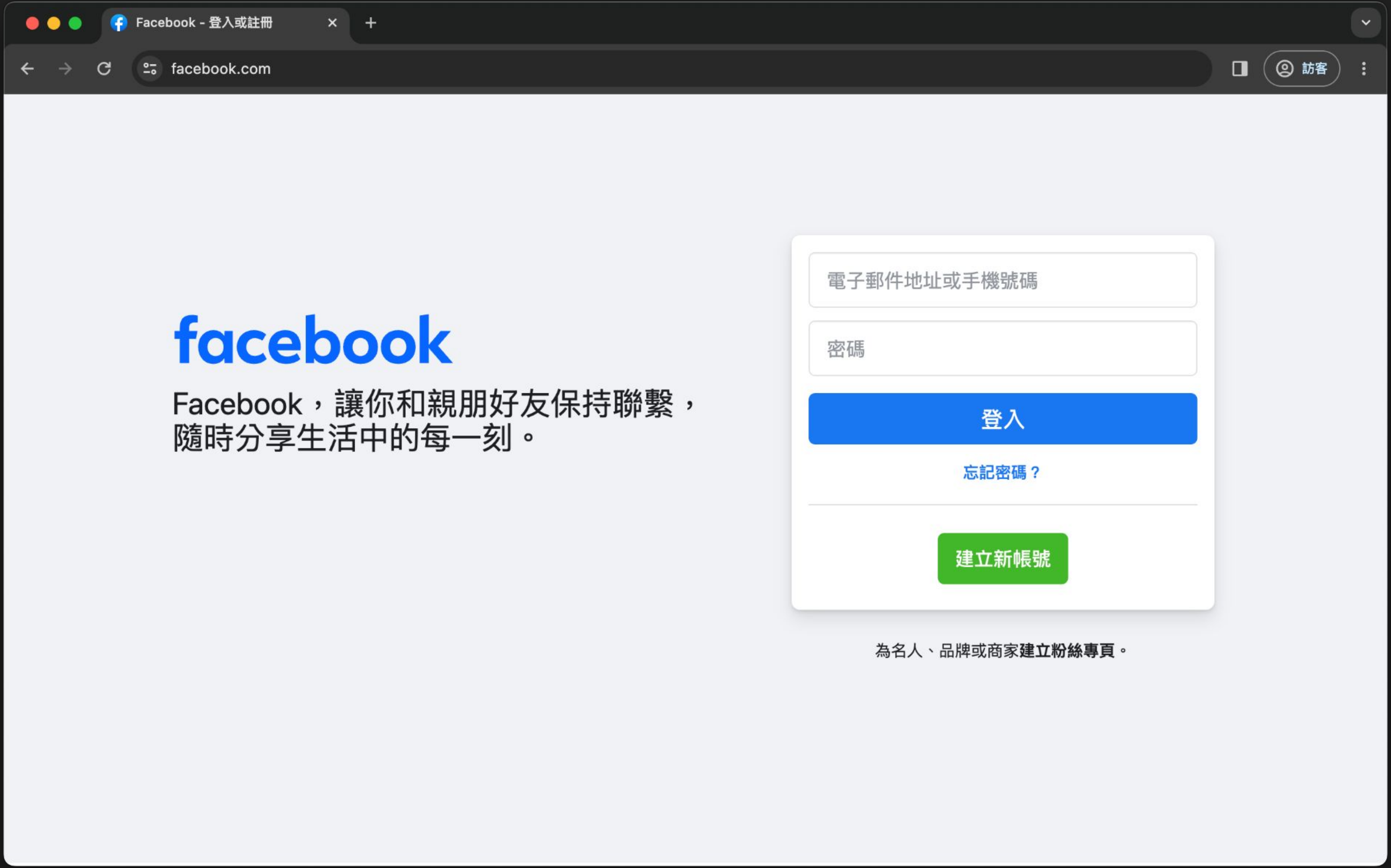
Backend

陳肇廷

https://github.com/GNTIOAHC/gdsc-backend-tutorial

Frontend

what we see



facebook

Facebook，讓你和親朋好友保持聯繫，
隨時分享生活中的每一刻。

電子郵件地址或手機號碼

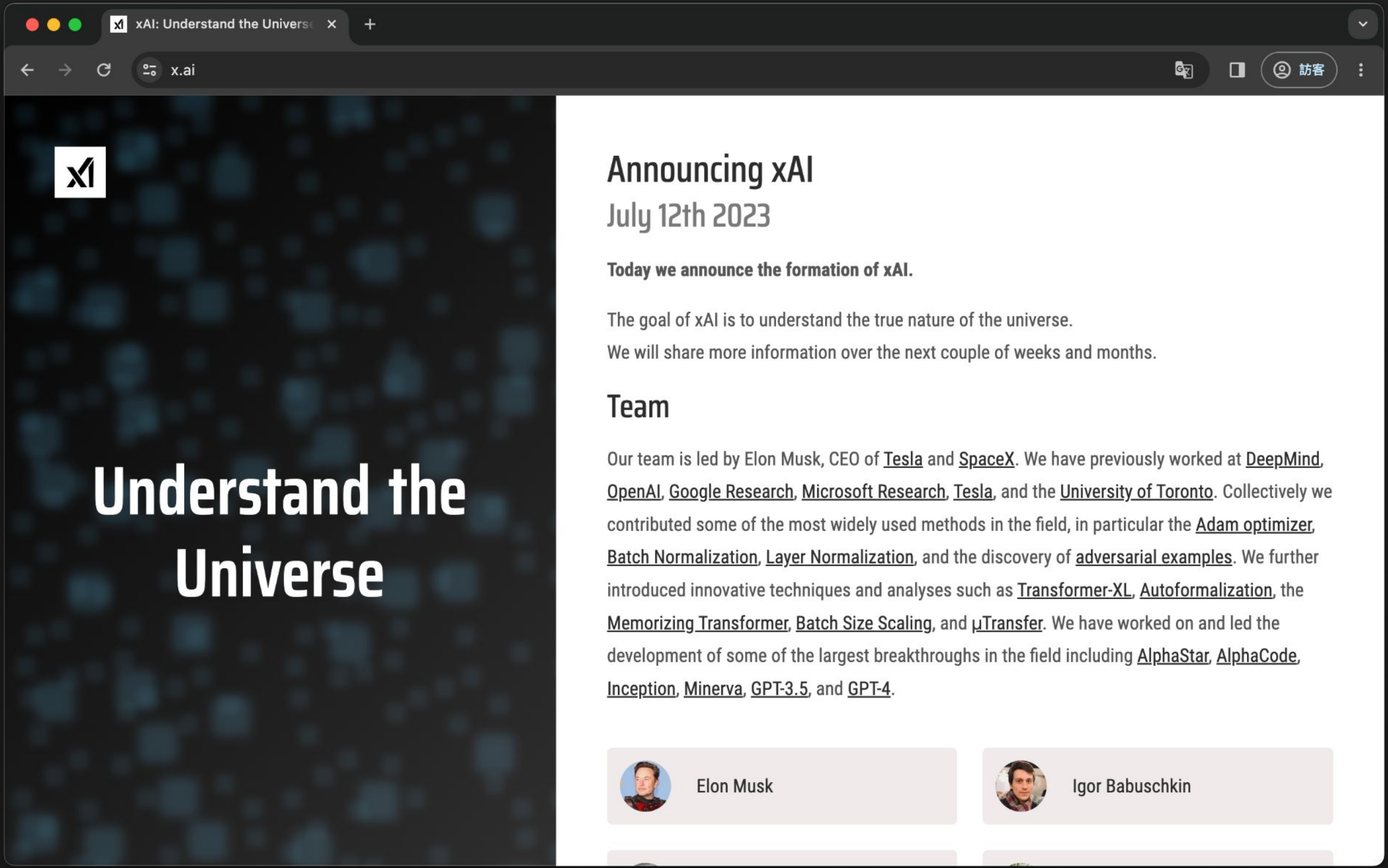
密碼

登入

忘記密碼？

建立新帳號

為名人、品牌或商家**建立粉絲專頁**。



Announcing xAI

July 12th 2023

Today we announce the formation of xAI.

The goal of xAI is to understand the true nature of the universe.
We will share more information over the next couple of weeks and months.

