



Squeezing the model

How to create a full stack app generator



Mihovil Ilakovac

Full-stack developer who
likes building stuff

I like contributing to open
source ([Directus](#), [Caprover](#))

Freelanced, worked at
enterprises and startups

Wasp 



ilakovac.com



@infomiho

Let's talk about the Magic App Generator

This is a Wasp powered project. If you like it, [star us on GitHub!](#) ☆ Star 17,851

**MAGE ✨ GPT Web App Generator**
Generate your full-stack web app in Wasp, React, Node.js and Prisma

Help ⓘ 👤

- Waiting for instructions

App name *

e.g. TodoApp or MyPlants

Description *

Describe your web app in a couple of sentences!
Mention pages you want + what should be happening on them, what data should be stored in the database, etc (check out the examples below).
The simpler and more specific the app is, the better the generated app will be.

App brand color

sky

Creativity level

Balanced

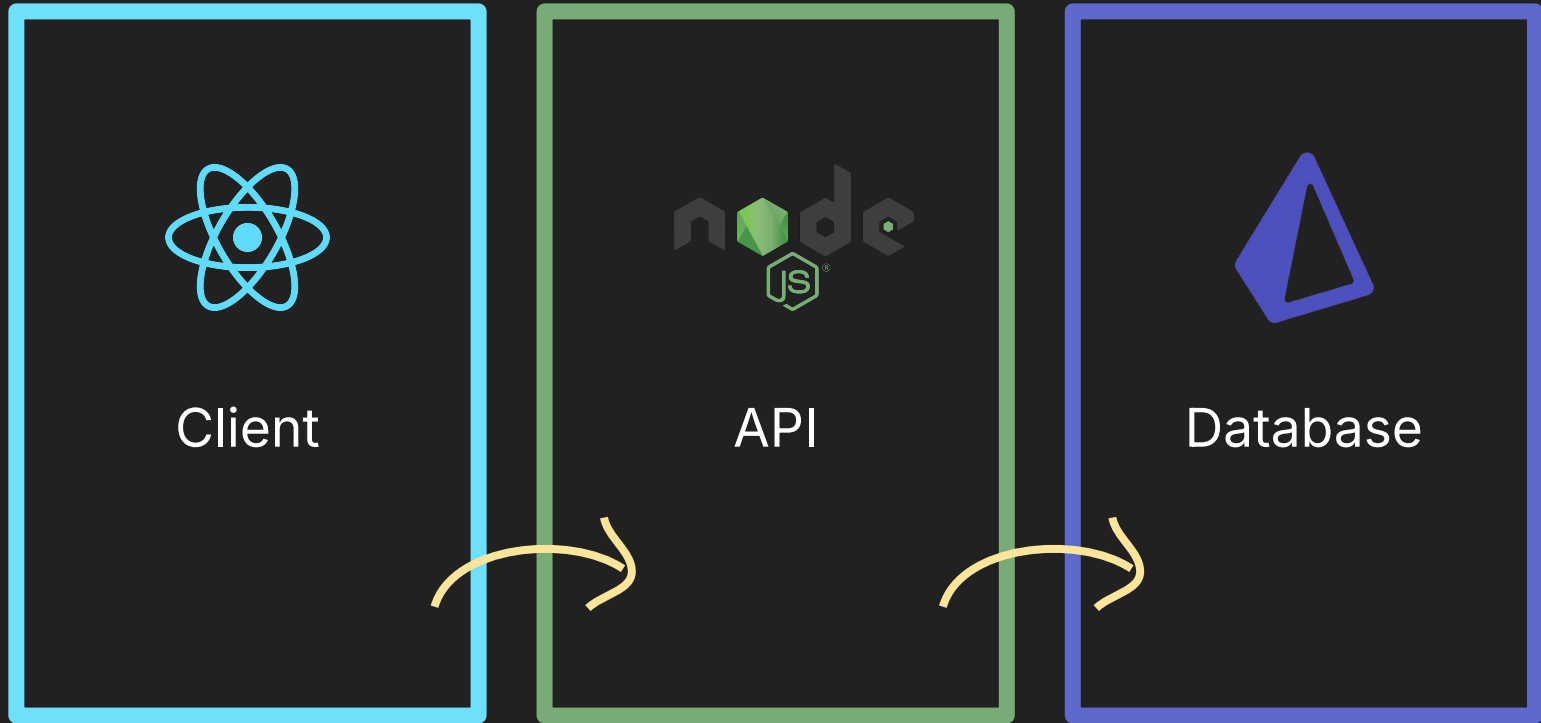
Auth method

Username & Password

Generate the app ⚡

July 2023

We wanted to generate full-stack apps



When GPT 3.5 was all the rage



+ New chat

Web Dev Help Offered

Explaining Node JS Popularity

Clear conversations

Dark mode

OpenAI Discord

Updates & FAQ

Log out



Generate code for a page with a play button over a picture



Sure! Here is some HTML code that will create a page with a play button over an image:



Copy code

```
<!DOCTYPE html>
<html>
<head>
  <title>Play Button Over Image</title>
</head>
<body>
  <style>
    /* Style the play button */
    .play-button {
      position: absolute;
      top: 50%;
      left: 50%;
      transform: translate(-50%, -50%);
      background-color: #fff;
      border: none;
      outline: none;
      cursor: pointer;
      box-shadow: 0 3px 6px rgba(0,0,0,0.23);
      border-radius: 50%;
    }
  </style>

```

Regenerate response

It wasn't looking good at first...



Username

Password

Log in

If you don't have an account go to [sign up](#).

My Todo List

New Task

Add Task

Use GPT to build a web app



Launch it on Reddit



Open source everything



Task × +

⌕

Filters

None

Fields

All

Showing 3 of 3

Add record

☐	id #	description A	isDone ☐	user ☐	userId #
☐	1	Use GPT to build a web app	true	User	1
☐	2	Launch it on Reddit	false	User	1
☐	3	Open source everything	true	User	1

Tournament Name

Description of the tournament

Home Team vs Away Team	Match Date
------------------------	------------

Nexus

Investment

Item

Item

Shipping

Ship

Notifications

Noti

Noti

FIFA2024 – Powered



Nexus – Powered by Wasp

John Doe! (Log out)

Remove

Remove

Powered by Wasp

Update Status

We built a system around GPT 3.5

Plan first

Skeleton

Compiler

Pre-built blocks

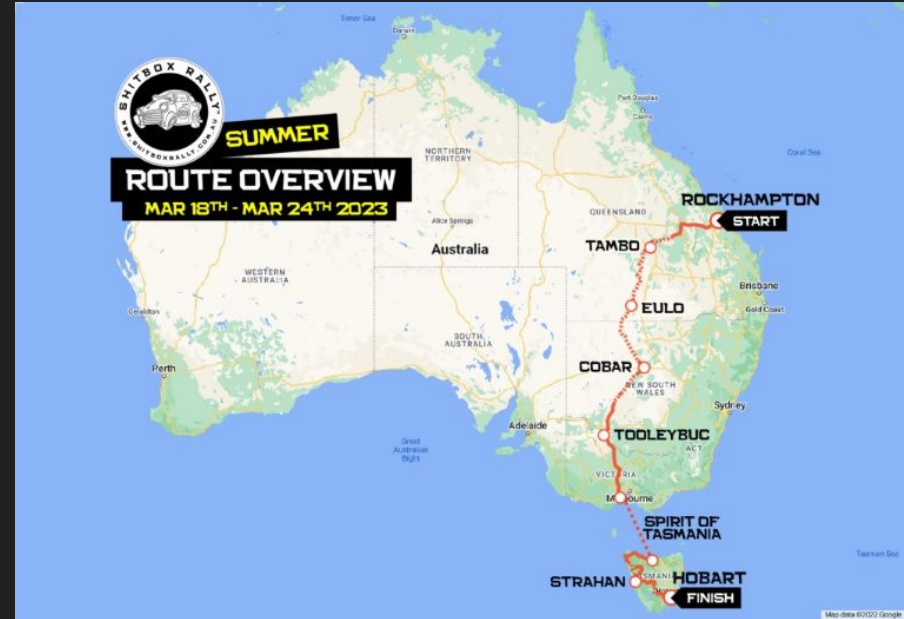
High-level spec

```
🤖 Generating a new Wasp project named TodoApp!
⌚ Generating project skeleton...
✅ Generated project skeleton.
⌚ Generating plan (slowest step, usually takes 30 to 90 seconds)
✅ Initial plan generated!
- 1 queries: ["getTasks"]
- 3 actions: ["createTask", "toggleTask", "editTaskDescription"]
- 2 entities: ["User", "Task"]
- 1 pages: ["DashboardPage"]
⌚ Fixing initial plan...
✅ Plan fixed!
✅ Updated the Wasp file with entities.
⌚ Generating actions
```

Today's goal:

**Take a cheap model and build
a full-stack app generator**

Sh*tbox rally



<https://www.abc.net.au/news/2023-03-18/s-box-rally-rockhampton-to-hobart-cancer-research/102110962>

Vibe coded engineered demo tool

Session: 469535a9-aa2a-4422-a60d-cfe968ecd7ab

Timeline (60) Files (31) Preview (running) Replay Mode Present

● **SYSTEM** Completing

12:48:23.620

Generation completed. Tokens: 56455, Cost: \$0.0238, Duration: 620.8s

📄 ● validateTypeScript

12:48:20.060

Click for details

📄 ● validatePrismaSchema

12:48:18.380

Click for details

🔧 ● installDependencies

12:48:16.671

Click for details

⚠️ ● **SYSTEM** Validation

12:48:16.669

✓ No errors found

🛠️ ASSISTANT

12:48:16.667

I fixed the reported Prisma schema validation errors and the two TypeScript issues. Summary of changes:

1. Prisma schema (prisma/schema.prisma)

- Problem: Ambiguous relation in PRRequestedReviewer (two fields pointed to GitHubUser without relation names), plus subsequent validation errors requiring opposite relation fields to be declared.
- Fixes:
 - Added explicit relation names to PRRequestedReviewer.reviewer and PRRequestedReviewer.requestedBy:
 - reviewer @relation(name: "PRRequestedReviewer_reviewer", ...)
 - requestedBy @relation(name: "PRRequestedReviewer_requestedBy", ...)
 - Added explicit relation names to PullRequest.author and PRReview.reviewer so they can be paired with opposite fields:
 - author @relation(name: "GitHubUser_authoredPRs", ...)
 - reviewer @relation(name: "GitHubUser_reviews", ...)
 - Added opposite/back-relation fields on GitHubUser for all relations pointing to it:
 - authoredPullRequests PullRequest[] @relation("GitHubUser_authoredPRs")
 - requestedByRequests PRRequestedReviewer[] @relation("PRRequestedReviewer_requestedBy")

Gen Fullstack

Capabilities

- Code Generation
- Smart Planning
- Template Base
- Auto Error-Fixing
- Building Blocks

Prompt

PR tracking app

- loads data from Github (user enters their own key)
- loads PRs from certain repos

user specifies

- shows a list of PRs with statuses and some filtering e.g. review requested from some user

Metrics

Status

Generation ● completed

Container ● running
<http://localhost:5001>

■ Stop

App Preview

Settings

Default perPage30

Save

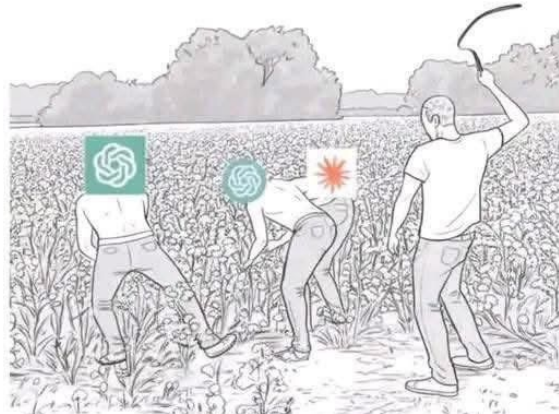
Container Logs

15:31:44 INFO 43 packages are looking for funding run 'npm fund' for details
15:31:44 INFO found 0 vulnerabilities
15:31:44 SYS Dependencies installed
15:31:44 SYS Generating Prisma client...
15:31:44 CMD \$ npx prisma generate -x 1.8s

Other People With ChatGPT:



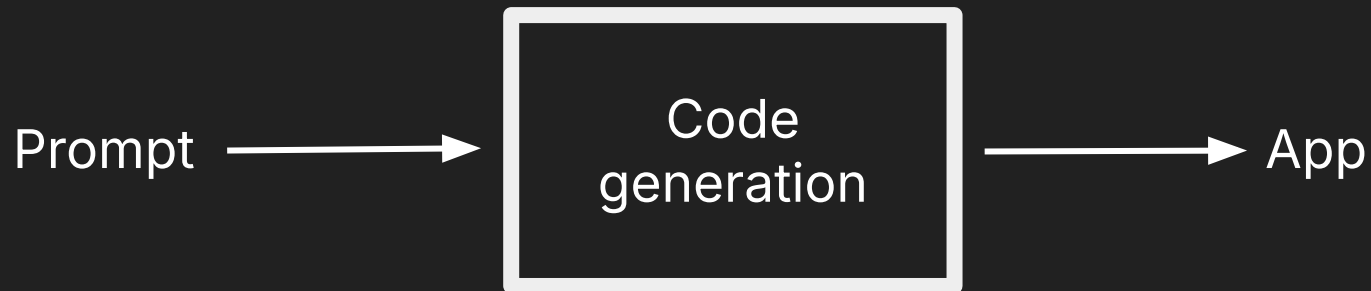
Random IT Guy At 3 AM:



Let's generate:

Pull request tracking app

Initial approach



Tools

writeFile

readFile

getFileTree

Full-stack apps system prompt

ARCHITECTURE:

You are building a monorepo with npm workspaces:

- Root package.json with **workspaces**: ["client", "server"]
- **client/** - Vite + React 19 + TypeScript +
Tailwind 4 + React Router 7
- **server/** - Express 5 + TypeScript with RESTful API
- **prisma/** - Prisma ORM + SQLite database

