



# Modern PHP Development

Sebastian Bergmann

# I can help you with

- adopting PHPUnit,
- optimizing its use,
- refining development processes, and
- writing more testable code



# Modern

Latin: *modernus*, from *modo* ("just now") + *-rnus*

Old French: *moderne*

Middle English: "pertaining to present times"

Contemporary: "recency", "up-to-dateness"

# Modern Practices

Methods, techniques, or approaches that are current, up to date, and reflect the present state of knowledge or standards in a particular field.

# Best Practices

Methods, techniques, or approaches that are current, up to date, and reflect the present state of knowledge or standards in a particular field.

# Agenda

Documentation

Guidelines

Testing

AI

# Agenda

Documentation

Guidelines

Testing

AI

# Documentation

- Software Architecture and Design
- Architecture Decision Records
- Technical Debt Records
- Quality Goals
- Coding Guidelines (beyond formatting)

# Software Architecture (Example: arc42)

- Introduction & Goals
- Constraints
- Context & Scope
- Solution Strategy
- Building Block View
- Runtime View
- Deployment View
- Crosscutting Concepts
- Architecture Decisions
- Quality
- Technical Debt
- Glossary

Project

Junie Ultimate

example ~/example/example

- > bin
- > config
- > custom
- > docs
- > files
- > public
- > src
- > var
- > vendor
  - ≡ .env
  - ≡ .env.dev
  - ≡ .env.local
- ≡ .gitignore
- ≡ .htaccess
- composer.json
- composer.lock
- ≡ symfony.lock

External Libraries

Scratches and Consoles

Search Everywhere Double ↑

Go to File ⌘O

Recent Files ⌘E

Navigation Bar ⌘↑

Drop files here to open them

← Document Project Architecture Using arc42 Format

✓ Start Again Rollback All

≡ Document this project's architecture in arc42 format.

ⓘ Plan

1. Review repository to identify architectural anchors: entrypoint, kernel, DI config, plugin locations, environments, and deployment artifacts.
  - ✓ - Inspect composer.json for packages and autoload.
  - ✓ - Locate and open public/index.php.
  - ! - Locate and open src/Kernel.php.
  - ✓ - Review config files: config/bundles.php, config/services.yaml, config/packages/\*.
2. Draft arc42 structure tailored to this project.
  - Prepare sections 1-12 with concise, actionable content (Intro/Goals, Constraints, Context, Solution)
  - ✓ Strategy, Building Blocks, Runtime, Deployment, Crosscutting, Decisions, Quality, Risks/Debt, Glossary).
  - Reflect Shopware-specific aspects (plugins, storefront/admin separation, DI/events, data, caches).
  - ✓ 3. Create documentation file in the repository.
    - Add docs/architecture/arc42.md (or fallback to docs/arc42.md if necessary).
    - Include placeholders for diagrams (PlantUML/Mermaid) and links to config/ and custom/plugins/.
  - ✓ 4. Validate and adjust.
    - Re-check paths and references to ensure they match this repo (e.g., exact directories and versions).
    - If directories are missing, adjust file placement and update links accordingly.

Type your task here, press ↲

example ~/example/example

- > bin
- > config
- > custom
- > docs
- > files
- > public
- > src
- > var
- > vendor
- ≡ .env
- ≡ .env.dev
- ≡ .env.local
- ∅ .gitignore
- ≡ .htaccess
- ∅ composer.json
- ∅ composer.lock
- ≡ symfony.lock

> External Libraries

> Scratches and Consoles

Search Everywhere Double ▲

Go to File ⌘O

Recent Files ⌘E

Navigation Bar ⌘↑

Drop files here to open them

← Document Project Architecture Using arc42 Format

✓ Start Again Rollback All

✓ 5. Deliver summary and next steps.

ⓘ Search in project "composer.json"

The search for "composer.json" in the project found one matching file.

ⓘ Search in project "Kernel.php"

The search for "Kernel.php" found multiple related cache files and one reference in the services configuration file.

ⓘ Open composer.json

The composer.json file was opened successfully, revealing its contents and dependencies.

ⓘ Search in project "src"

The search for the term "src" returned over 100 results, indicating many occurrences across the project files.

ⓘ Update on task progress

We need to produce arc42 documentation for this Shopware 6 (Symfony-based) project. Initial quick peek at composer.json confirms Shopware core with plugins path. Next, we'll inventory key files (kernel, entrypoint, configs) to tailor the doc, then create a docs/arc42.md with a filled template aligned to this repo.

