

Deviens full-stack avec TypeScript

Présenté par Estéban Soubiran



Estéban Soubiran

Développeur web full-stack à Maïa Space



Vite



Vue



UnJS



Nuxt



Laravel

2024: Reflections on a Year of Change and What Comes Next

The Perfect Guide to Setting Up a New Nuxt Project

For a Friend: From Infrastructure to Web Development in 2025

Enhancing Integration with Unplugin and Nuxt Module



soubiran.dev



@soubiran_



@soubiran.dev



Estéban S

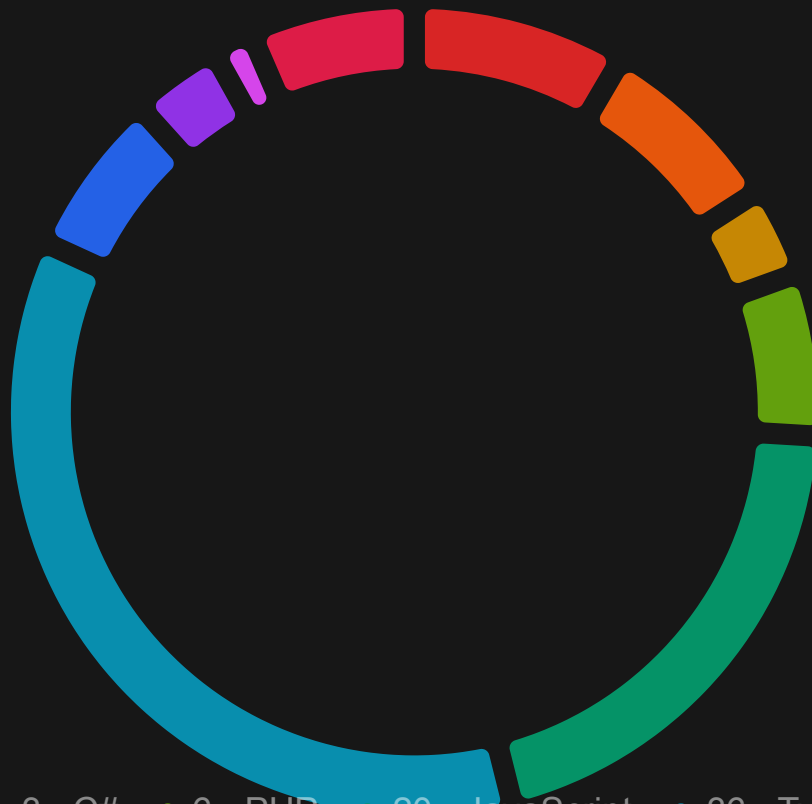


Barbapapazes

Frontend, Backend ou Full-stack ?



Language utilisé pour le backend ?



8 - Python 7 - Java 3 - C# 6 - PHP 20 - JavaScript 36 - TypeScript 6 - Go
0 - C++ 3 - Kotlin 1 - Ruby 6 - Rust



ex




High-scope





Low-scope





 Lucid


 Auth


 Bouncer

 FlyDrive

 Attachment lite


 Limiter


 Edge

 Bentocache


 Vine


 Ally


 I18n


 Heath checks

 Mailer


 Transmit


 Lock

 Japa

 HMR

 Vite


 Inertia

 Tuyau

Concrètement, à quoi ça ressemble ?

TS start/routes.ts

```
import PostsController from '#controllers/posts_controller'  
import router from '@adonisjs/core/services/router'  
  
router.get('posts', [PostsController, 'index'])  
router.post('posts', [PostsController, 'store'])  
router.get('posts/:id', [PostsController, 'show'])
```

 terminal

```
node ace make:controller posts
```


TS app/controllers/http/posts_controller.ts

```
import type { HttpContext } from '@adonisjs/core/http'

export default class PostsController {
  async index({}: HttpContext) {
    // we want to return a paginated list of posts
  }

  async store({}: HttpContext) {
    // we want to save a post
  }

  async show({}: HttpContext) {
    // we want to return a post by its id
  }
}
```

 terminal

```
node ace make:model post
```


TS app/models/post.ts

```
import { BaseModel, column } from '@adonisjs/lucid/orm'

export default class Post extends BaseModel {
  @column()
  declare id: number

  @column()
  declare title: string

  @column()
  declare contents: string
}
```

 terminal

```
node ace make:validator posts
```

TS app/validators/create_post_validator.ts

```
import vine from '@vinejs/vine'

export const createPostValidator = vine.compile(
  vine.object({
    title: vine.string().trim().minLength(4).maxLength(256),
    contents: vine.string().trim()
  })
)
```

TS app/controllers/http/posts_controllers.ts

```
import type { HttpContext } from '@adonisjs/core/http'
import Post from '#models/post'
import { createPostValidator } from '#validators/post'

export default class PostsController {
  async index({ request }: HttpContext) {
    const page = request.input('page', 1)
    return Post.query().paginate(page)
  }

  async store({ request }: HttpContext) {
    const payload = await request.validateUsing(createPostValidator)
    return Post.create(payload)
  }

  async show({ params }: HttpContext) {
    return Post.findOrFail(params.id)
  }
}
```

TS app/http/controllers/session_controller.ts

```
import User from '#models/user'
import { HttpContext } from '@adonisjs/core/http'

export default class SessionController {
  async store({ request }: HttpContext) {
    const { email, password } = request.only(['email', 'password'])

    const user = await User.verifyCredentials(email, password)

    await auth.use('web').login(user)

    response.redirect('/dashboard')
  }
}
```