Demo #1

Code gen demo

[Open demo](#)

Plan architecture first



What is a “plan”?

```
{  
  "databaseModels": [...],  
  "apiRoutes": [...],  
  "clientRoutes": [...],  
  "clientComponents": [...]  
}
```

Database model

```
{
  "name": "PullRequest",
  "fields": [
    { "name": "id", "type": "Int", "attributes": ["@id", "@default(autoincrement())"] },
    { "name": "githubId", "type": "Int", "attributes": ["@unique"] },
    ...
  ],
  "description": "Stores GitHub pull request data"
}
```

API route

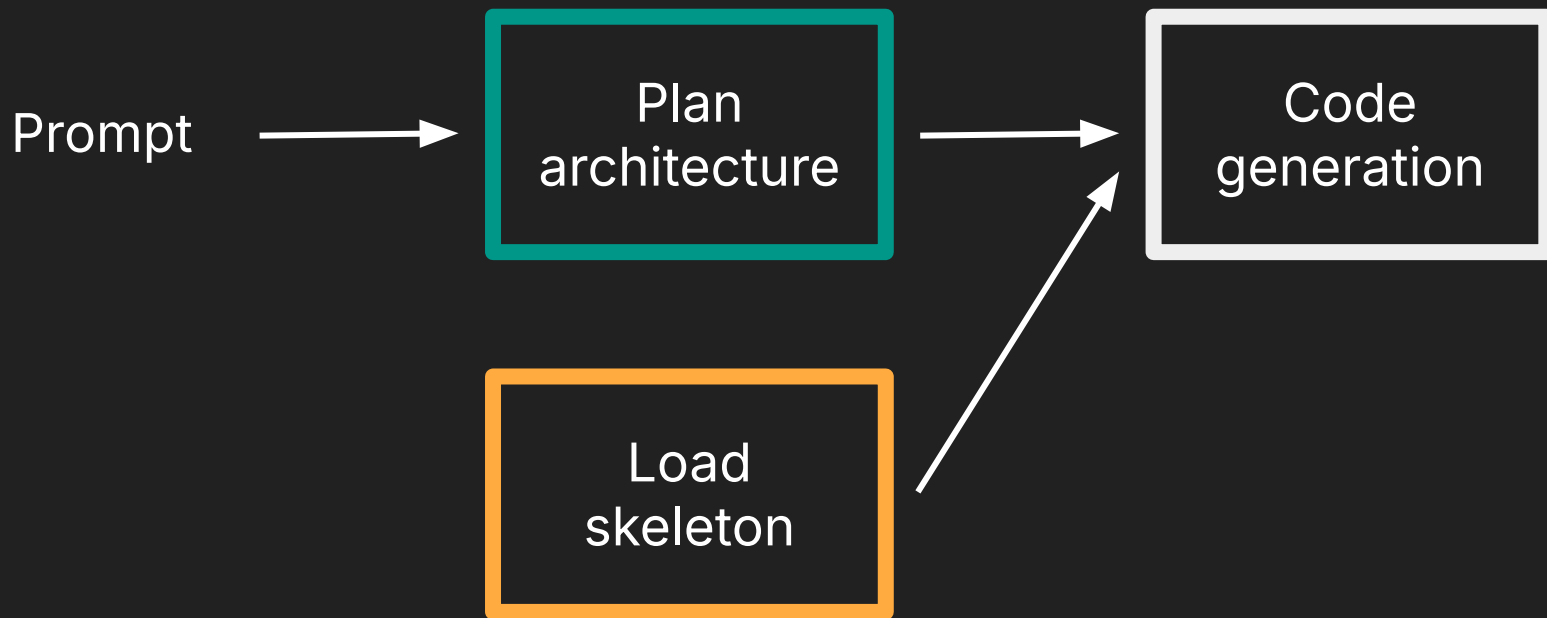
```
{  
  "method": "GET",  
  "path": "/api/pull-requests",  
  "description": "Fetch all PRs with optional filters",  
  "requestBody": null,  
  "responseBody": "{ pullRequests: PullRequest[] }"  
}
```

Demo #2

Planning demo

[Open demo](#)

Start with a skeleton



The “vite-fullstack-base” template

```
templates/vite-fullstack-base/
├── client/                                # React 19 + Vite frontend
│   ├── src/
│   │   ├── pages/
│   │   │   └── Home.tsx                # Example home page
│   │   ├── App.tsx                    # Main app component with React Router
│   │   ├── index.css                  # Global styles
│   │   └── main.tsx                   # Entry point
│   ├── index.html
│   ├── package.json                   # Client dependencies
│   ├── tsconfig.json
│   ├── tsconfig.node.json
│   └── vite.config.ts
├── server/                              # Express 5 + TypeScript backend
│   ├── src/
│   │   └── index.ts                   # Server with example CRUD API
│   ├── package.json                   # Server dependencies
│   └── tsconfig.json
├── prisma/
│   └── schema.prisma                  # Example User model + SQLite
└── package.json                       # Root package.json with npm workspaces
```

