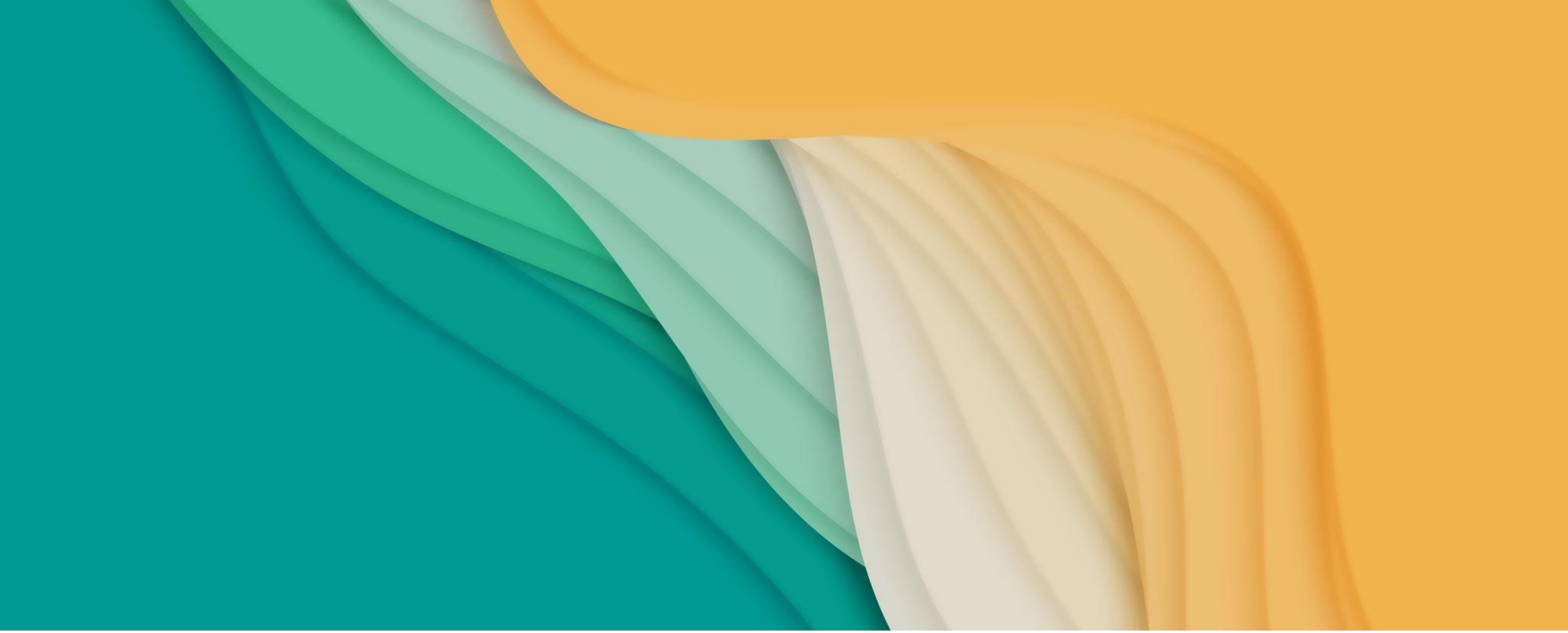


Corso Drupal per sviluppatori

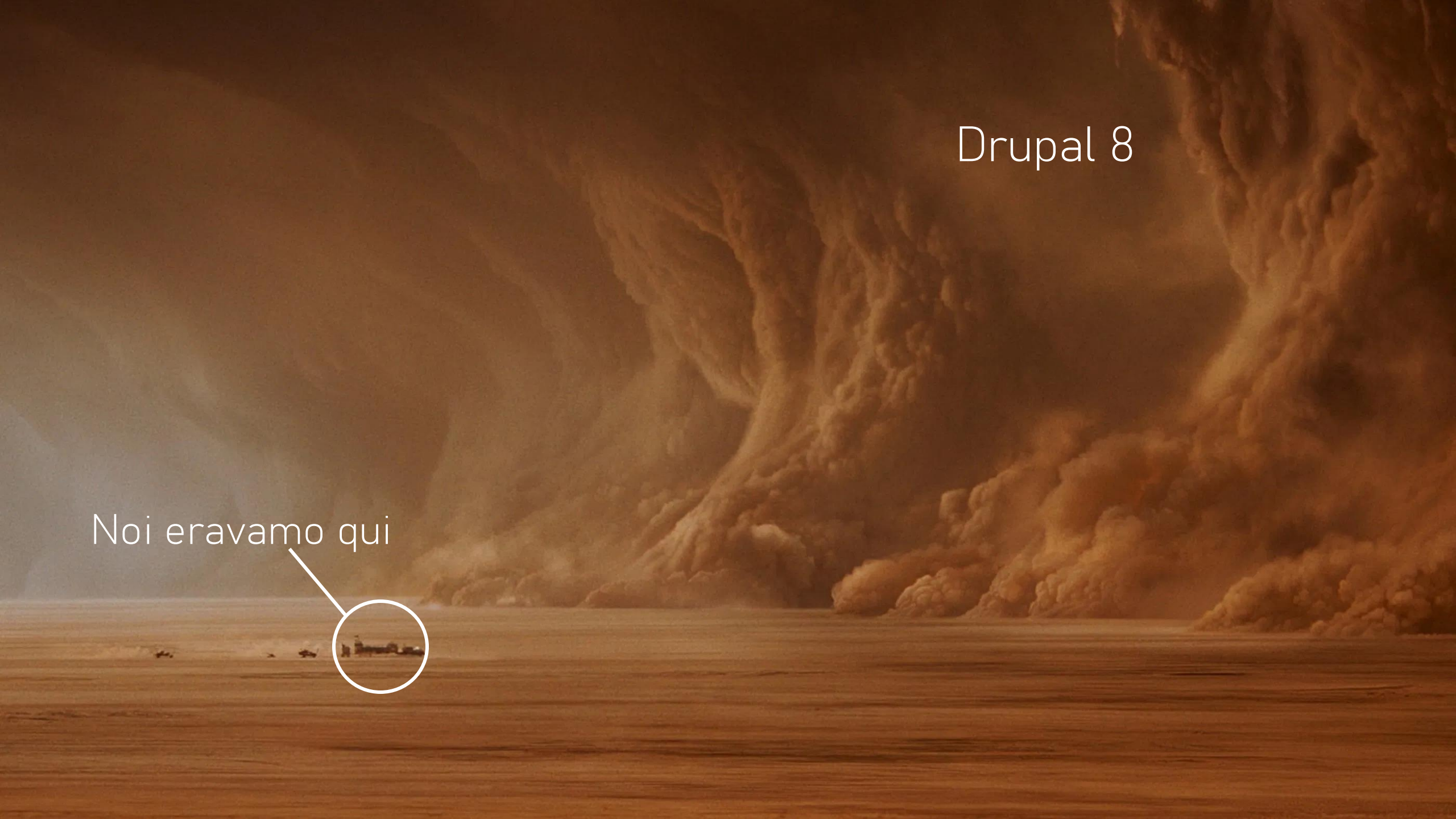
Moduli e temi custom

Programma

- 1 Architettura e basi per lo sviluppo
- 2 Moduli personalizzati: logiche e form
- 3 Theming: personalizzare il frontend
- 4 Integrazione moduli e tema twig

Architettura e basi per lo sviluppo

- Architettura generale di Drupal (Core, moduli, temi, librerie)
- Struttura delle directory e convenzioni principali
- Composer e gestione delle dipendenze
- Strumenti essenziali: Drush, Devel
- Installazione moduli contrib, temi contrib e sistema di aggiornamento


Drupal 8
Noi eravamo qui


I bisonti istintivamente corrono verso la tempesta in arrivo

Struttura delle directory

```
/drupal
│
├── composer.json        → definisce le dipendenze del progetto
├── composer.lock        → blocca le versioni installate
├── vendor/              → librerie PHP installate da Composer
│
└── web/                 → root pubblica del sito
    ├── core/            → codice sorgente di Drupal
    ├── modules/         → moduli contrib e personalizzati
    ├── themes/          → temi grafici
    ├── profiles/        → profili di installazione
    └── sites/           → configurazioni dei siti (es. settings.php)
```

