ResCon'25

Fullstack at half the cost

and twice the performance

RESOURCIO COMMUNITY

ResCon'25

Sayan

Engineer at YC Startup

RESOURCIO COMMUNITY

So what is FullStack????

The Hackathon Stack

Next.js

Prisma

Mongo

The Hackathon Stack

Next.js

Prisma

Mongo

clerk

shadcn + tailwind

The Hackathon Stack

Next.js

/

clerk

shadcn + tailwind

So what wrong with this….

NOTHING, IT'S PERFECT

- very very popular technologies

- very very popular technologies
- insanely fast idea to product time

- very very popular technologies
- insanely fast idea to product time
- you can be up and running in seconds

But…

There are some big big problems…

it's kinda expensive
$$$

$

This is your daily notification that your team [redacted] has used **24166% of your monthly included Serverless Function Execution amount** which has added **$96,280** to your bill thus far. You'll continue to be charged **$40 per 100 GB Hrs**.

$

it's kinda expensive
$$$

$

$

$

This is your daily notification that your team [redacted] has used **24166% of your monthly included Serverless Function Execution amount** which has added **$96,280** to your bill thus far. You'll continue to be charged **$40 per 100 GB Hrs**.

it's kinda expensive

$$$

r/Firebase • 7 mo. ago
Glamiris

Firebase bill of 121,000 for last 2 days

Billing

My firebase cost jumped from under $50 per month to $121,000 for last 2 days. using translate and it ran millions of times due to error in code.How do I resolve one time pass on this.Did anyone else face this and how did they resolve this?

This is your daily notification
24166% of your monthly in
amount which has added **$9**
to be charged **$40 per 100 G**

r/webdev • 3 mo. ago
liubanghoudai24

Netlify just sent me a $104K bill for a simple static site

Question

So I received an email from Netlify last weekend saying that I have a $104,500.00 bill overdue. At first I t
joke or some scam email but after checking my dashboard it seems like I am truly owing them 104K do

it's kinda expensive $$$

r/Firebase • 7 mo. ago
Glamiris

Firebase bill of 121,000 for last 2 days

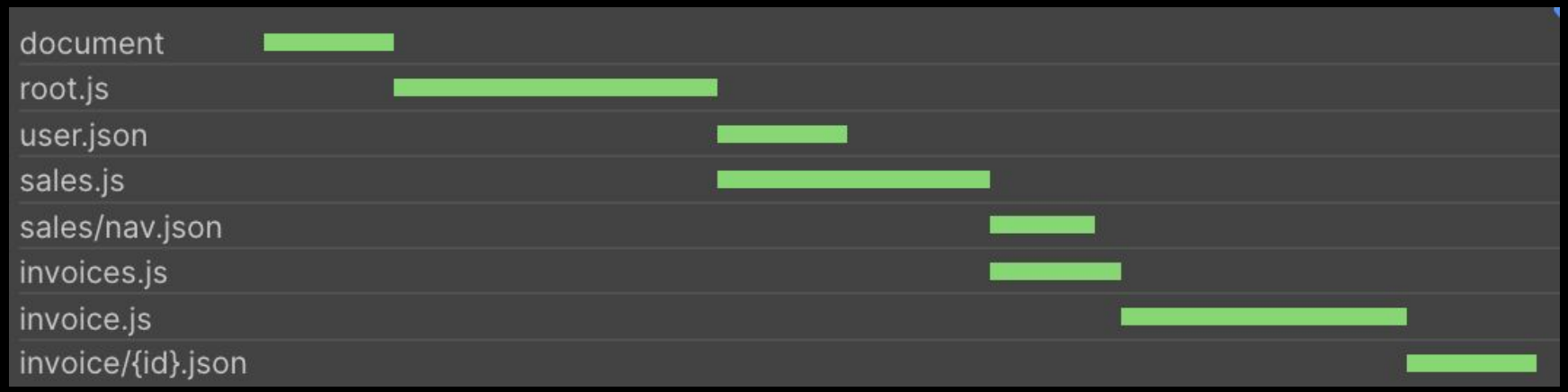
Billing

My firebase cost jumped from under $50 per month to $121,000 for last 2 days.
using translate and it ran millions of times due to error in code.How do I resolve
one time pass on this.Did anyone else face this and how did they resolve this?