ⓘ Search in project "public/index.php"

The search found one file named "public/index.php" and one occurrence of the term in a lock file.

Type your task here, press ↲

example ~/example/example

- > bin
- > config
- > custom
- > docs
- > files
- > public
- > src
- > var
- > vendor
  - ≡ .env
  - ≡ .env.dev
  - ≡ .env.local
- ∅ .gitignore
- ≡ .htaccess
- ∅ composer.json
- ∅ composer.lock
- ≡ symfony.lock

> External Libraries

> Scratches and Consoles

Search Everywhere Double ↑

Go to File ⌘O

Recent Files ⌘E

Navigation Bar ⌘↑

Drop files here to open them

← Document Project Architecture Using arc42 Format



Start Again

Rollback All

Show Plan

ⓘ Open index.php

The specified PHP file was opened successfully, revealing its full content of 49 lines.

ⓘ Search in project "config/bundles.php"

The search located the file "config/bundles.php" within the project and found related metadata in the cache directory.

ⓘ Open services.yaml

The services configuration file was opened, revealing default service settings and parameters for the application.

ⓘ Update on task progress

Initialized [CODE] mode to create arc42 architecture docs. Confirmed Shopware/Symfony app via composer.json and public/index.php. Next: inspect key config/kernel files to tailor documentation, then create docs/architecture/arc42.md with filled sections.

ⓘ Open bundles.php

The bundles configuration file was opened successfully, displaying a list of enabled bundles for different environments.

ⓘ Search in project "config/packages"

The search for "config/packages" in the project found one symbol and three text occurrences related to configuration files and cached resources.



Type your task here, press ↲

Project

- example ~/example/example
  - bin
  - config
  - custom
  - docs
  - files
  - public
  - src
  - var
  - vendor
    - .env
    - .env.dev
    - .env.local
  - .gitignore
  - .htaccess
  - composer.json
  - composer.lock
  - symfony.lock
- External Libraries
- Scratches and Consoles

Search Everywhere Double ⌂

Go to File ⌂⌘O

Recent Files ⌂⌘E

Navigation Bar ⌂↑

Drop files here to open them

Junie Ultimate

← Document Project Architecture Using arc42 Format

✓ Start Again Rollback All Show Plan

⌚ Search in project "config/packages" The search for "config/packages" in the project found one symbol and three text occurrences related to configuration files and cached resources.

⌚ Open file The file requested to open does not exist in the current directory, but several possible candidate files were identified in the cache directory.

⌚ Search in project "docs" The search for the term "docs" returned over 100 results across multiple files, indicating widespread occurrences in the project.

⌚ Update on task progress Proceeding in [CODE] mode. We identified it's a Shopware 6 (Symfony) app. Key anchors located: public/index.php, config/bundles.php, services.yaml. No custom Kernel; KernelFactory used. Next: create arc42 markdown under docs/architecture/ with tailored content and links to this repo's structure.

⌚ Search in project "custom/plugins" The term "custom/plugins" was found in six locations across cache files, the composer.json, and symfony.lock files within the project.

>Edit arc42.md +321 An arc42 architecture documentation file for a Shopware Symfony-based application was created.

Type your task here, press ↲

example ~/example/example

- > bin
- > config
- > custom
- > docs
- > files
- > public
- > src
- > var
- > vendor
  - ≡ .env
  - ≡ .env.dev
  - ≡ .env.local
- ∅ .gitignore
- ≡ .htaccess
- ∅ composer.json
- ∅ composer.lock
- ≡ symfony.lock

> External Libraries

> Scratches and Consoles

Search Everywhere Double ↑

Go to File ⌘O

Recent Files ⌘E

Navigation Bar ⌘↑

Drop files here to open them

← Document Project Architecture Using arc42 Format



Start Again

Rollback All

Show Plan

The term "custom/plugins" was found in six locations across cache files, the composer.json, and symfony.lock files within the project.

>Edit arc42.md +321

An arc42 architecture documentation file for a Shopware 6 Symfony-based application was created.

Done

New Files

arc42.md +321

### What I delivered

- Added a complete arc42 architecture document tailored to this Shopware 6 (Symfony-based) project.
- Location: `docs/architecture/arc42.md`
- It includes sections 1-12 (Intro/Goals, Constraints, Context, Solution Strategy, Building Blocks, Runtime, Deployment, Crosscutting Concepts, Decisions, Quality, Risks/Debt, Glossary), plus an appendix with repo-specific paths.
- Referenced concrete files in this repo (e.g., `public/index.php`, `config/bundles.php`, `config/services.yaml`, `composer.json`).