Struttura delle directory

```
/drupal
│
├── .git/                → repository Git (controllo di versione)
├── .gitignore           → regole per escludere file dal versionamento
├── .editorconfig        → standard di formattazione del codice
├── composer.json        → definisce le dipendenze del progetto
├── composer.lock        → blocca le versioni installate
├── vendor/              → librerie PHP installate da Composer
│
├── config/              → configurazioni esportate di Drupal
├── private/             → file e dati non accessibili pubblicamente
│
└── web/                 → root pubblica del sito
    ├── core/            → codice sorgente di Drupal
    ├── modules/         → moduli contrib e personalizzati
    ├── themes/          → temi grafici
    ├── profiles/        → profili di installazione
    └── sites/           → configurazioni dei siti (es. settings.php)
```

Drupal e Symfony

Un legame strategico

- A partire da Drupal 8, il CMS adotta componenti del framework PHP Symfony
- L’obiettivo: rendere Drupal più moderno, modulare e orientato agli standard del mondo PHP
- Symfony fornisce la base infrastrutturale, Drupal la logica applicativa e i contenuti


Drupal™


COMPOSER

Drupal 7: un’apparente semplicità

- Installare un modulo era facile: bastava un link o un file “.zip”
- Con un clic su “Installa”, il modulo veniva aggiunto senza difficoltà
- Sembrava una rivoluzione, ma presto emersero limiti importanti

La svolta: l’introduzione di Composer

- Per risolvere i problemi di dipendenza nasce **Composer**
- È un **gestore di pacchetti PHP open source** che installa moduli e librerie in modo automatico
- Garantisce coerenza tra le versioni e semplifica l’aggiornamento del sistema

L’installazione con Composer

- Si passa da un’interfaccia grafica alla **linea di comando**.
- Inizialmente può sembrare più complesso, ma offre grandi vantaggi:
  - Installa automaticamente tutte le dipendenze
  - Evita conflitti tra moduli e versioni
  - Aggiorna e gestisce l’intero progetto in modo centralizzato

Il Manifesto del Progetto
composer.json

- Descrive **le dipendenze** del progetto (moduli, librerie, versioni)
- Contiene informazioni su:
  - pacchetti richiesti (`require`, `require-dev`)
  - nome, versione e licenza del progetto
  - script di installazione o build
- In pratica:
  - *“Quello che voglio installare”*

Lo Stato del Sistema
composer.lock

- Viene **generato automaticamente** da Composer
- Elenca **le versioni esatte** dei pacchetti installati
- Garantisce che ogni ambiente (dev, test, produzione) usi **le stesse versioni**.
- Non si modifica manualmente: va **committato nel repository**.
- In pratica:
  - *“Quello che è stato effettivamente installato”*

PSR-4

PSR-4 è **lo standard PHP per l'autoloading delle classi**.

ovvero un modo per caricare automaticamente le classi quando vengono utilizzate.

PSR-4

- Approccio autoloading delle classi
- Autoloading basato sui percorsi dei file

- miomodulo
  - **/src**
    - **/Plugin**
      - **/Block**
        - BloccoCustom.php

PSR-4

```php
// Namespace dichiarato nel file
namespace Drupal\mio_modulo\Controller;

class HelloController {
  public function content() {
    return ['#markup' => 'Ciao dal mio
modulo!'];
  }
}
```