also, it's doesn't scale well

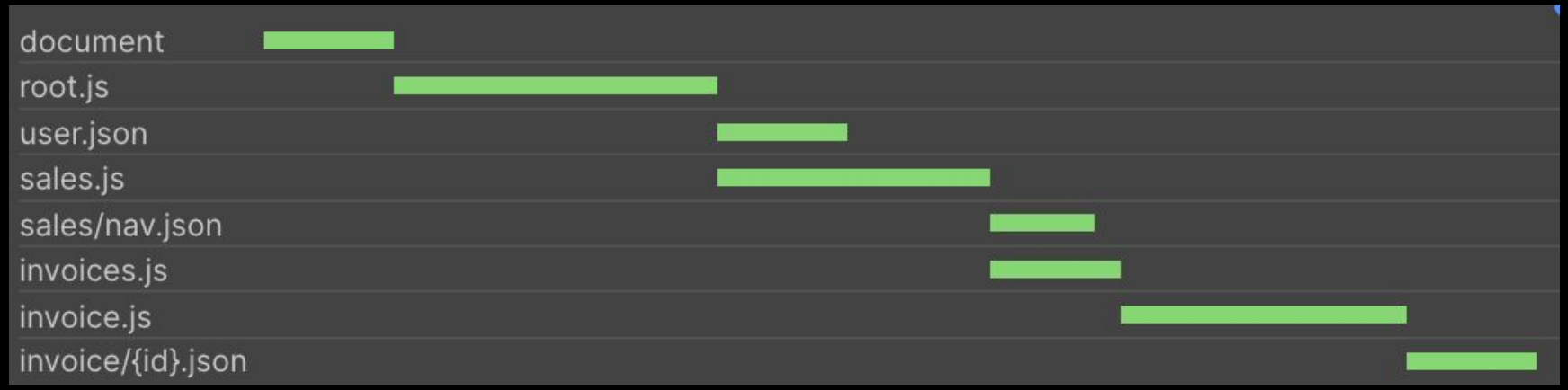
also, it's doesn't scale well

Prisma	TypeORM
8.00ms	5.24ms
`prisma.customer.findMany()`	`AppDataSource.getRepository(Customer).find()`

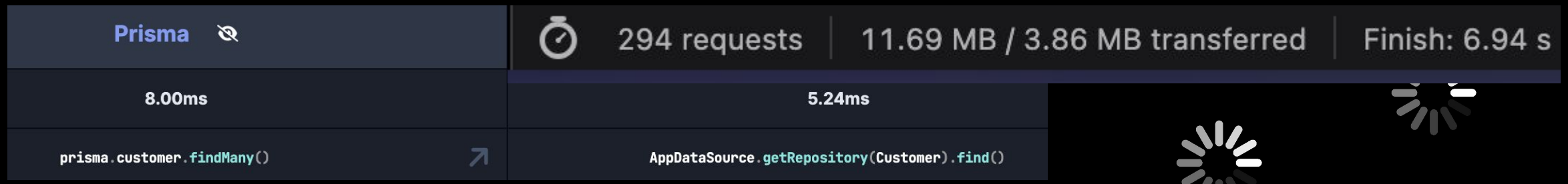


also, it's doesn't scale well

Prisma	TypeORM
8.00ms	5.24ms
`prisma.customer.findMany()`	`AppDataSource.getRepository(Customer).find()`



also, it's doesn't scale well



abstractions are good but you
NEED to understand the tech

Authorization Bypass in Next.js Middleware

Critical severity | GitHub Reviewed | Published last week in **vercel/next.js**

Vulnerability details | Dependabot alerts 0

Package	Affected versions	Patched versions
next (npm)	>= 13.0.0, < 13.5.9	13.5.9
	>= 14.0.0, < 14.2.25	14.2.25
	>= 15.0.0, < 15.2.3	15.2.3
	>= 11.1.4, < 12.3.5	12.3.5

gaining access to anyones browser without them even visiting a website

and of course, firebase was the cause (CVE-2024-45489)

