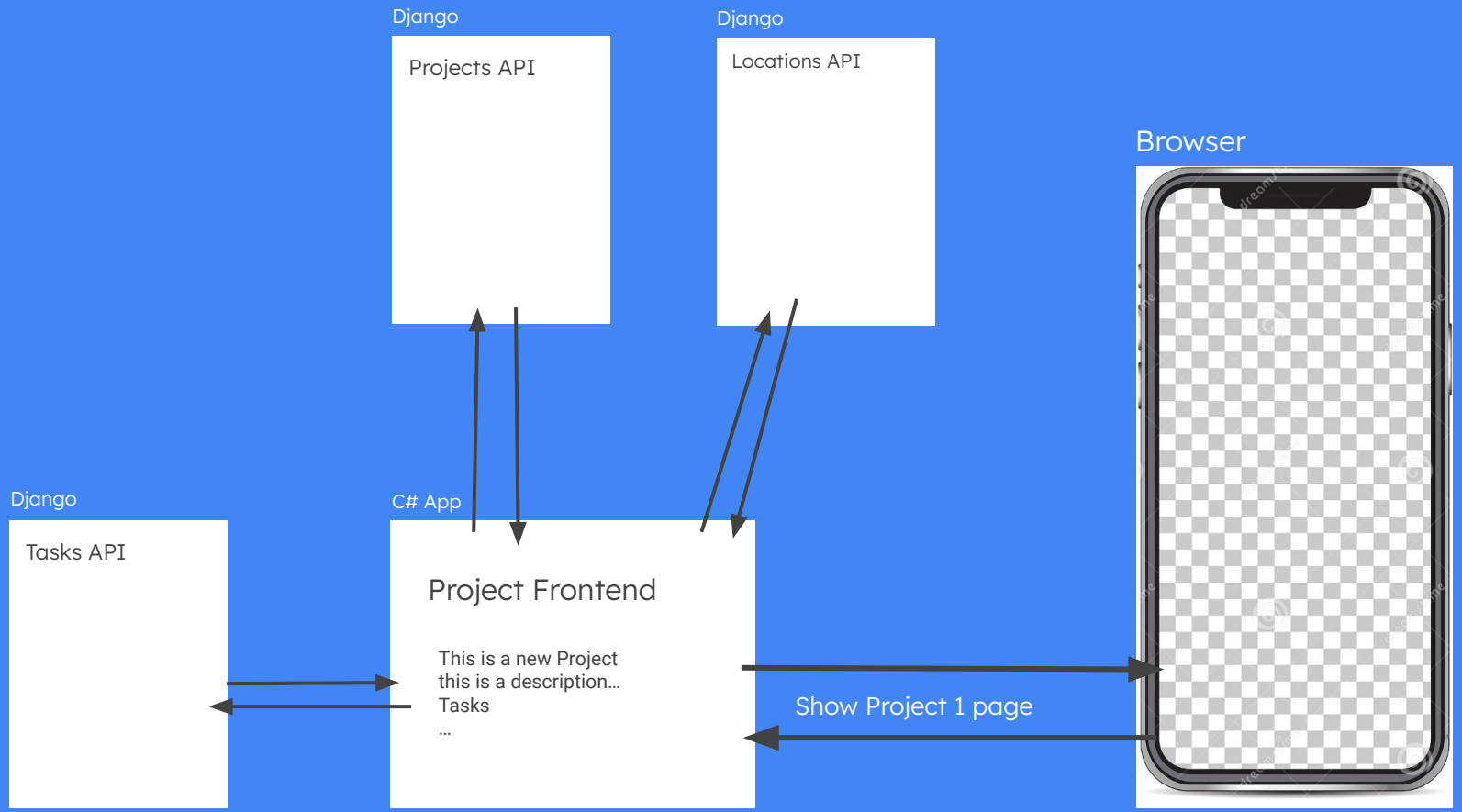
Microservices Are Flexible!