Team

Our team is led by Elon Musk, CEO of Tesla and SpaceX. We have previously worked at DeepMind, OpenAI, Google Research, Microsoft Research, Tesla, and the University of Toronto. Collectively we contributed some of the most widely used methods in the field, in particular the Adam optimizer, Batch Normalization, Layer Normalization, and the discovery of adversarial examples. We further introduced innovative techniques and analyses such as Transformer-XL, Autoformalization, the Memorizing Transformer, Batch Size Scaling, and μTransfer. We have worked on and led the development of some of the largest breakthroughs in the field including AlphaStar, AlphaCode, Inception, Minerva, GPT-3.5, and GPT-4.

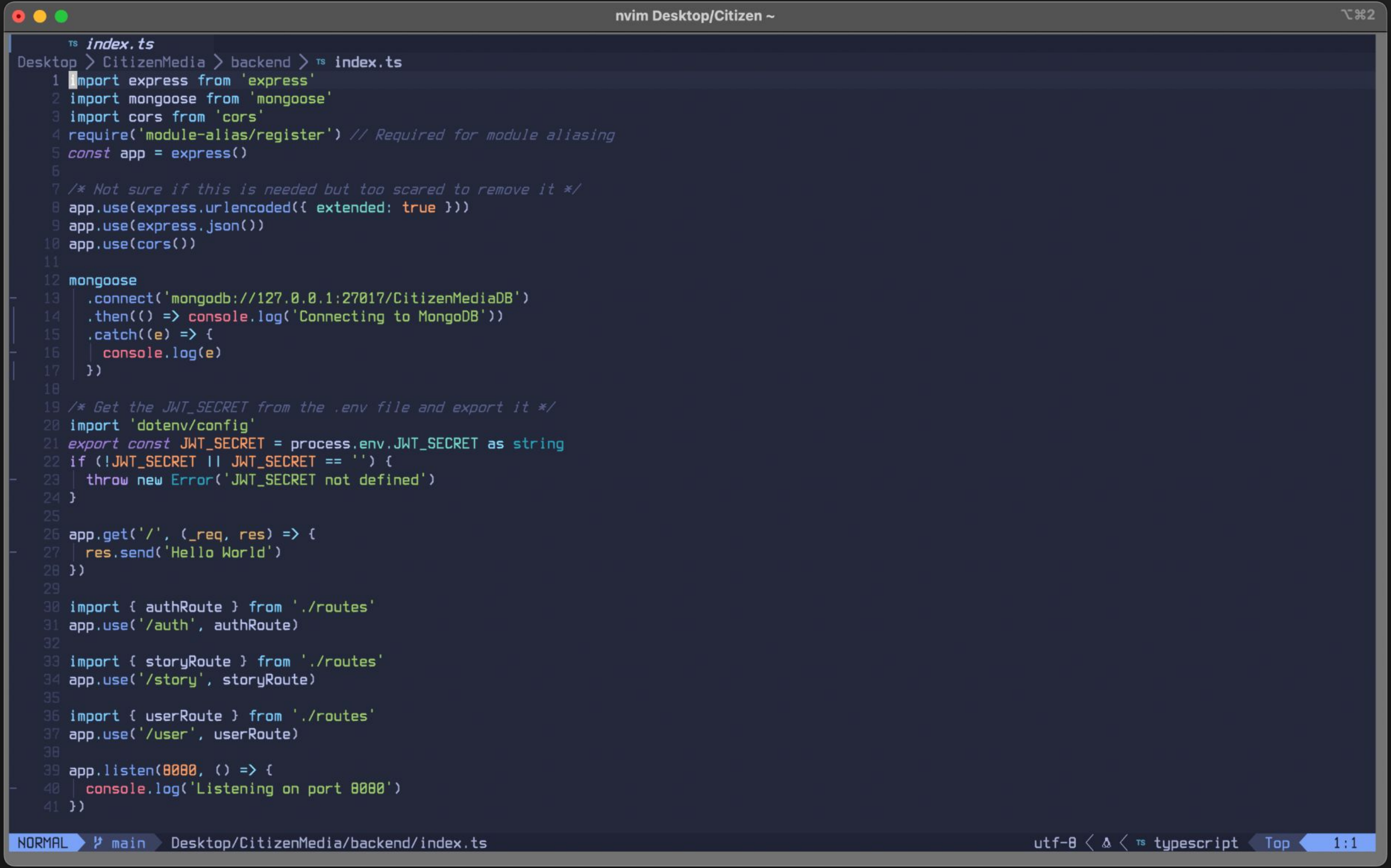
Elon Musk

Igor Babuschkin

Backend

what provide the service

just code ?



```typescript
import express from 'express'
import mongoose from 'mongoose'
import cors from 'cors'
require('module-alias/register') // Required for module aliasing
const app = express()

/* Not sure if this is needed but too scared to remove it */
app.use(express.urlencoded({ extended: true }))
app.use(express.json())
app.use(cors())

mongoose
  .connect('mongodb://127.0.0.1:27017/CitizenMediaDB')
  .then(() => console.log('Connecting to MongoDB'))
  .catch((e) => {
    console.log(e)
  })

/* Get the JWT_SECRET from the .env file and export it */
import 'dotenv/config'
export const JWT_SECRET = process.env.JWT_SECRET as string
if (!JWT_SECRET || JWT_SECRET == '') {
  throw new Error('JWT_SECRET not defined')
}

app.get('/', (_req, res) => {
  res.send('Hello World')
})

import { authRoute } from './routes'
app.use('/auth', authRoute)

import { storyRoute } from './routes'
app.use('/story', storyRoute)

import { userRoute } from './routes'
app.use('/user', userRoute)

app.listen(8080, () => {
  console.log('Listening on port 8080')
})
```