https://kibty.town/blog/arc/

how to gain code execution on millions of people and hundreds of popular apps

and of course, firebase was (partially) the cause

https://kibty.town/blog/todesktop/

and many many
more…

we do much better

ResCon'25

lmnzx

Test Leaderboard

WPM 0

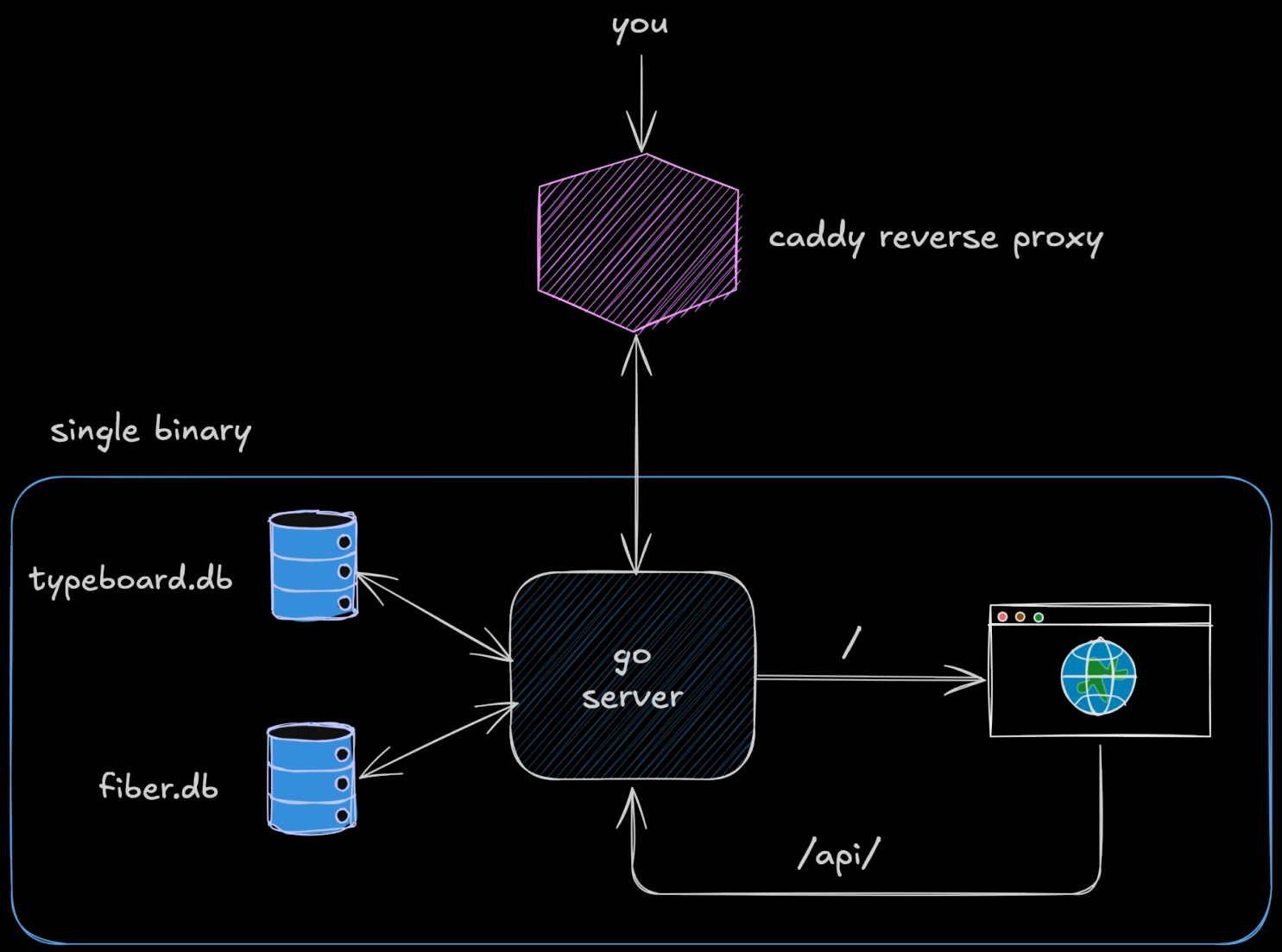
TIME 14s

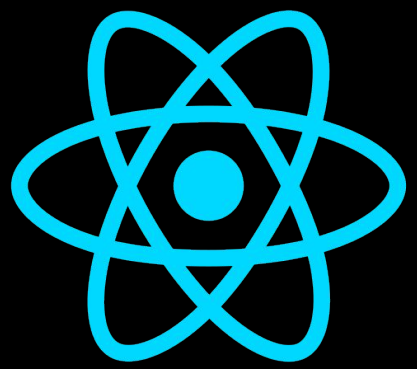
the internet is a global network of billions of computers and other electronic devices with the internet it is possible to access almost any information communicate with anyone else in the world and do much more you can do all this by connecting a computer to the internet which is also called going online

press space to skip to next word

typeboard.lemon.software

a simpler stack






GO


SQLite

frontend

```
"dependencies": {
  "@radix-ui/react-avatar": "^1.1.3",
  "@radix-ui/react-slot": "^1.1.2",
  "class-variance-authority": "^0.7.1",
  "clsx": "^2.1.1",
  "lucide-react": "^0.363.0",
  "react": "^18.2.0",
  "react-dom": "^18.2.0",
  "react-router-dom": "^7.4.1",
  "tailwind-merge": "^2.6.0",
  "tailwindcss-animate": "^1.0.7"
},
"devDependencies": {
  "@types/node": "^20.11.30",
  "@types/react": "^18.2.66",
  "@types/react-dom": "^18.2.22",
  "@vitejs/plugin-react": "^4.2.1",
  "autoprefixer": "^10.4.18",
  "postcss": "^8.4.35",
  "tailwindcss": "^3.4.1",
  "typescript": "^5.2.2",
  "vite": "^5.1.6"
},
```

as simple as it gets

- react for reactivity
- react-router for routing
- shadcn & tailwind for styling
- vite to build and bundle everything

auth is not that hard

```
const AuthContext = createContext<AuthContextType | undefined>(undefined)

export function AuthProvider({ children }: { children: ReactNode }) {
    const [user, setUser] = useState<User>(null)
    const [isLoading, setIsLoading] = useState(true)

    useEffect(() => {
        const checkLoginStatus = async () => {
            try {
                const response = await fetch("/api/auth/status", {
                    credentials: "include",
                })

                if (response.ok) {
                    const userData = await response.json()
                    setUser(userData)
                }
            } catch (error) {
                console.error(error)
            } finally {
                setIsLoading(false)
            }
        }

        checkLoginStatus()
    }, [])

    const login = () => {
        window.location.href = "/api/login"
    }

    return <AuthContext.Provider value={{ user, isLoading, login }}>{children}</AuthContext.Provider>
}
```

auth is not that hard

```
const AuthContext = createContext<AuthContextType | undefined>(undefined)

export function AuthProvider({ children }: { children: ReactNode }) {
    const [user, setUser] = useState<User>(null)
    const [isLoading, setIsLoading] = useState(true)

    useEffect(() => {
        const checkLoginStatus = async () => {
            try {
                const response = await fetch("/api/auth/status", {
                    credentials: "include",
                })

                if (response.ok) {
                    const userData = await response.json()
                    setUser(userData)
                }
            } catch (error) {
                console.error(error)
            } finally {
                setIsLoading(false)
            }
        }

        checkLoginStatus()
    }, [])

    const login = () => {
        window.location.href = "/api/login"
    }

    return <AuthContext.Provider value={{ user, isLoading, login }}>{children}</AuthContext.Provider>
}
```