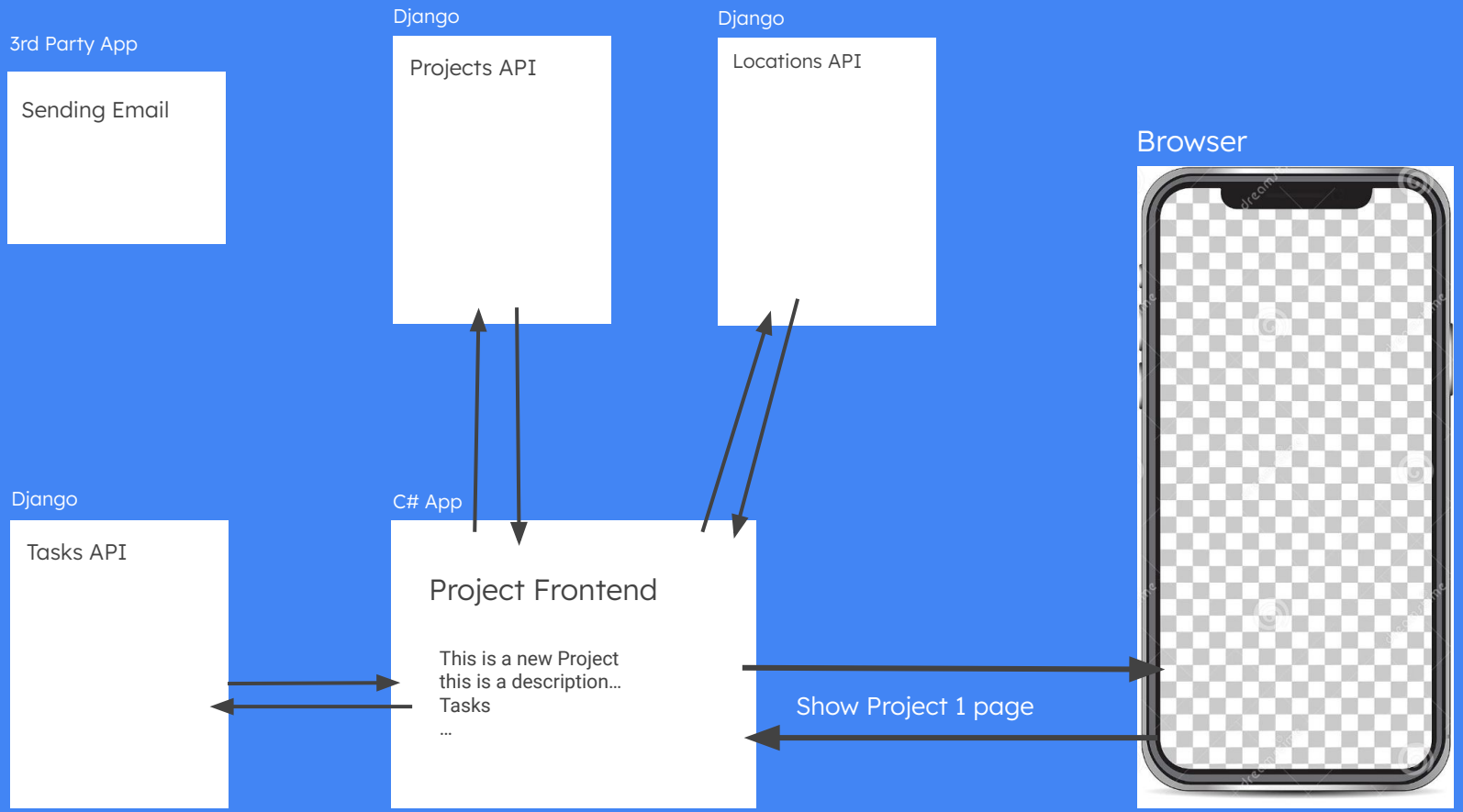
Microservices Are Flexible!