PSR-4

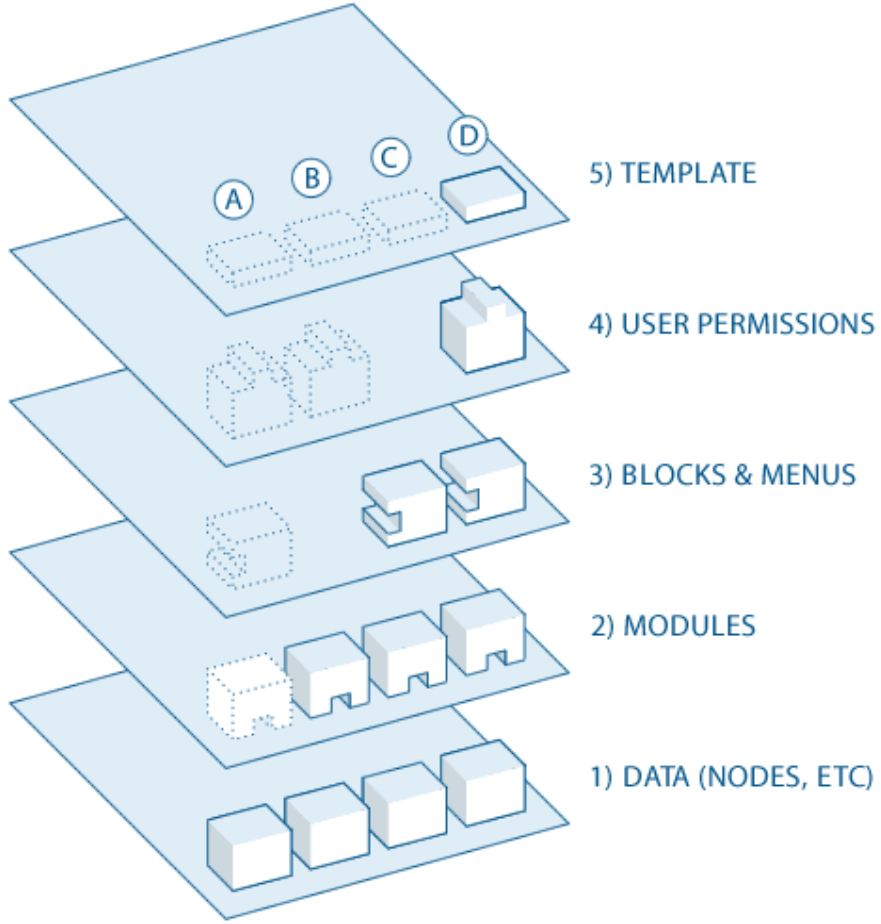
Posizione del file:

`/modules/custom/mio_modulo/src/Controller/HelloController.php`

Il namespace `Drupal\mio_modulo\Controller`

mappa la directory `modules/custom/mio_modulo/src/Controller`

Drupal flow


A
B
C
D
5) TEMPLATE
4) USER PERMISSIONS
3) BLOCKS & MENUS
2) MODULES
1) DATA (NODES, ETC)

DRUSH La CLI di Drupal

- Drush (Drupal Shell) è la riga di comando ufficiale
- Permette di gestire il sito senza usare l’interfaccia grafica.

DRUSH La CLI di Drupal

- È **più veloce** per operazioni ripetitive.
- Automatizza installazioni, e cache.
- Indispensabile per chi sviluppa moduli e temi
- Scriptabile anche con composer

Devel Il modulo per gli sviluppatori

- **Devel** è un modulo pensato per **chi sviluppa in Drupal**.
- Offre strumenti utili per il **debug**, la **profilazione** e la **visualizzazione dei dati** durante lo sviluppo.
- Aiuta a “guardare dentro” Drupal
- Essenziale per ogni sviluppatore in fase di test e debugging