[Click to expand](#)

[Rollback](#)

[Start New Task](#)

Type your task here, press ↲

# arc42 Architecture Documentation

Project: Shopware 6 application (Symfony-based) Date: 2025-10-27 15:16

Note: This document follows the arc42 structure (v8) and is tailored to this repository. It references concrete files and directories that exist in this project.

## 1. Introduction and Goals

- Business goals
  - Provide a production-grade e-commerce platform based on Shopware 6 with a classic Storefront and an Administration (back office) UI.
  - Enable extensibility through plugins in `custom/plugins` and optional static plugins in `custom/static-plugins`.
  - Support search via Elasticsearch/OpenSearch and CDN strategies for media delivery.
- Stakeholders
  - Merchant/business owner: cares about stability, SEO, conversion, content.
  - Shop admin/content editor: uses Administration to manage catalog, orders, customers.
  - Developers/DevOps: implement custom features as plugins, operate and deploy the system.
  - Support/Operations: monitor logs, errors, and performance.
- Quality goals
  - Reliability and data integrity for checkout and order processing.
  - Performance (fast storefront, responsive admin, efficient search).
  - Extensibility and upgradeability across Shopware releases.
  - Security and compliance (authentication, authorization, updates).
- Scope
  - This repository contains the application scaffold for Shopware 6, including configuration and integration points for custom code under `src/` and `custom/plugins`.

References in repo:

- Composer manifest: `composer.json` (Shopware packages and plugin paths)

## References in repo:

- Composer manifest: `composer.json` (Shopware packages and plugin paths)
- Front controller: `public/index.php`
- Bundles (Symfony/Shopware modules): `config/bundles.php`
- Service wiring: `config/services.yaml`

## 2. Constraints

- Technological
  - Shopware core version: `shopware/core: v6.7.3.1` (see `composer.json` ).
  - Symfony Flex used for recipes: `symfony/flex ~2` .
  - Bundles enabled (see `config/bundles.php` ):
    - Shopware modules: `Framework` , `System` , `Content` , `Checkout` , `Maintenance` , `Profiling` , `Service` , `Administration` , `Elasticsearch` , `Storefront` .
    - Symfony bundles: `FrameworkBundle` , `TwigBundle` , `MonologBundle` , `DebugBundle` (dev only).
    - Additional: `PentatrionViteBundle` .
  - Autoload namespace: `App\` → `src/` (PSR-4).
- Organizational
  - Customizations should be encapsulated as plugins (in `custom/plugins` ) or as application code under `src/` with Symfony DI.
  - Infrastructure and deployment are expected to follow standard PHP/Symfony hosting patterns.
- Environmental and external systems
  - The app can integrate with Elasticsearch/OpenSearch (parameter `env(OPENSEARCH_URL)` present in `config/services.yaml` ).
  - CDN strategy can be configured (parameters `default_cdn_strategy` , `shopware.cdn.strategy` ).
  - Installer is used unless `install.lock` exists (see `public/index.php` ).

## 3. Context and Scope

- Business context

## 3. Context and Scope

- Business context
  - External actors: end customers (browsing and purchasing), shop administrators, payment service providers (configured via plugins), search services (Elasticsearch/OpenSearch), media/CDN.
- Technical context
  - Entry point: `public/index.php` → Shopware Installer (first run) or `KernelFactory::create(...)` for the app kernel.
  - HTTP requests handled by Symfony/Shopware routing → controllers/services in Shopware and in custom plugins.
  - Data storage: Shopware typically uses MySQL/MariaDB; configuration not shown here but part of standard Shopware deployments.
  - Search: Elasticsearch/OpenSearch optional integration via `shopware/elasticsearch`.
- Out of scope
  - Payment, shipping, and ERP integrations are provided by plugins and are not described in detail here.

## 4. Solution Strategy

- Use Shopware 6 as the application framework (built on Symfony) to leverage its domain modules (Content, Checkout, System) and plugin/event system.
- Frontend channels
  - Storefront (server-rendered via Twig and Shopware Storefront bundle).
  - Administration (SPA built on Vue, compiled assets available under `public/bundles/...`).
- Extensibility via plugins in `custom/plugins` using the Shopware plugin API; repositories in `composer.json` allow local, symlinked plugin development.
- Dependency injection
  - Symfony DI autowiring and autoconfiguration enabled in `config/services.yaml` for the `App\` namespace.
- Observability and operations
  - Logging via Monolog; caching and other infrastructure configured through standard Shopware/Symfony settings.
- Search strategy
  - Optionally enable Elasticsearch through `shopware/elasticsearch`; endpoint configurable via environment.