Backend v.s. Frontend

client-server model

client: frontend

server: backend

client-server model
客戶：frontend
伺服器：backend

client-server model

客戶: frontend

廚師: backend

client-server model

客戶: frontend
服務生: api
廚師: backend

client-server model

客戶: frontend

服務生: api

點單: json

廚師: backend

client-server model

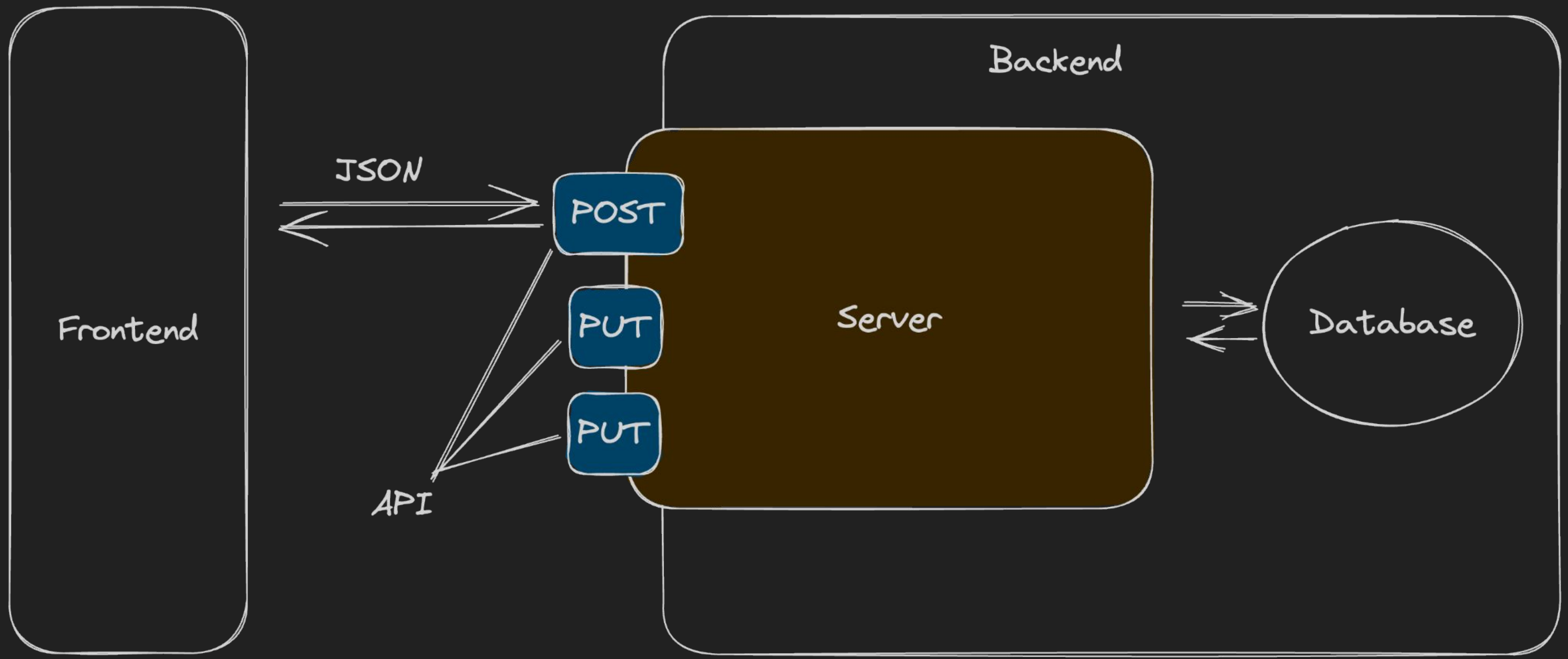
客戶: frontend

服務生: api

點單: json

廚師: backend

冰箱: Database


Frontend
JSON
POST
PUT
PUT
API
Backend
Server
Database

Program

```python
# main.py
if __name__ == "__main__":
    main()
```

```cpp
// main.cpp
int main(void) {
    // do something...
    return 0;
}
```

```
// index.js index.ts
app.listen(8080, () => {
  console.log('Listening on port 8080')
})
```

Demonstration

Prerequisite

NodeJs

```
# Mac
brew install node
# Windows
winget install -e --id OpenJS.NodeJS
```

```
git clone https://github.com/GNITOAHC/gdsc-backend-tutorial.git