The “vite-fullstack-base” template

```
templates/vite-fullstack-base/
├── client/                                # React 19 + Vite frontend
│   ├── src/
│   │   ├── pages/
│   │   │   └── Home.tsx                # Example home page
│   │   ├── App.tsx                    # Main app component with React Router
│   │   ├── index.css                  # Global styles
│   │   └── main.tsx                   # Entry point
│   ├── index.html
│   ├── package.json                   # Client dependencies
│   ├── tsconfig.json
│   ├── tsconfig.node.json
│   └── vite.config.ts
├── server/                               # Express 5 + TypeScript backend
│   ├── src/
│   │   └── index.ts                   # Server with example CRUD API
│   ├── package.json                   # Server dependencies
│   └── tsconfig.json
├── prisma/
│   └── schema.prisma                  # Example User model + SQLite
└── package.json                       # Root package.json with npm workspaces
```

The “vite-fullstack-base” template

```
templates/vite-fullstack-base/
├── client/                                # React 19 + Vite frontend
│   ├── src/
│   │   ├── pages/
│   │   │   └── Home.tsx                # Example home page
│   │   ├── App.tsx                    # Main app component with React Router
│   │   ├── index.css                  # Global styles
│   │   └── main.tsx                   # Entry point
│   ├── index.html
│   ├── package.json                   # Client dependencies
│   ├── tsconfig.json
│   ├── tsconfig.node.json
│   └── vite.config.ts
├── server/                               # Express 5 + TypeScript backend
│   ├── src/
│   │   └── index.ts                   # Server with example CRUD API
│   ├── package.json                   # Server dependencies
│   └── tsconfig.json
├── prisma/
│   └── schema.prisma                  # Example User model + SQLite
└── package.json                       # Root package.json with npm workspaces
```

Demo #3

Template demo

[Open demo](#)

Pre-made blocks

Code
generation

Tools

writeFile

readFile

genFileTree

Pre-made blocks

Code
generation

Tools

writeFile

readFile

genFileTree

requestBlock



```
client/  
├── LoginForm.tsx      # Login UI component  
├── ProtectedRoute.tsx # Route wrapper for auth  
├── RegisterForm.tsx   # Registration UI component  
└── useAuth.tsx        # Auth context + hooks
```

```
server/  
├── auth-middleware.ts # requireAuth middleware  
├── auth-routes.ts     # API routes  
└── auth.ts            # Auth utilities
```

```
prisma/  
└── schema.prisma     # User & Session models
```