## 5. Building Block View

- Level 1 (system decomposition)
  - Storefront ( `Shopware\Storefront\Storefront` )
    - Controllers, pages, Twig templates, checkout UX.
  - Administration ( `Shopware\Administration\Administration` )
    - Back office SPA for catalog, orders, configurations.
  - Core ( `Shopware\Core` )
    - Submodules: `Framework`, `System`, `Content`, `Checkout`, `Maintenance`, `Profiling`, `Service`.
  - Integration/Search ( `Shopware\Elasticsearch\Elasticsearch` )
    - Search synchronization and queries (optional).
  - Application code ( `App\` in `src/` )
    - Custom services, controllers, subscribers, command-line tasks.
  - Plugins ( `custom/plugins` )
    - Feature encapsulation via Shopware's plugin mechanism.
- Level 2 (inside App/ and Plugins)
  - `src/` (PSR-4 `App\`) is auto-registered by `config/services.yaml` (excludes `DependencyInjection`, `Entity`, and `Kernel.php`).
  - Event subscribers, handlers, and services are auto-wired; explicit service definitions can be added in `config/services.yaml` or module-specific files.
- Example diagram (Mermaid)

```
flowchart TD
  A[HTTP Client] --> B[public/index.php]
  B -->|Installer (if no install.lock)| I[InstallerKernel]
  B -->|Else| K[Shopware Kernel via KernelFactory]
  K --> R[Routing]
  R --> C[Controller/Handler]
  C --> S[Services (DI)]
  S --> D[(DB)]
  S --> ES[(Elasticsearch/OpenSearch)]
  S --> M[(Media/CDN)]
```