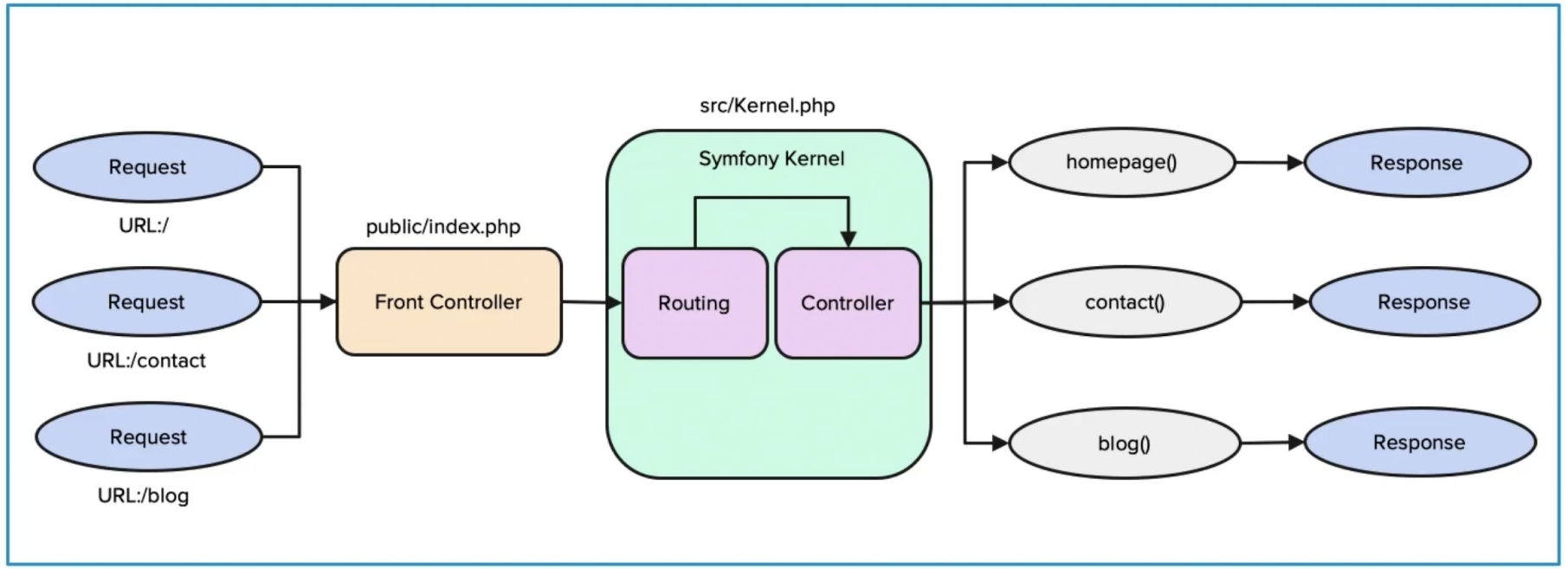
2 Moduli personalizzati: logiche e form

- Struttura base di un modulo: file .info.yml, .routing.yml, .services.yml
- Creazione di una pagina custom tramite routing e controller
- Creazione di un blocco custom posizionabile da interfaccia di Drupal
- Creazione filtro
- Hook system
- Plugin
- Esercitazione: Creare un modulo custom con una sua route che renderizza

Page Call Process

- Processo di rendering HTML
- Response formats
  - HTML, JSON
- Front Controller