simple user state

auth is not that hard

```
const AuthContext = createContext<AuthContextType | undefined>(undefined)

export function AuthProvider({ children }: { children: ReactNode }) {
    const [user, setUser] = useState<User>(null)
    const [isLoading, setIsLoading] = useState(true)

    useEffect(() => {
        const checkLoginStatus = async () => {
            try {
                const response = await fetch("/api/auth/status", {
                    credentials: "include",
                })

                if (response.ok) {
                    const userData = await response.json()
                    setUser(userData)
                }
            } catch (error) {
                console.error(error)
            } finally {
                setIsLoading(false)
            }
        }

        checkLoginStatus()
    }, [])

    const login = () => {
        window.location.href = "/api/login"
    }

    return <AuthContext.Provider value={{ user, isLoading, login }}>{children}</AuthContext.Provider>
}
```

simple user state

checking the api for user's status

auth is not that hard

```
const AuthContext = createContext<AuthContextType | undefined>(undefined)

export function AuthProvider({ children }: { children: ReactNode }) {
    const [user, setUser] = useState<User>(null)
    const [isLoading, setIsLoading] = useState(true)

    useEffect(() => {
        const checkLoginStatus = async () => {
            try {
                const response = await fetch("/api/auth/status", {
                    credentials: "include",
                })

                if (response.ok) {
                    const userData = await response.json()
                    setUser(userData)
                }
            } catch (error) {
                console.error(error)
            } finally {
                setIsLoading(false)
            }
        }

        checkLoginStatus()
    }, [])

    const login = () => {
        window.location.href = "/api/login"
    }

    return <AuthContext.Provider value={{ user, isLoading, login }}>{children}</AuthContext.Provider>
}
```

simple user state

checking the api for user's status

setting the state

auth is not that hard

```
const AuthContext = createContext<AuthContextType | undefined>(undefined)

export function AuthProvider({ children }: { children: ReactNode }) {
    const [user, setUser] = useState<User>(null)
    const [isLoading, setIsLoading] = useState(true)

    useEffect(() => {
        const checkLoginStatus = async () => {
            try {
                const response = await fetch("/api/auth/status", {
                    credentials: "include",
                })

                if (response.ok) {
                    const userData = await response.json()
                    setUser(userData)
                }
            } catch (error) {
                console.error(error)
            } finally {
                setIsLoading(false)
            }
        }

        checkLoginStatus()
    }, [])

    const login = () => {
        window.location.href = "/api/login"
    }

    return <AuthContext.Provider value={{ user, isLoading, login }}>{children}</AuthContext.Provider>
}
```

simple user state

checking the api for user's status

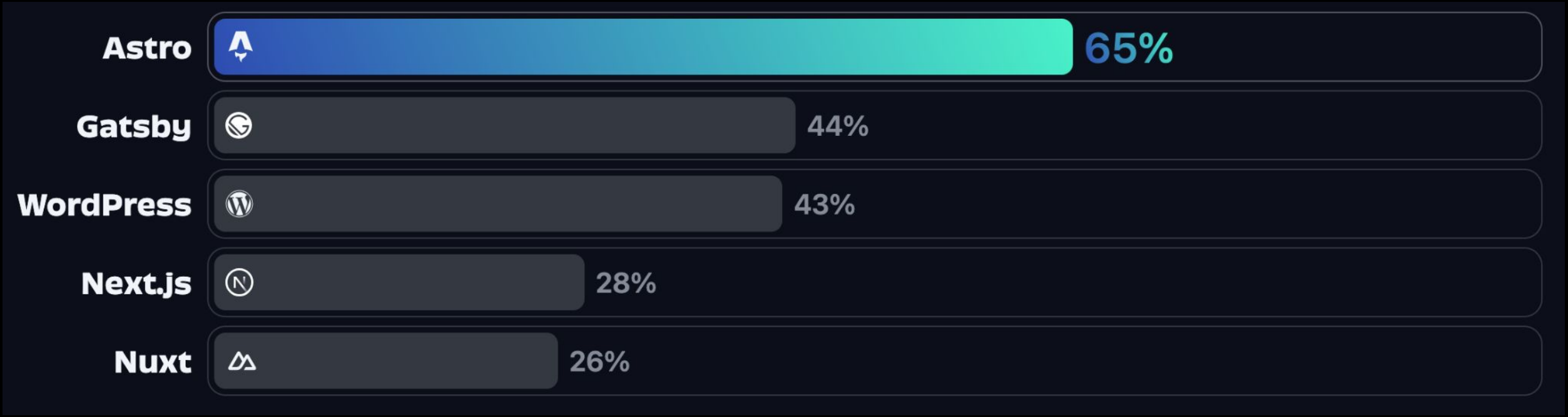
setting the state

login logic handled by the backend

keep your frontend as light as possible

maybe you don't even need react

> **jQuery is used by 91.0%** of all the websites whose JavaScript library we know. **This is 74.4% of all websites**.