```

RESTful API

GET/PUT/POST/DELETE

- *GET:* 獲得資料
- *PUT:* 更改資料
- *POST:* 新增資料
- *DELETE:* 刪除資料

Code Review

```
/* API post ( upload image ) */
app.post('/upload', upload.single('image'), (_req, res) => {
  res.send('File uploaded') // Send message back to frontend
})
```

```
/* API get ( display image ) */
app.get('/display', async (req, res) => {
  const name = req.query.name
  // Send file to frontend for display
  res.sendFile(name as string, { root: './upload' })
})
```

```
/* API delete ( delete uploaded image ) */
app.delete('/delete', async (req, res) => {
  const name = req.query.name
  // Delete file from database
  fs.unlinkSync('./upload/' + name)
  res.send('File deleted')
})
```

Start Server

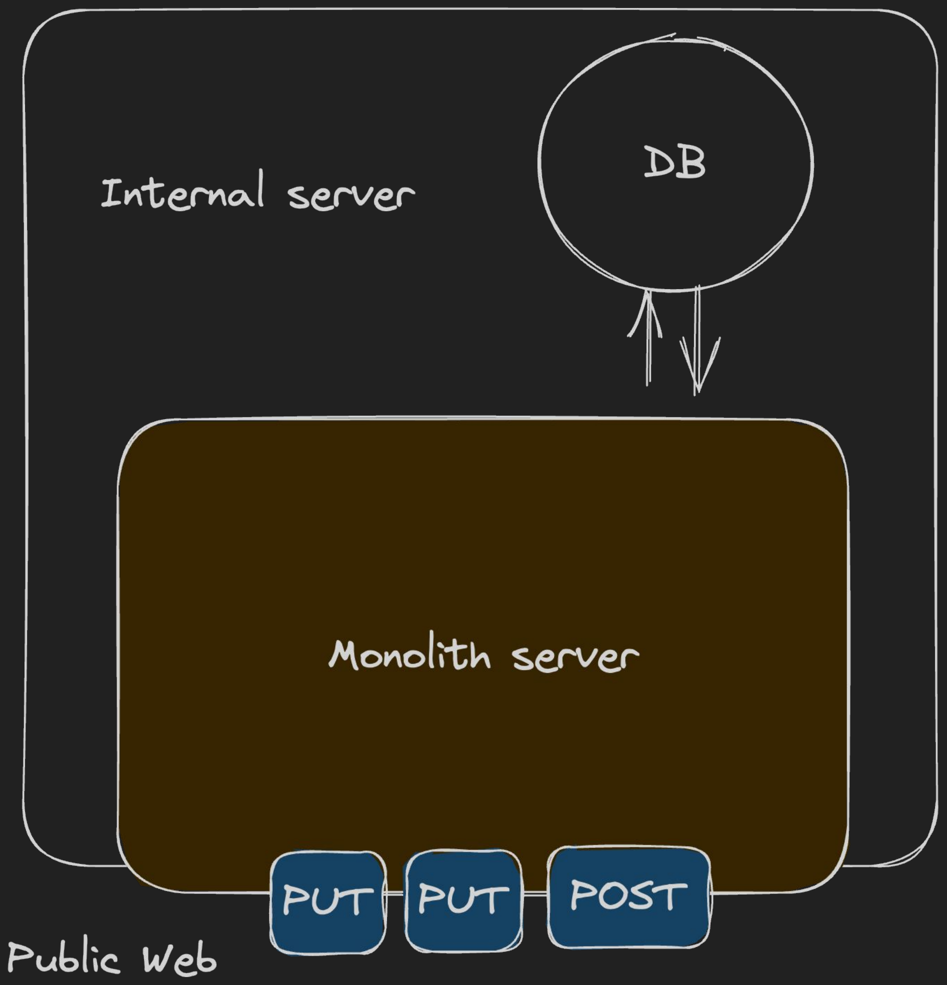
```
# cd into repo
cd gdsc-backend-tutorial/server
# start the server
npm install && npm run dev
```