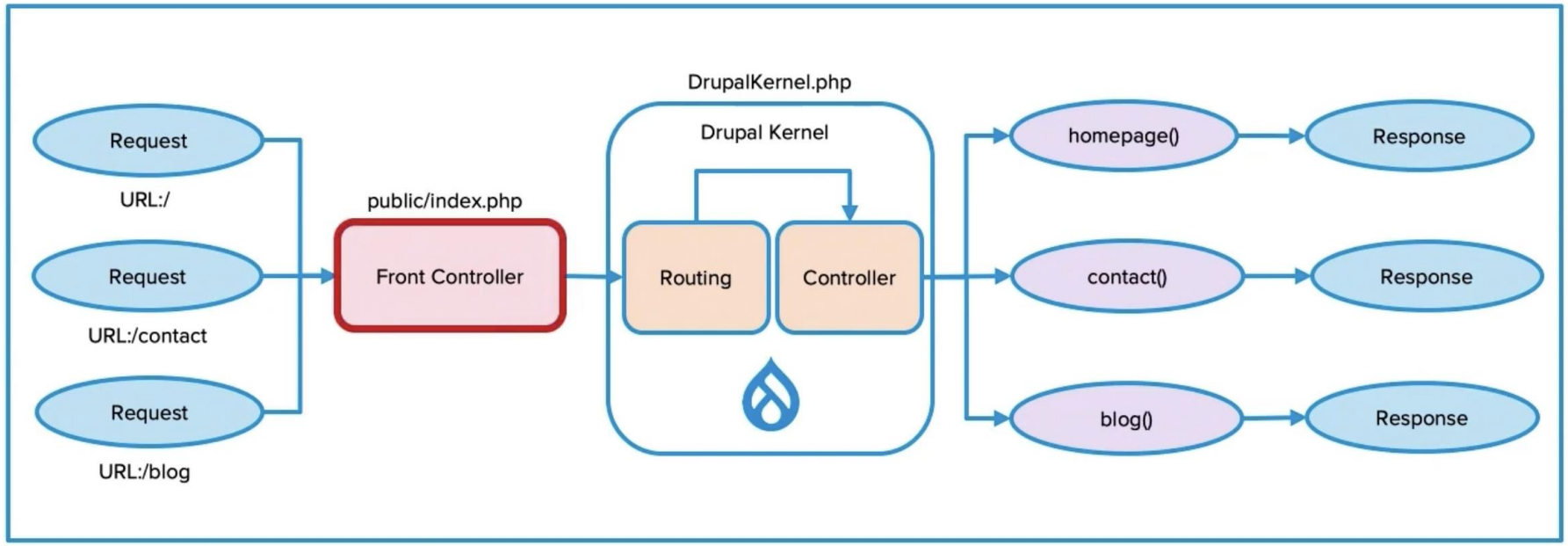
Symphony Flow



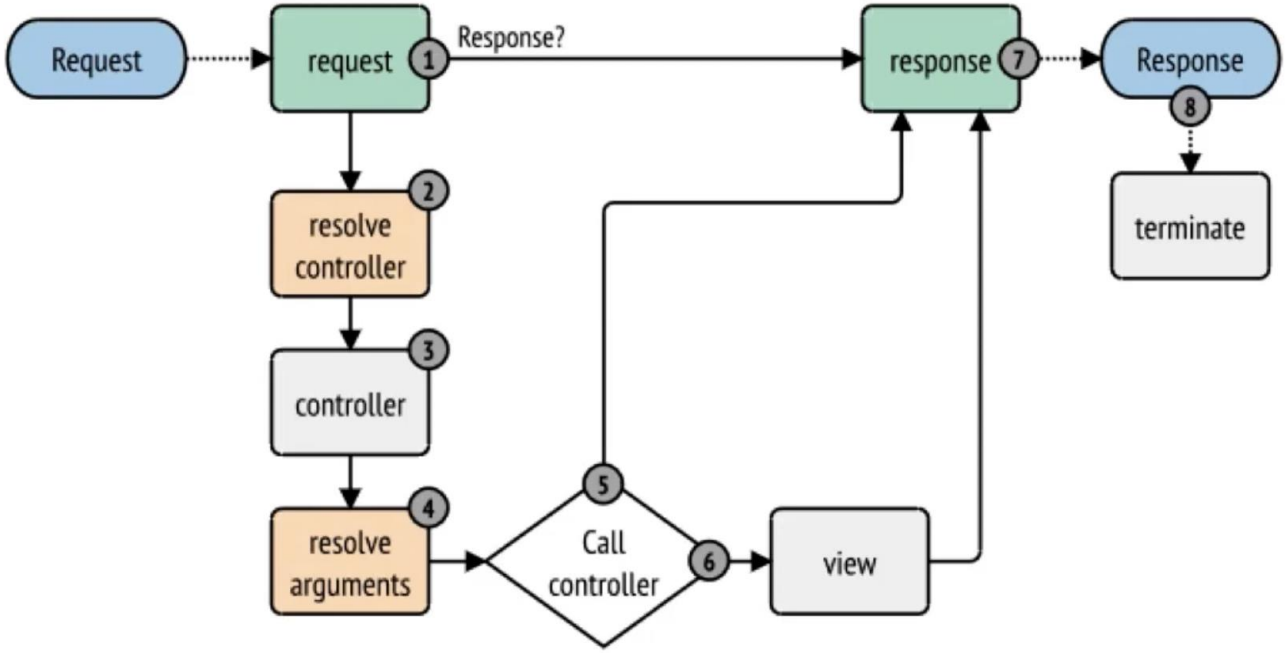
Perché capire il processo di chiamata di una pagina

- Un **modulo** è una parte fondamentale di Drupal
- Drupal include **funzionalità di sicurezza integrate**
- Capire il flusso della pagina rende il **debug del codice dei moduli molto più semplice**