benchmark from astro's website, your use case might differ

there are much faster alternatives

Remix

there are much faster alternatives

Remix

FRESH

there are much faster alternatives

Remix

FRESH

there are much faster alternatives

SOLIDJS

Remix

FRESH

there are mucl
alternatives

SOLIDJS

Remix

FRESH

there are mucl
alternatives

SOLIDJS

no more
javascript

backend time

backend

```go
func main() {
    logger := slog.New(slog.NewJSONHandler(os.Stderr, nil))
    slog.SetDefault(logger)

    app := fiber.New(fiber.Config{
        JSONEncoder: json.Marshal,
        JSONDecoder: json.Unmarshal,
        Prefork:     true,
    })

    app.Use(AuthMiddleware())

    app.Get("/api/login", InitiateGitHubLogin)
    app.Get("/api/login/callback", HandleGitHubCallback)
    app.Get("/api/auth/status", GetAuthStatus)

    app.Get("/api/health_check", HealthCheck)
    app.Post("/api/submit", SubmitTypeTest)
    app.Get("/api/leaderboard", cache.New(cache.Config{
        Expiration:   30 * time.Second,
        CacheControl: true,
    }), GetLeaderBoard)

    app.Static("/", "/root/typeboard/dist/")
    app.Get("/*", func(ctx *fiber.Ctx) error {
        return ctx.SendFile("/root/typeboard/dist/index.html")
    })

    app.Listen(":3000")
}
```

backend

```go
func main() {
    logger := slog.New(slog.NewJSONHandler(os.Stderr, nil))
    slog.SetDefault(logger)

    app := fiber.New(fiber.Config{
        JSONEncoder: json.Marshal,
        JSONDecoder: json.Unmarshal,
        Prefork:     true,
    })

    app.Use(AuthMiddleware())

    app.Get("/api/login", InitiateGitHubLogin)
    app.Get("/api/login/callback", HandleGitHubCallback)
    app.Get("/api/auth/status", GetAuthStatus)

    app.Get("/api/health_check", HealthCheck)
    app.Post("/api/submit", SubmitTypeTest)
    app.Get("/api/leaderboard", cache.New(cache.Config{
        Expiration:   30 * time.Second,
        CacheControl: true,
    }), GetLeaderBoard)

    app.Static("/", "/root/typeboard/dist/")
    app.Get("/*", func(ctx *fiber.Ctx) error {
        return ctx.SendFile("/root/typeboard/dist/index.html")
    })

    app.Listen(":3000")
}
```

making
json fast

backend

```go
func main() {
    logger := slog.New(slog.NewJSONHandler(os.Stderr, nil))
    slog.SetDefault(logger)

    app := fiber.New(fiber.Config{
        JSONEncoder: json.Marshal,
        JSONDecoder: json.Unmarshal,
        Prefork:     true,
    })

    app.Use(AuthMiddleware())

    app.Get("/api/login", InitiateGitHubLogin)
    app.Get("/api/login/callback", HandleGitHubCallback)
    app.Get("/api/auth/status", GetAuthStatus)

    app.Get("/api/health_check", HealthCheck)
    app.Post("/api/submit", SubmitTypeTest)
    app.Get("/api/leaderboard", cache.New(cache.Config{
        Expiration:   30 * time.Second,
        CacheControl: true,
    }), GetLeaderBoard)

    app.Static("/", "/root/typeboard/dist/")
    app.Get("/*", func(ctx *fiber.Ctx) error {
        return ctx.SendFile("/root/typeboard/dist/index.html")
    })

    app.Listen(":3000")
}
```

making
json fast

github
oauth

ResCon'25

backend

```go
func main() {
    logger := slog.New(slog.NewJSONHandler(os.Stderr, nil))
    slog.SetDefault(logger)

    app := fiber.New(fiber.Config{
        JSONEncoder: json.Marshal,
        JSONDecoder: json.Unmarshal,
        Prefork:     true,
    })

    app.Use(AuthMiddleware())

    app.Get("/api/login", InitiateGitHubLogin)
    app.Get("/api/login/callback", HandleGitHubCallback)
    app.Get("/api/auth/status", GetAuthStatus)

    app.Get("/api/health_check", HealthCheck)
    app.Post("/api/submit", SubmitTypeTest)
    app.Get("/api/leaderboard", cache.New(cache.Config{
        Expiration:   30 * time.Second,
        CacheControl: true,
    }), GetLeaderBoard)

    app.Static("/", "/root/typeboard/dist/")
    app.Get("/*", func(ctx *fiber.Ctx) error {
        return ctx.SendFile("/root/typeboard/dist/index.html")
    })

    app.Listen(":3000")
}
```

making
json fast

github
oauth

endpoints

ResCon'25

backend

```go
func main() {
    logger := slog.New(slog.NewJSONHandler(os.Stderr, nil))
    slog.SetDefault(logger)

    app := fiber.New(fiber.Config{
        JSONEncoder: json.Marshal,
        JSONDecoder: json.Unmarshal,
        Prefork:     true,
    })

    app.Use(AuthMiddleware())

    app.Get("/api/login", InitiateGitHubLogin)
    app.Get("/api/login/callback", HandleGitHubCallback)
    app.Get("/api/auth/status", GetAuthStatus)

    app.Get("/api/health_check", HealthCheck)
    app.Post("/api/submit", SubmitTypeTest)
    app.Get("/api/leaderboard", cache.New(cache.Config{
        Expiration:   30 * time.Second,
        CacheControl: true,
    }), GetLeaderBoard)

    app.Static("/", "/root/typeboard/dist/")
    app.Get("/*", func(ctx *fiber.Ctx) error {
        return ctx.SendFile("/root/typeboard/dist/index.html")
    })

    app.Listen(":3000")
}
```