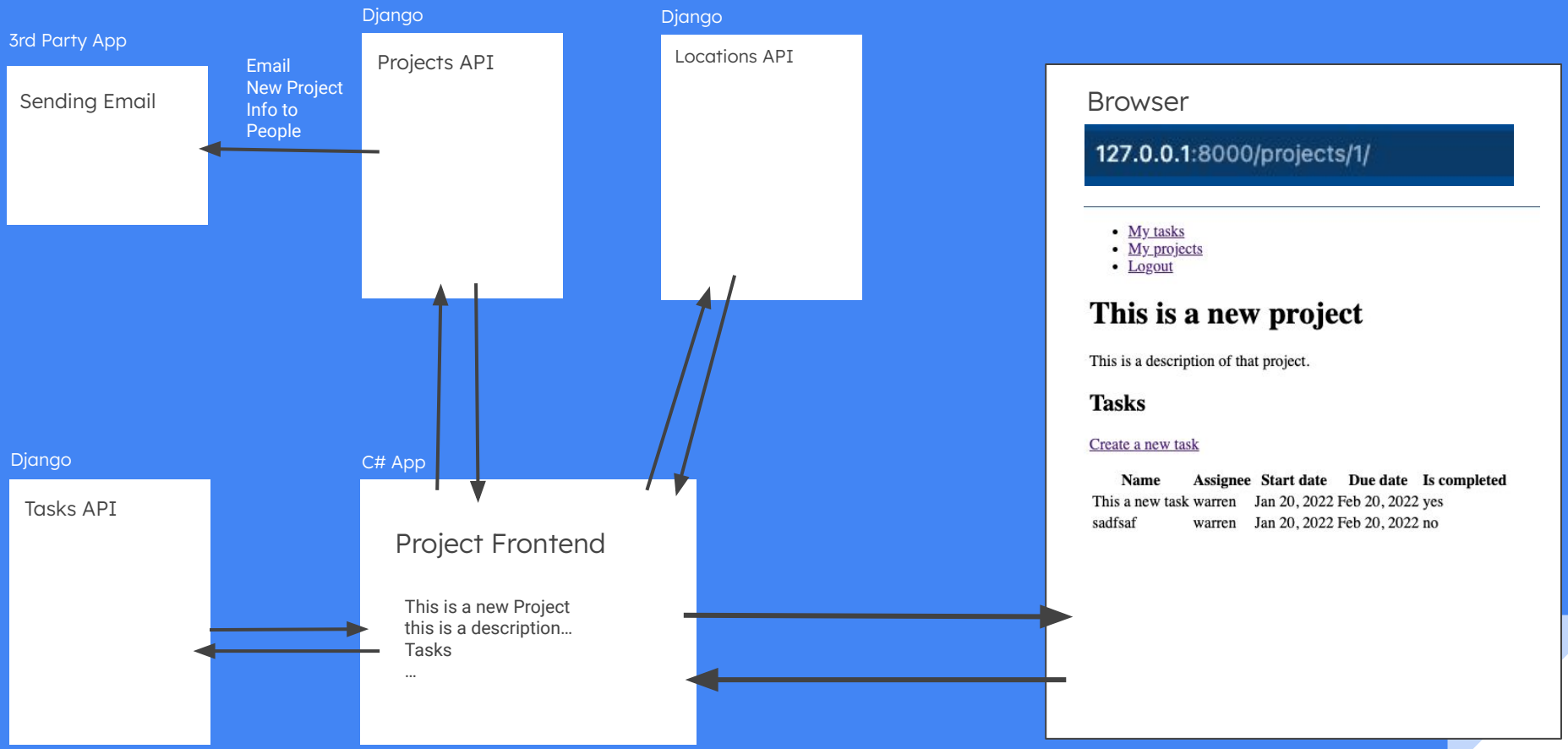
Microservices Are Flexible!