username-auth block

block.json

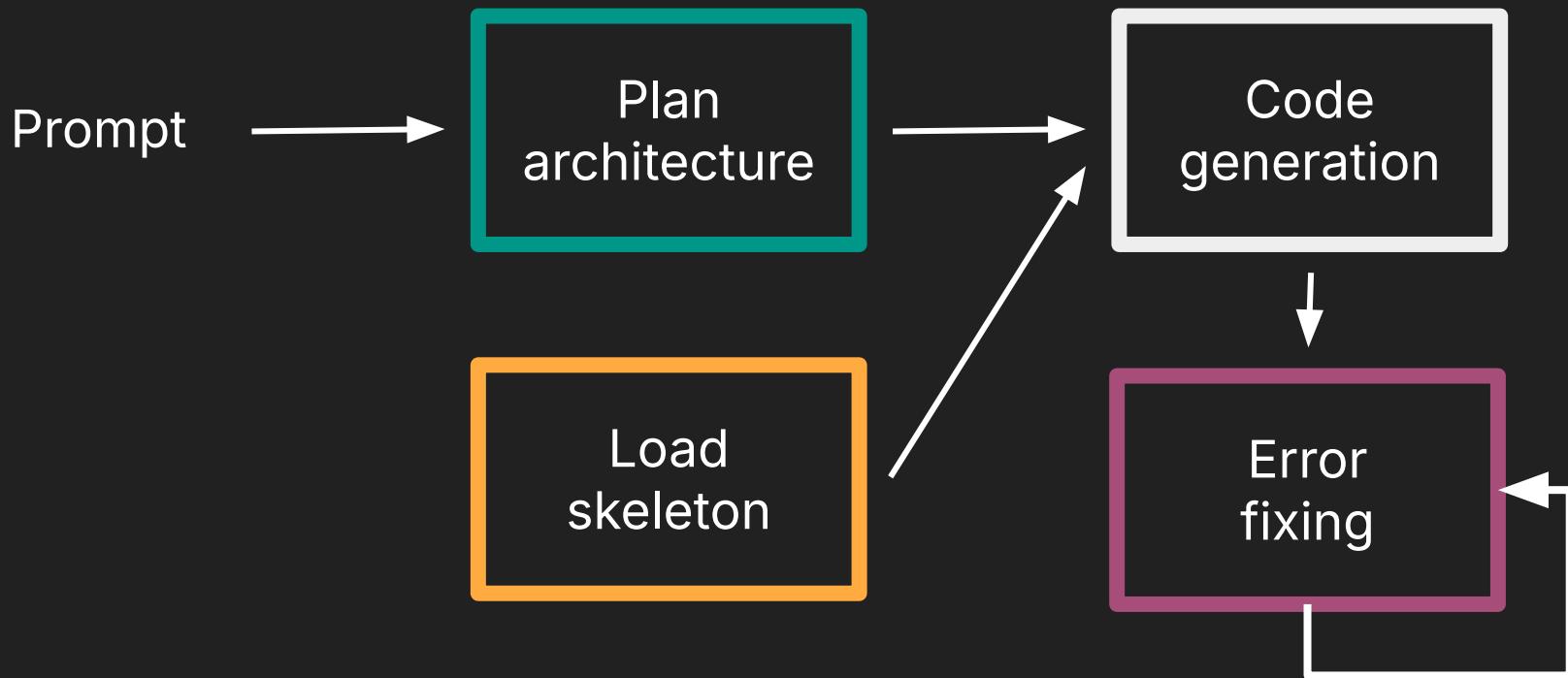
```
... ],
... "prisma": ["prisma/schema.prisma"]
... },
... "integrationGuide": {
...   "steps": [
...     "1. Install dependencies: npm install bcryptjs @types/bcryptjs",
...     "2. Merge Prisma models from prisma/blocks/auth-password.prisma into your main prisma/schema.prisma",
...     "3. Run: npx prisma generate && npx prisma migrate dev --name add-auth",
...     "4. Import and mount auth router in server/src/index.ts:",
...     "   import { authRouter } from './auth-routes';",
...     "   app.use('/api/auth', authRouter);",
...     "5. Wrap your app with AuthProvider in the root component (React Router 7):",
...     "   Option A -- In client/src/root.tsx or client/src/routes/_layout.tsx:",
...     "     import { Outlet } from 'react-router';",
...     "     import { AuthProvider } from './useAuth';",
...     "     export default function Root() {",
...     "       return <AuthProvider><Outlet /></AuthProvider>;",
...     "     }",
...     "   Option B -- In client/src/main.tsx (wrap the entire app):",
...     "     import { AuthProvider } from './useAuth';",
...     "     <AuthProvider><RouterProvider router={router} /></AuthProvider>",
...     "6. Use useAuth() hook in components to access user, login, register, logout",
...     "7. Use <ProtectedRoute> component to guard authenticated routes"
...   ],
... }
```


Demo #4

Pre-made blocks demo

[Open demo](#)

Check the code with compilers



What kind of checks can we run?



Type check the code



Prisma compiler



Import/export validation



Validate package.json

...