TS start/routes.ts

```
router.get('/github/redirect', ({ ally }) => {  
  return ally.use('github').redirect()  
})  
  
router.get('/github/callback', async ({ ally }) => {  
  const gh = ally.use('github')  
  
  if (gh.accessDenied()) {  
    return 'You have cancelled the login process'  
  }  
  if (gh.stateMismatch()) {  
    return 'We are unable to verify the request. Please try again'  
  }  
  if (gh.hasError()) {  
    return gh.getError()  
  }  
  
  const user = await gh.user()  
  
  return user  
})
```


TS app/policies/post_policy.ts

```
import Post from '#models/post'
import User from '#models/user'
import { BasePolicy } from '@adonisjs/bouncer'
import { AuthorizerResponse } from '@adonisjs/bouncer/types'

export default class PostPolicy extends BasePolicy {
  create(user: User): AuthorizerResponse {
    return true
  }

  edit(user: User, post: Post): AuthorizerResponse {
    return user.id === post.userId
  }
}
```

TS app/controllers/http/posts_controller.ts

```
import type { HttpContext } from '@adonisjs/core/http'
import Post from '#models/post'
import PostPolicy from '#policies/post_policy'

export default class PostsController {
  async create({ bouncer, response }: HttpContext) {
    await bouncer.with(PostPolicy).authorize('create')

    // create a new post
  }

  async edit({ bouncer, params, response }: HttpContext) {
    const post = await Post.findOrFail(params.id)

    await bouncer.with(PostPolicy).authorize('edit', post)

    // edit the post
  }
}
```

TS tests/http/posts/index.ts

```
import Post from '#models/post'
import { test } from '@japa/runner'

test.group('Posts', () => {
  test('get a list of posts', async ({ client }) => {
    await PostFactory.merge({ title: 'Hello world' }).create()

    const response = await client.get('posts')

    response.assertStatus(200)
    response.assertBody({
      data: [
        {
          id: 1,
          title: 'Hello world',
        }
      ]
    })
  })
})
```

TS app/contracts/repositories_services.ts

```
export abstract class RepositoriesService {  
  abstract repositories(): Promise<Repository[]>  
}
```

TS app/services/github_repositories_service.ts

```
import { RepositoriesService } from '#contracts/services'  
  
export class GitHubRepositoriesService implements RepositoriesService {  
  async repositories() {}  
}
```

TS providers/app_provider.ts

```
import { RepositoriesService } from '#contracts/repositories_services'

export default class AppProvider {
  async boot() {
    const { GitHubRepositoriesService } = await import('#services/github_repositories_service')

    this.app.container.singleton(RepositoriesService, async (resolver) => {
      const configService = await resolver.make('config')
      const token = configService.get<string>('services.github.token')

      return new GitHubRepositoriesService(token)
    })
  }
}
```

TS tests/http/repositories/index.ts

```
import RepositoriesService from '#contracts/repositories_services'
import app from '@adonisjs/core/services/app'






















test('get all repositories', async ({ client }) => {
  class FakeRepositoriesService implements RepositoriesService {
    repositories() {
      return [{ id: 1, repo: 'barbapapazes/talks' }]
    }
  }

  app.container.swap(RepositoriesService, () => {
    return new FakeRepositoriesService()
  })















  const response = await client.get('repositories')

  response.assertBody({
    data: [{ id: 1, repo: 'barbapapazes/talks' }]
  })
})
```
















Folder by Type

-  app
 -  controllers
 -  enums
 -  events
 -  exceptions
 -  listeners
 -  middleware
 -  models
 -  services
 -  validators
-  bin
-  config
-  resources
-  start
-  tests
-  package.json
-  JS ace.js
-  TS adonisrc.ts
-  eslint.config.js
-  tsconfig.json
-  vite.config.ts

Folder by Feature

-  src
 -  users
 -  posts
-  bin
-  config
-  resources
-  start
-  tests
-  package.json
-  JS ace.js
-  TS adonisrc.ts
-  eslint.config.js
-  tsconfig.json
-  vite.config.ts

Hexagonal Architecture

-  src
 -  core
 -  application
 -  infrastructure
-  bin
-  config
-  resources
-  start
-  tests
-  package.json
-  JS ace.js
-  TS adonisrc.ts
-  eslint.config.js
-  tsconfig.json
-  vite.config.ts

Series.

STEP-BY-STEP LEARNING

[Browse All Series](#) →

Building with AdonisJS & Inertia

📖 72 Lessons ⌚ 11h 0m

LATEST FROM THIS SERIES

LESSON 10.5 · Jan 10

Updating & Deleting an Organization

LESSON 10.4 · Jan 10

Account Deletion & Cleaning Dangling Organizations

LESSON 10.3 · Jan 10

Alerting Users When Their Account Email Is Changed



AdonisJS Quick Tip

📖 22 Lessons ⌚ 2h 28m



AdonisJS In 30

📖 9 Lessons ⌚ 2h 58m



Let's Learn AdonisJS 6

📖 113 Lessons ⌚ 14h 9m

LATEST FROM THIS SERIES

LESSON 11.11 · Jun 29, 24

Thank You for Watching!

LESSON 11.10 · Jun 29, 24

Allowing Admins to Delete Movies and their Relationships

LESSON 11.9 · Jun 26, 24

Managed Transactions and Syncing Movie Cast Members



Et maintenant, envie d'essayer Adonis ?



● 38 - Oui ● 6 - Non

Intéressé(e)... ?

- Consulte la documentation docs.adonisjs.com
- Apprends avec adocasts.com
- Rejoins la communauté sur [Discord](#)