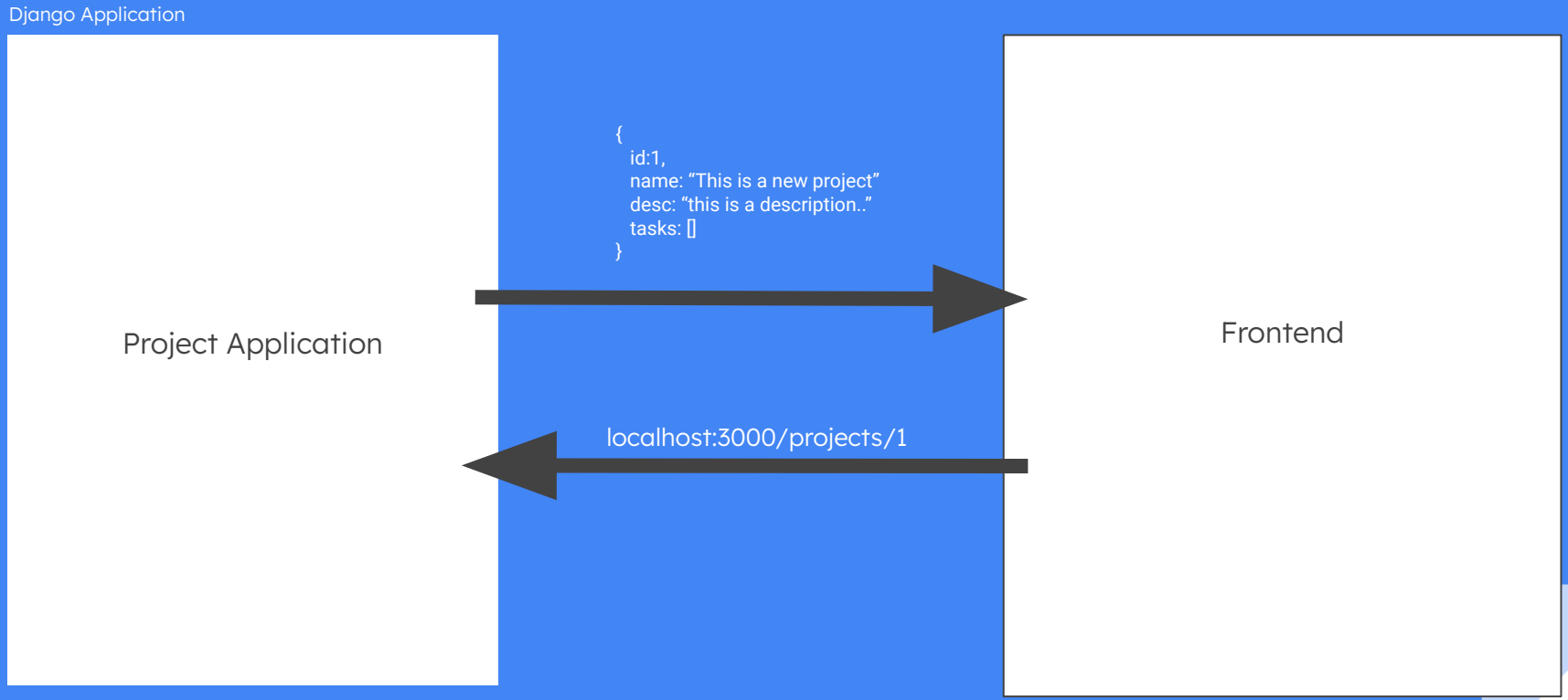
Microservices Are Flexible!



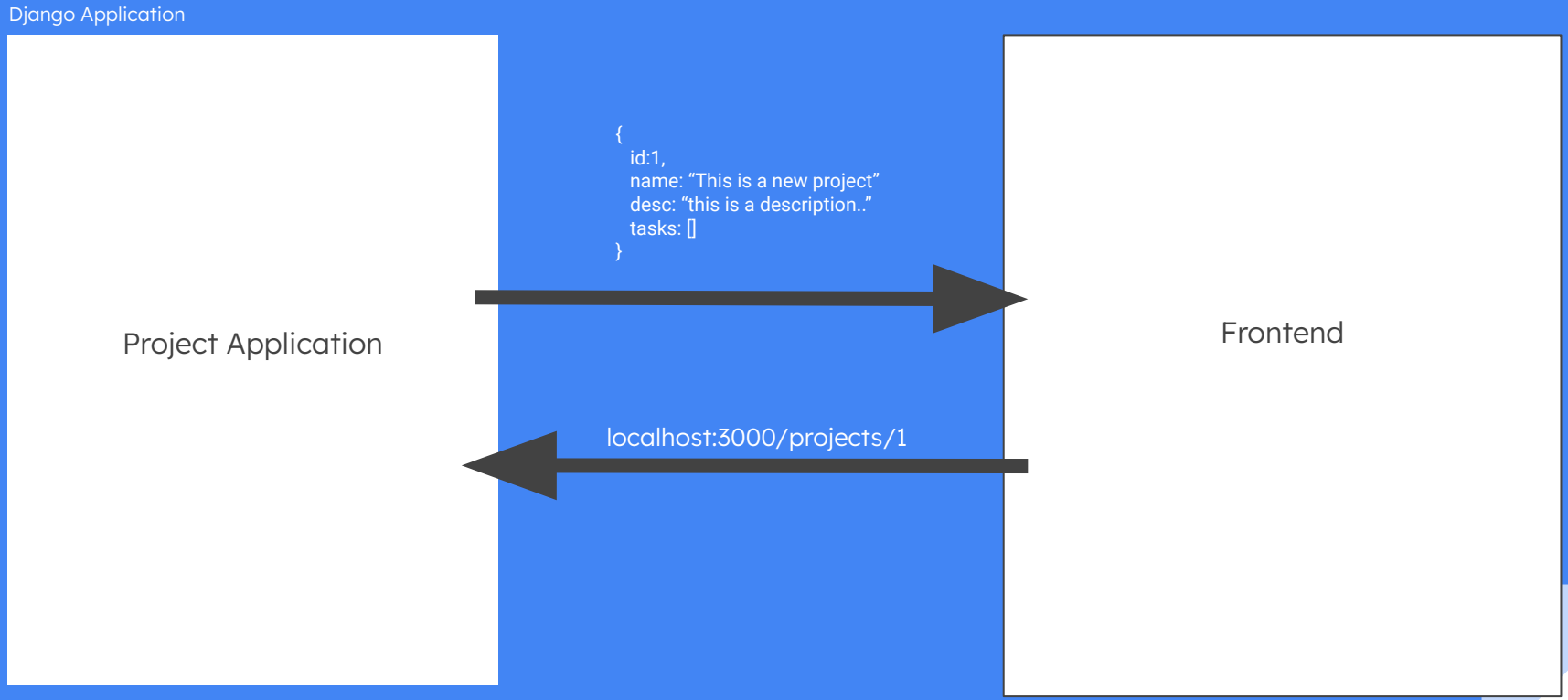
Microservices Are Flexible!