Hello Front Controller



The Drupal Kernel Flow



Oggetti Request & Response

- **Symfony Request Object**

  → **Rappresentazione a oggetti** della richiesta HTTP

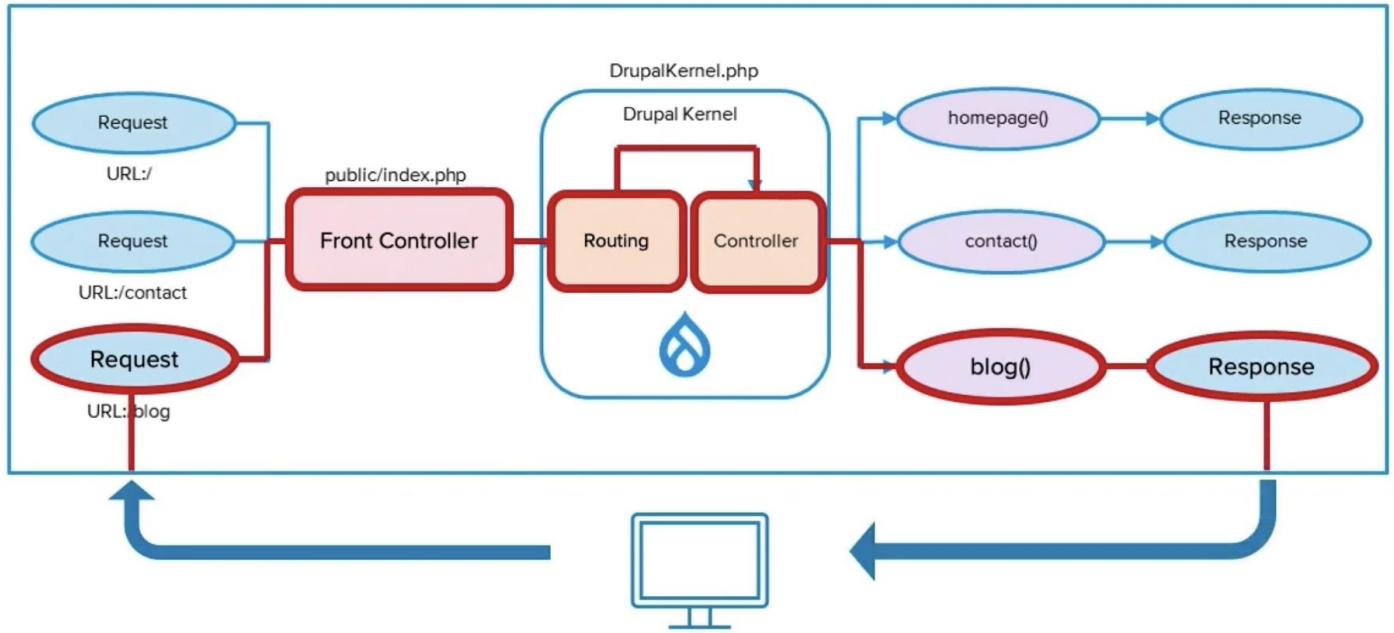
  → Contiene dati come URL, parametri, cookie, header, metodo (GET/POST)

- **Symfony Response Object**

  → **Rappresentazione a oggetti** della risposta HTTP

  → Contiene lo **stato HTTP**, gli **header** e il **contenuto HTML** restituito al browser

The Drupal Application Flow



Il flusso Request–Response in Drupal

- Un **client (browser)** invia una **richiesta HTTP**
- Ogni richiesta esegue **lo stesso file di ingresso**
  → **il cosiddetto “front controller”** (`index.php`)
- Il **front controller** inizializza (*bootstrappa*) Drupal
- **Route** e **Controller** generano l’oggetto **Response**
- Drupal trasforma l’oggetto **Response** in una **risposta HTTP**
  → **che viene inviata di nuovo al browser del client**

3 Theming: personalizzare il frontend

- Struttura di un tema: file .info.yml, librerie e Twig
- Twig template
- Override dei template e funzioni preprocess
- Creazione di un subtema basato su un tema esistente (es. Bootstrap)
- Gestione di CSS/JS con le librerie (.libraries.yml)
- Inserimento di variabili dinamiche nei template

4 Integrazione moduli e tema twig

- Collegare un modulo custom con il tema
- Aggiungere template personalizzati per i dati del modulo
- Gestione della cache e rendering di contenuti dinamici