Now, open the index.html inside gdsc-backend-tutorial via web browser

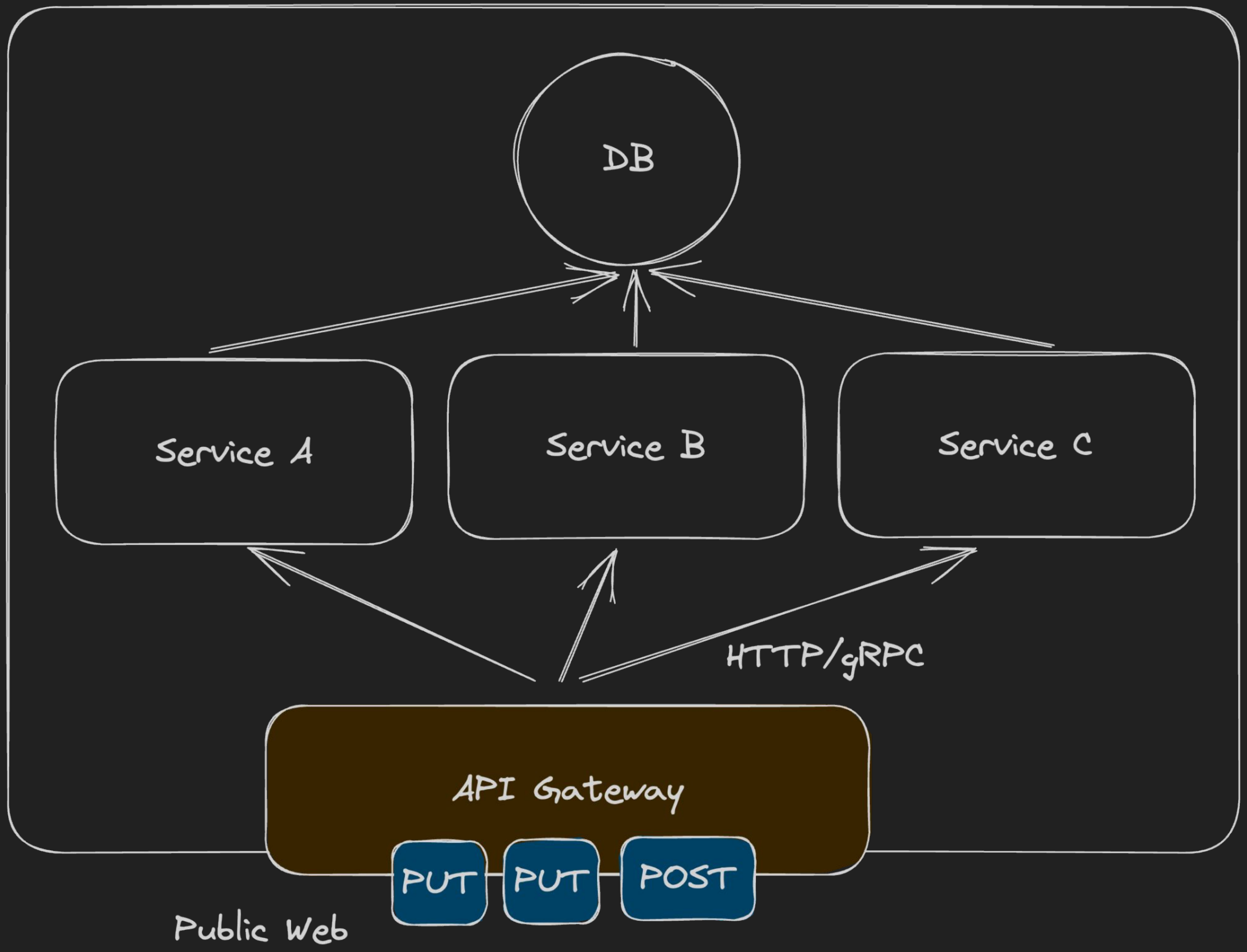
Appendix: Architecture

1. Monolith
2. Microservice
3. Serverless

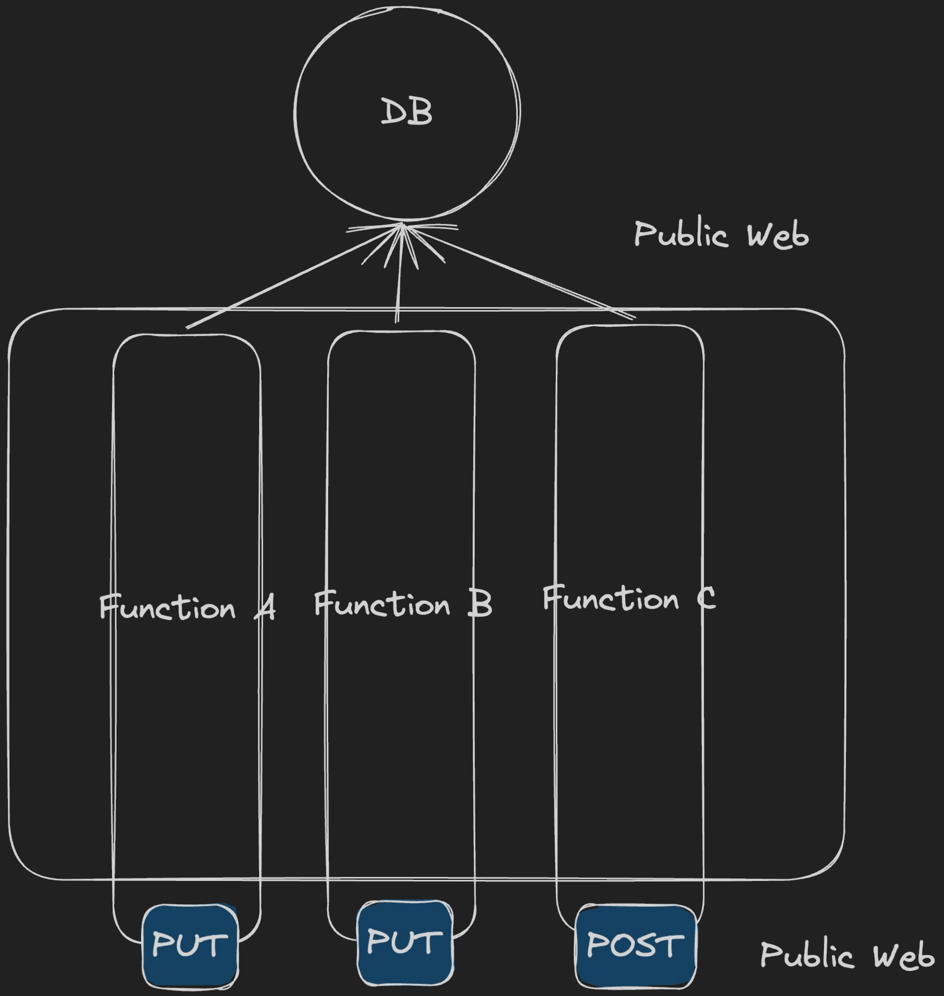
1. Monolith
2. Microservice
3. Serverless


Internal server
DB
Monolith server
PUT
PUT
POST
Public Web

1. Monolith
2. Microservice
3. Serverless


DB
Service A
Service B
Service C
HTTP/gRPC
API Gateway
PUT
PUT
POST
Public Web

1. Monolith
2. Microservice
3. Serverless


DB
Public Web
Function A
Function B
Function C
PUT
PUT
POST
Public Web

Contact me

chaotingchen10@gmail.com