This week: Application Programming Interfaces



This week: APIS



This week: APIS

Django Project

Project Application

```
{
  id:1,
  name: “This is a new project”
  desc: “this is a description..”
  tasks: []
}
```

localhost:3000/projects/1

Goals of the Week

	Conceptual:	Technical:
Overall:	Domain Driven Design	Building an API Backend
Today:	Entities & VO - What are our Models?	Models and JSON
Tuesday:	Aggregates - What are our Collections?	Building a JSON Library
Wednesday:	Bounded Context - What are our Systems?	Making API Interfaces
Thursday:	Anti-Corruption - How do we structure data?	Working with 3rd Party
Friday:	Containerizing - Where do system runs	Working with Docker

But first:

Data Modeling

Ubiquitous Language:
the common names

Ubiquitous Language:
the common names for software design pieces

Ubiquitous Language:
the common names for application processes

Ubiquitous Language:

the common names for your databases, variables and methods

Ubiquitous Language:

Common names for:

- **Data:** Model
- **Data Associations:** Relationships
- **Processes :** Ordering / Flow

Let’s build a hotel...

Let’s build a hotel...

Entities  Value Objects

Entities  Value Objects

- Indicates Persistent Identity
- Has a lifecycle
- Values it contains don’t change what it is - mutable

Examples:

Entities

Value Objects

- Indicates Persistent Identity
- Has a lifecycle
- Values it contains don’t change what it is - mutable

Examples:

- Automobile

Entities  Value Objects

- Indicates Persistent Identity
- Has a lifecycle
- Values it contains don’t change what it is - mutable

Examples:

- Automobile
- Customer

Entities  Value Objects

- Indicates Persistent Identity
- Has a lifecycle
- Values it contains don’t change what it is - mutable

Examples:

- Automobile
- Customer
- Distributor

Entities  Value Objects

- Indicates Persistent Identity
- Has a lifecycle
- Values it contains don’t change what it is - mutable

Examples:

- Automobile
- Customer
- Distributor
- Invoice

Entities

- Indicates Persistent Identity
- Has a lifecycle
- Values it contains don't change what it is - mutable

Examples:

- Automobile
- Customer
- Distributor
- Invoice

Value Objects

- Indicates Static Value
- Probably doesn't have an id
- It IS it's value
- It is used to define entity values

Examples:

Entities

- Indicates Persistent Identity
- Has a lifecycle
- Values it contains don't change what it is

Examples:

- Automobile
- Customer
- Distributor
- Invoice

Value Objects

- Indicates Static Value
- Probably doesn't have an id
- It IS it's value
- It is used to define entity values

Examples:

- Paint Job Color

Entities

- Indicates Persistent Identity
- Has a lifecycle
- Values it contains don't change what it is

Examples:

- Automobile
- Customer
- Distributor
- Invoice

Value Objects

- Indicates Static Value
- Probably doesn't have an id
- It IS it's value
- It is used to define entity values

Examples:

- Paint Job Color
- Currency

Entities

- Indicates Persistent Identity
- Has a lifecycle
- Values it contains don't change what it is

Examples:

- Automobile
- Customer
- Distributor
- Invoice

Value Objects

- Indicates Static Value
- Probably doesn't have an id
- It IS it's value
- It is used to define entity values

Examples:

- Paint Job Color
- Currency
- Vehicle Type

Entities

- Indicates Persistent Identity
- Has a lifecycle
- Values it contains don't change what it is

Examples:

- Automobile
- Customer
- Distributor
- Invoice

Value Objects

- Indicates Static Value
- Probably doesn't have an id
- It IS it's value
- It is used to define entity values

Examples:

- Paint Job Color
- Currency
- Vehicle Type
- Service Type

Let’s talk about JSON!!