**Anything that we can
parse and validate!**

Demo #5

Compiler checks demo

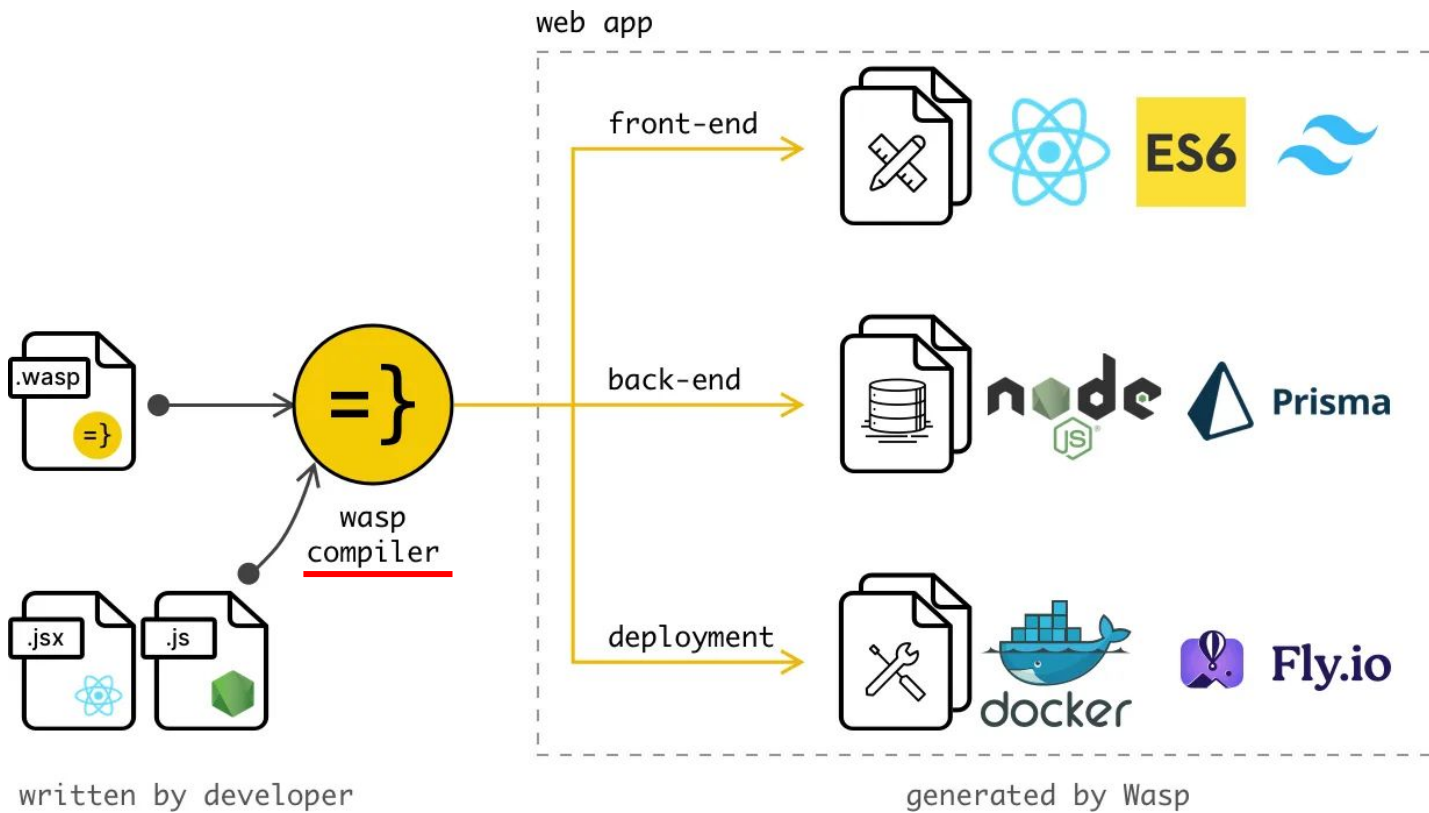
[Open demo](#)

Higher level spec as the last trick

todoApp.wasp

```
app todoApp {  
  title: "ToDo App", // visible in the browser tab  
  auth: { // full-stack auth out-of-the-box  
    userEntity: User,  
    methods: { google: {}, gitHub: {}, email: {...} }  
  }  
}  
  
route RootRoute { path: "/", to: MainPage }  
page MainPage {  
  authRequired: true, // Limit access to logged in users.  
  component: import Main from "@client/Main.tsx" // Your React component  
}  
  
query getTasks {  
  fn: import { getTasks } from "@server/tasks.js", // Your Node.js function  
  entities: [Task] // Automatic cache invalidation.  
}
```

Wasp is a full-stack JS framework



How frameworks aid LLMs

```
resources :photos
```

[COPY](#)

creates seven different routes in your application, all mapping to the `PhotosController` actions:

HTTP Verb	Path	Controller#Action	Used to
GET	/photos	photos#index	display a list of all photos
GET	/photos/new	photos#new	return an HTML form for creating a new photo
POST	/photos	photos#create	create a new photo
GET	/photos/:id	photos#show	display a specific photo
GET	/photos/:id/edit	photos#edit	return an HTML form for editing a photo
PATCH/PUT	/photos/:id	photos#update	update a specific photo
DELETE	/photos/:id	photos#destroy	delete a specific photo



How frameworks aid LLMs

```
1  use App\Jobs\ProcessPodcast;  
2  
3  // This job is sent to the default connection's default queue...  
4  ProcessPodcast::dispatch();  
5  
6  // This job is sent to the default connection's "emails" queue...  
7  ProcessPodcast::dispatch()→onQueue('emails');
```