making json fast

github oauth

endpoints

serving frontend

ResCon'25

backend

```go
func main() {
    logger := slog.New(slog.NewJSONHandler(os.Stderr, nil))
    slog.SetDefault(logger)

    app := fiber.New(fiber.Config{
        JSONEncoder: json.Marshal,
        JSONDecoder: json.Unmarshal,
        Prefork:     true,
    })

    app.Use(AuthMiddleware())

    app.Get("/api/login", InitiateGitHubLogin)
    app.Get("/api/login/callback", HandleGitHubCallback)
    app.Get("/api/auth/status", GetAuthStatus)

    app.Get("/api/health_check", HealthCheck)
    app.Post("/api/submit", SubmitTypeTest)
    app.Get("/api/leaderboard", cache.New(cache.Config{
        Expiration:   30 * time.Second,
        CacheControl: true,
    }), GetLeaderBoard)

    app.Static("/", "/root/typeboard/dist/")
    app.Get("/*", func(ctx *fiber.Ctx) error {
        return ctx.SendFile("/root/typeboard/dist/index.html")
    })

    app.Listen(":3000")
}
```

making json fast

github oauth

endpoints

caching

serving frontend

why i chose golang

why i choose golang

- all languages are great and use what you like

why i choose golang

- all languages are great and use what you like
- very easy to implement complex logic

why i choose golang

- all languages are great and use what you like
- very easy to implement complex logic
- great ecosystem for api development

why i chose golang

- all languages are great and use what you like
- very easy to implement complex logic
- great ecosystem for api development
- easy to deploy, the entire app is a single binary

why i choose golang

```go
type TypeTestRequest struct {
    Wpm      int `json:"wpm"`
    Accuracy int `json:"accuracy"`
}

type Claims struct {
    Username string `json:"username"`
    jwt.RegisteredClaims
}

type User struct {
    ID        int    `json:"id"`
    Login     string `json:"login"`
    Name      string `json:"name"`
    AvatarURL string `json:"avatar_url"`
}

type LeaderboardEntry struct {
    Login    string `json:"login"`
    WPM      int    `json:"max_wpm"`
    Accuracy int    `json:"accuracy"`
}
```

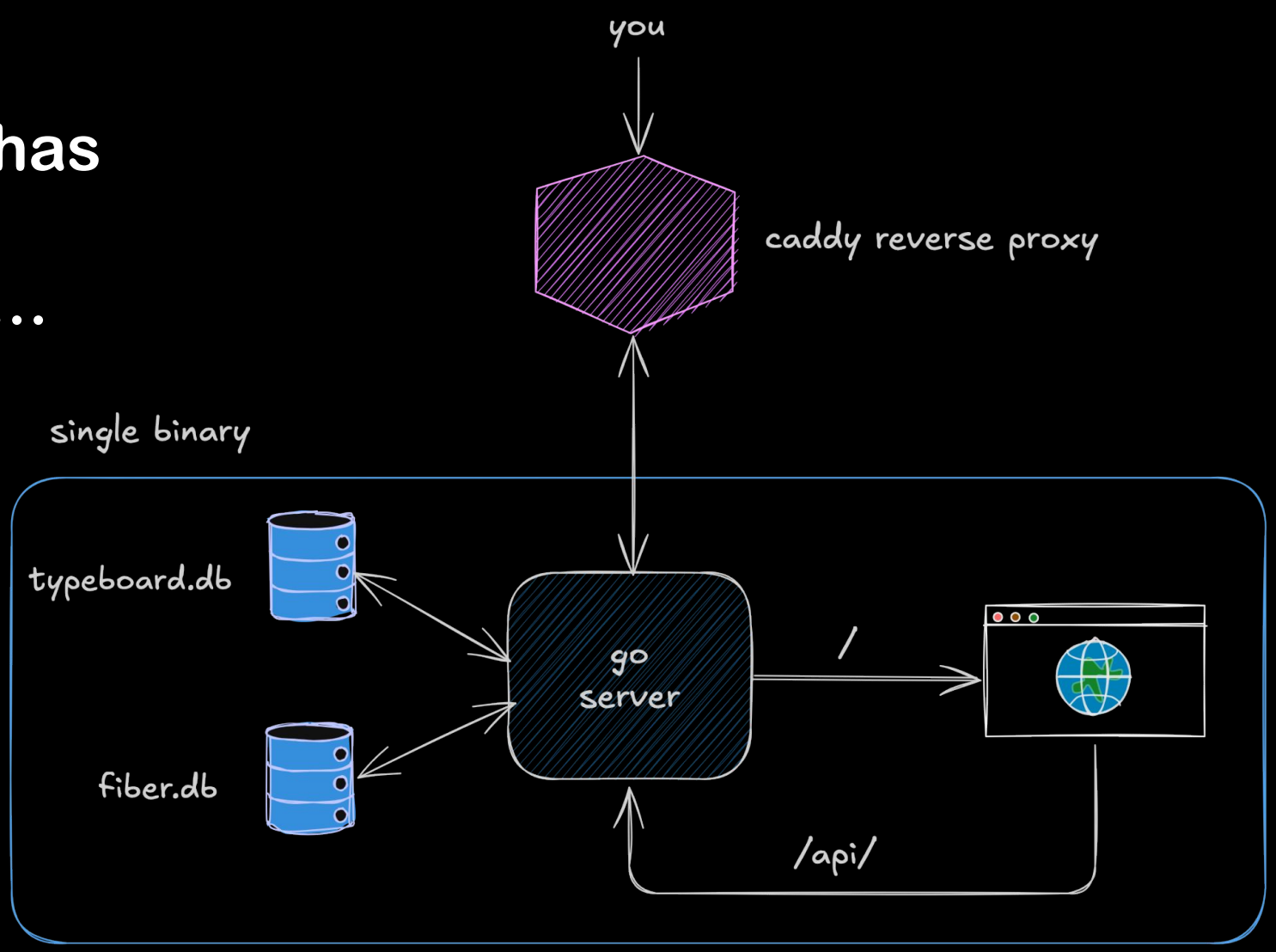
why i choose golang

```go
func AuthMiddleware() fiber.Handler {
    return func(c *fiber.Ctx) error {
        tokenString := ""
        cookie := c.Cookies("token")
        if cookie != "" {
            tokenString = cookie
        } else if len(tokenString) > 7 && tokenString[:7] == "Bearer " {
            tokenString = tokenString[7:]
        }

        if tokenString == "" {
            return c.Next()
        }

        claims := &Claims{}
        token, err := jwt.ParseWithClaims(tokenString, claims, func(token *jwt.Token) (any, error) {
            return jwtSecret, nil
        })

        if err != nil || !token.Valid {
            return c.Next()
        }

        c.Locals("username", claims.Username)
        return c.Next()
    }
}
```