What is JSON?

What is JSON?

Java**S**cript **O**bject **N**otation

Myopic name

```
for language in programming_languages_worth_their_salt:
    print(f'{language} uses JSON!')
```

- Python
- Javascript
- Go, Ruby, Scratch, etc.

Cool, but what IS JSON?

Old busted....

```xml
<!-- XML, gross -->
<birds_are_real type="boolean">false</birds_are_real>
<meaning_of_life type="integer">42</meaning_of_life>
```

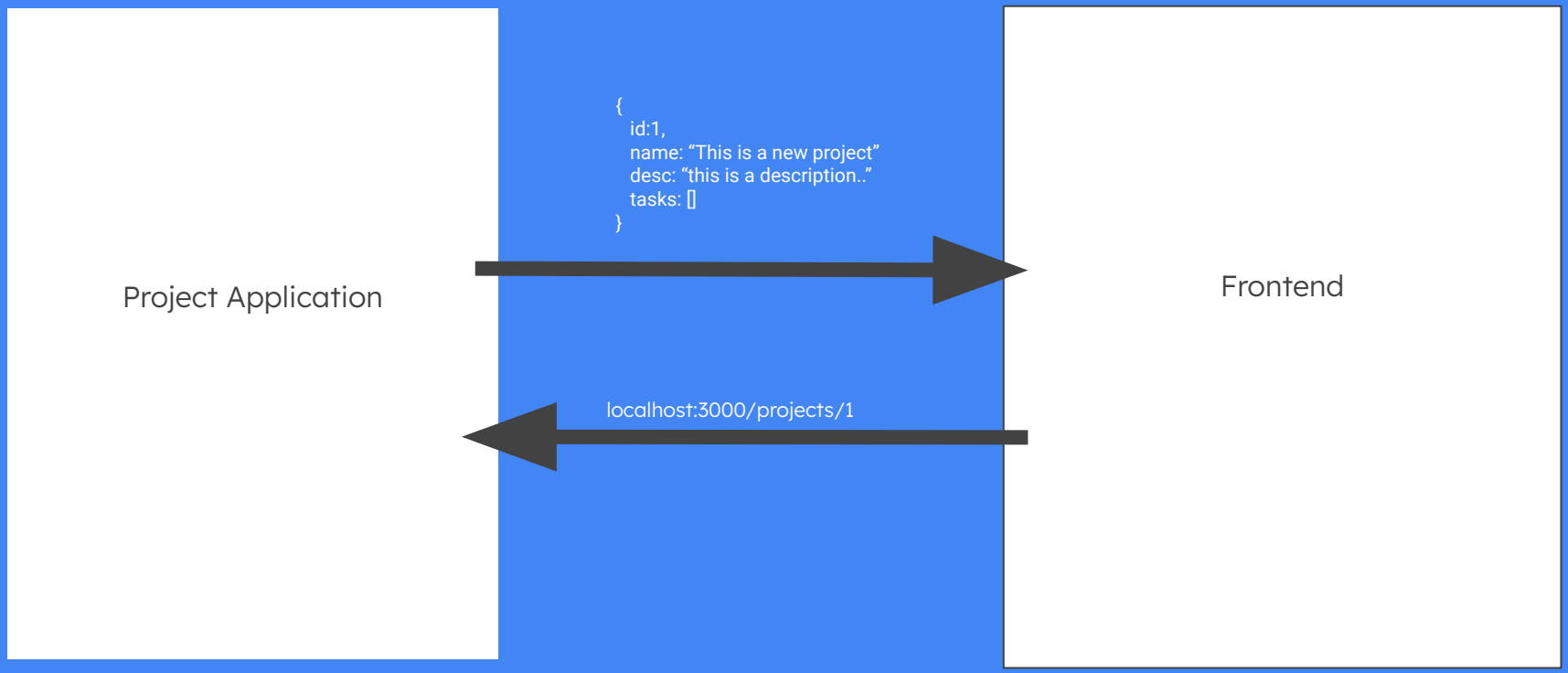
New Hotness!

```json
{
  "birds_are_real": false,
  "meaning_of_life": 42
}
```

What are we using it for?

What are we using it for?

Django Application



What are we using it for?

Django Application

Project Application

```
{
  id:1,
  name: “This is a new project”
  desc: “this is a description..”
  tasks: []
}
```

localhost:3000/projects/1