Laravel

How frameworks aid LLMs

Full stack

```
// In a route loader
export const Route = createFileRoute('/posts')({
  loader: () => getPosts(),
})

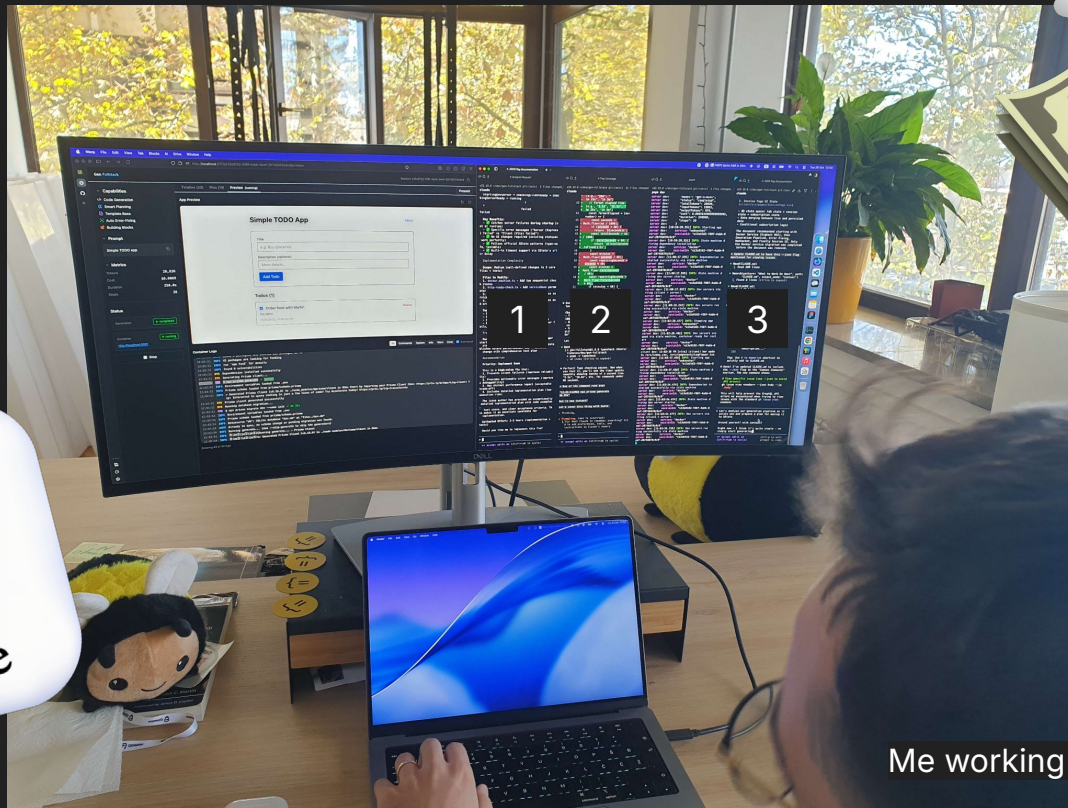
// In a component
function PostList() {
  const getPosts = useServerFn(getServerPosts)

  const { data } = useQuery({
    queryKey: ['posts'],
    queryFn: () => getPosts(),
  })
}
```

TANSTACK START

RC

Has the meta changed?



Me working on the demo app



*Me being excited about a
new idea*

Bolt, Lovable, v0...

bolt

I'll help you create a pull request tracking app. Let me start by understanding the current state of the project and then plan the implementation.

2 files read

Plan

- ✓ Design database schema for pull requests
- ✓ Create migration for pull requests table with RLS
- ✓ Set up Bolt Database client
- ✓ Create PR list component
- ✓ Create PR form component
- ✓ Implement main app layout and state management
- ✓ Run build to verify everything works
- ✓ Plan completed

Planning

Pet Name Picker Web App

Project Settings

- General
- Domains & Hosting
- Analytics
- Database**
- Authentication
- Server Functions
- Secrets
- User Management
- File Storage
- Knowledge
- Backups

Personal Settings

- General
- Subscription & Tokens
- Applications
- Cloud
- Knowledge

Tables

View and manage the data stored in your app.

user_profiles	0 rows	favorite
---------------	--------	----------

Database service

components > pr-filters.tsx

```
1 "use client"
2
3 import { Input } from "@/components/ui/input"
4 import { Label } from "@/components/ui/label"
5 import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select"
6 import { Search } from "lucide-react"
7
8 export interface FilterOptions {
9   search: string
10   status: "all" | "open" | "closed" | "draft"
11   reviewer: string
12 }
13
14 interface PRFiltersProps {
15   filters: FilterOptions
16   onFiltersChange: (filters: FilterOptions) => void
17   reviewers: string[]
18 }
19
20 export function PRFilters({ filters, onFiltersChange, reviewers }) {
21   return (
22     <div className="flex flex-col md:flex-row gap-4 mb-6">
23       <div className="flex-1">
24         <Label htmlFor="search" className="sr-only">
25           Search
26         </Label>
27         <div className="relative">
28           <Search className="absolute left-3 top-1/2 -transform: rotate(-50%);"/>
```

Pre-built components

LLMs are getting better and more accessible

LLMs still can't be 100% trusted! ⚠

LLMs in a human designed system is the best way to extract value

Deterministic tools keep the LLM honest

*Most precise work – is the work LLM
doesn't need to do*

Thank you Belgrade! 🇷🇸