why i choose golang

```go
func (u *User) InsertNewUser() {
    _, err := typeboardDb.Exec(`
        INSERT INTO users (id, login, name, avatar_url)
        VALUES (?, ?, ?, ?)
        `, u.ID, u.Login, u.Name, u.AvatarURL)
    if err != nil {
        fmt.Println(err)
    }
}
```

this still has one BIG problem…



let's talk about databases

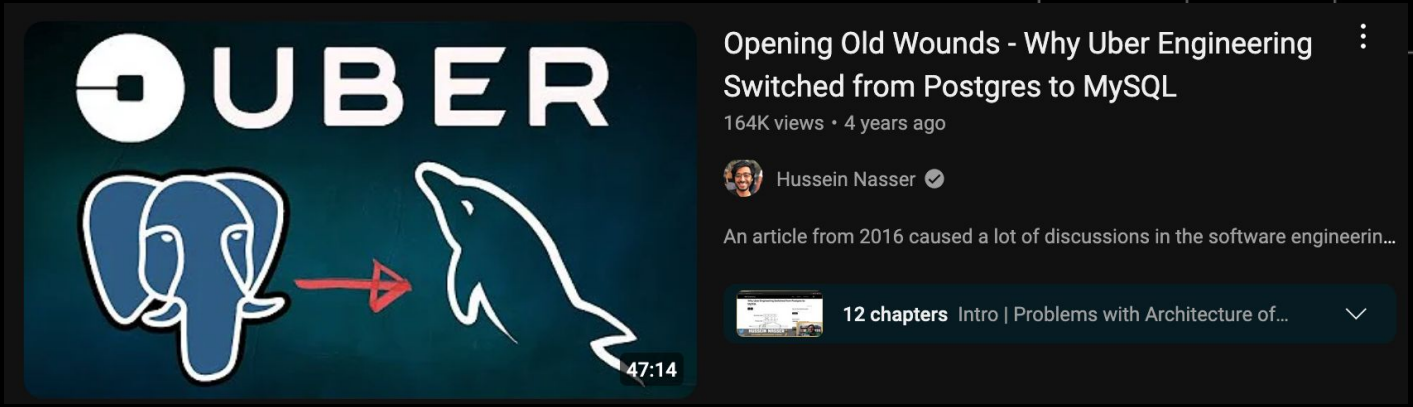
picking the right database is difficult

~424 different databases

https://db-engines.com/en/ranking

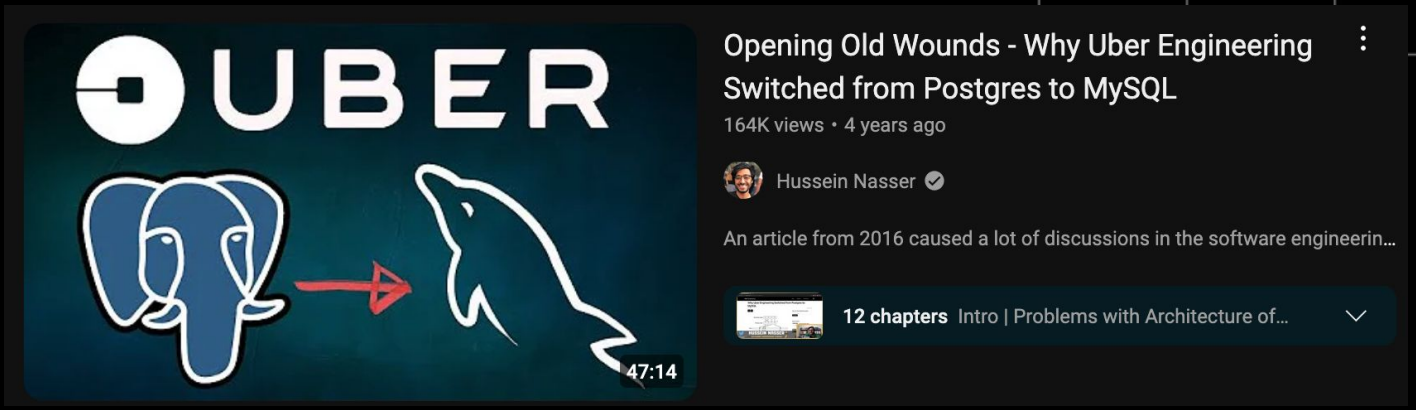
~424 different databases

https://db-engines.com/en/ranking



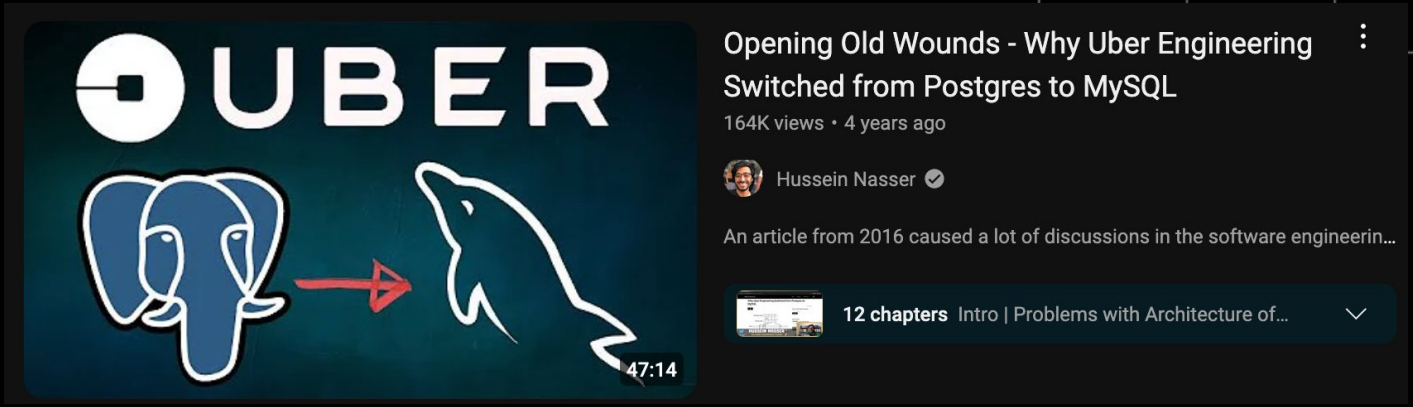
~424 different databases

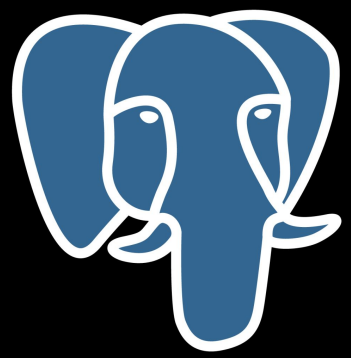
https://db-engines.com/en/ranking



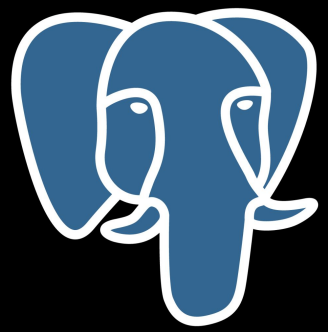
~424 different databases

https://db-engines.com/en/ranking



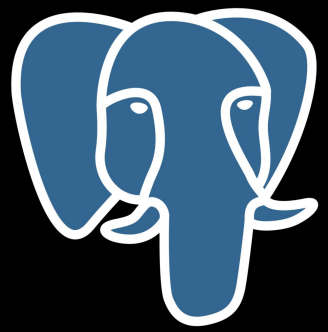


for structure data, users, payments…



for structure data, users, payments…

for caching, sessions, real-time, pub-sub….



for structure data, users, payments…

for caching, sessions, real-time, pub-sub….

unstructured data, blog post, product description…

auth is kinda hard
😥

auth is hard, and there is no easy way out of this
clerk, firebase makes it easy use with the frontend but hard to integrate with the backend and easy to mess up the config for potential risk

roll your own auth is a shooting yourself in the foot

oauth is a sweet middleground

all this makes the backend very difficult 🥺

well we can settle for something in the middle

PocketBase

Open Source backend in **1** file

convex

pocketbase.io

convex.dev

why go through all this trouble???

why go through all this trouble???

- you are in control

why go through all this trouble???

- you are in control
- you decide where to host it

why go through all this trouble???

- you are in control
- you decide where to host it
- doing hard things are fun

why go through all this trouble???

- you are in control
- you decide where to host it
- doing hard things are fun
- you will be learning a lot about the underling abstractions

why go through all this trouble???

- you are in control
- you decide where to host it
- doing hard things are fun
- you will be learning a lot about the underling abstractions
- explore the alternative and think a lot before you write the first line of code

ResCon '25

Thank you all!!!

All the side and repo