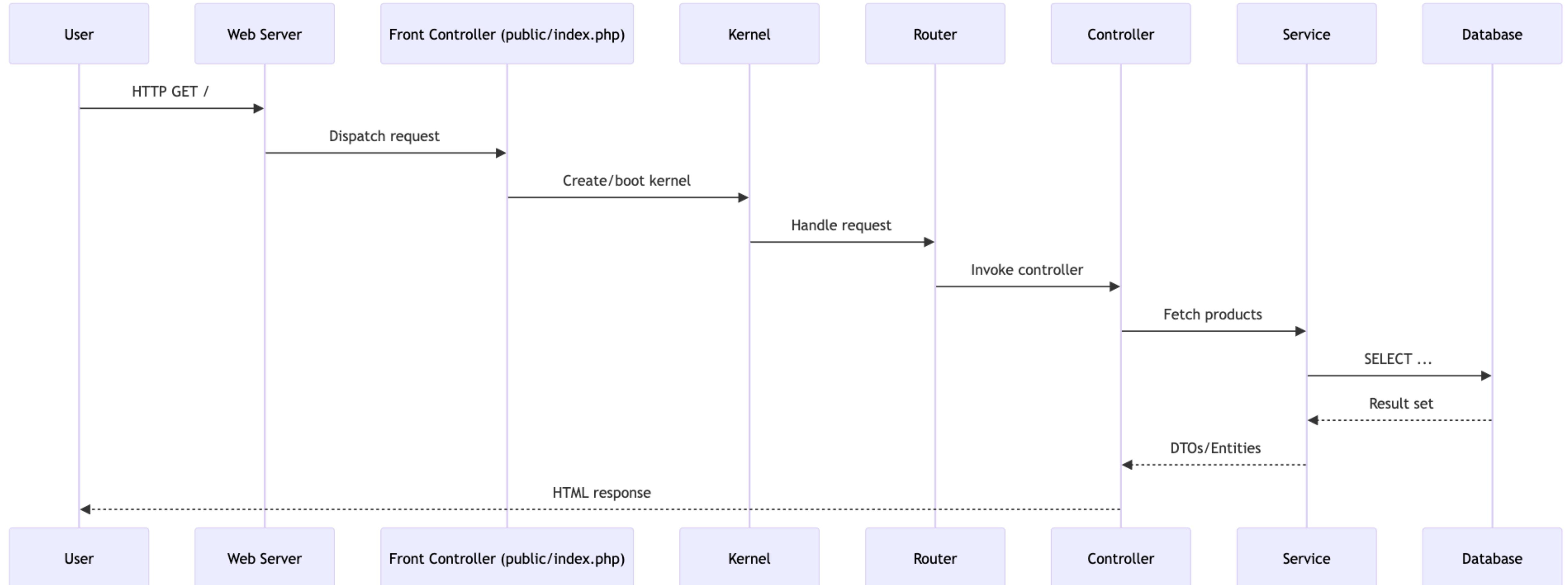
## 6. Runtime View

- Typical storefront request
  - i. Request hits web server and `public/index.php` .
  - ii. If not installed, user is redirected to `/installer` ( `InstallerKernel` ).
  - iii. Otherwise, Shopware kernel is created ( `KernelFactory::create(...)` ).
  - iv. Symfony routing maps to a controller (Shopware core, Storefront, or custom plugin).
  - v. Controller uses services (DI) to query data, execute business logic, and render a Twig response.
- Admin runtime
  - Administration SPA consumes Shopware API endpoints; authentication and permissions are enforced by core.
- Background tasks
  - Shopware scheduled tasks and message handling (not explicitly configured in this repo, but standard in Shopware) can process indexing, imports, or emails.

### Sequence overview (Mermaid)

```
sequenceDiagram
    participant U as User
    participant W as Web Server
    participant F as Front Controller (public/index.php)
    participant K as Kernel
    participant R as Router
    participant C as Controller
    participant S as Service
    participant DB as Database

    U->>W: HTTP GET /
    W->>F: Dispatch request
    F->>K: Create/boot kernel
    K->>R: Handle request
    R->>C: Invoke controller
    C->>S: Fetch products
    S->>DB: SELECT ...
    DB-->>S: Result set
    S-->>C: DT0s/Entities
    C-->>U: HTML response
```



## 7. Deployment View

- Packaging & artifacts
  - Composer-managed PHP application; vendor dependencies in `vendor/` .
  - Frontend assets for Administration and Storefront under `public/` .
- Execution environment
  - Typical: Nginx/Apache + PHP-FPM (or Symfony CLI for local dev).
  - Entry point: `public/index.php` .
- Filesystem and state
  - Caches and generated files in `var/` .
  - Media and documents in `files/` and `public/` (depending on config and CDN strategy).
- External services
  - Database (MySQL/MariaDB).
  - Elasticsearch/OpenSearch (optional, `env(OPENSEARCH_URL)` ).
  - CDN or object storage per `shopware.cdn.strategy` .
- Deployment diagram (Mermaid)

```
deploymentDiagram
node Browser
node WebServer
node PHPFPM
node DB[(MySQL/MariaDB)]
node ES[(Elasticsearch/OpenSearch)]
node CDN[(CDN/Media Storage)]

Browser --> WebServer
WebServer --> PHPFPM
PHPFPM --> DB
PHPFPM --> ES
PHPFPM --> CDN
```

## 8. Cross-cutting Concepts

- Configuration management
  - Environment variables ( `.env` , `.env.local.php` ) and Symfony config in `config/` .
  - Installer controls first-time setup and generates `install.lock` .
- Security
  - Authentication/authorization provided by Shopware core modules; administration requires appropriate roles.
- Internationalization (i18n) and localization (l10n)
  - Shopware supports multiple languages, currencies, and sales channels.
- Logging and monitoring
  - Monolog bundle for logs; logs typically under `var/log/` .
- Error handling
  - Standard Symfony exception handling; error pages customizable in Storefront.
- Build/Assets
  - `PentatrichtViteBundle` present; Shopware toolchain compiles Administration/Storefront assets, distributed under `public/bundles/...` .
- Extensibility patterns
  - Event subscribers, decoration, DI compiler passes, and plugin life-cycle hooks.

## 9. Architecture Decisions (ADR Summary)

- ADR-001: Use Shopware 6 with Storefront and Administration
  - Status: accepted (see `config/bundles.php` ).
- ADR-002: Enable optional Elasticsearch/OpenSearch integration
  - Status: accepted (bundle present; `OPENSEARCH_URL` parameter).
- ADR-003: Organize customizations as plugins under `custom/plugins`
  - Status: accepted (composer path repositories with symlinks).
- ADR-004: Use Symfony DI autowiring/autoconfiguration for `App\\` code
  - Status: accepted (see `config/services.yaml` ).
- ADR-005: Use Installer for first run and runtime KernelFactory

## 10. Quality Requirements

- Performance
  - Caching and HTTP caching via Storefront; consider search indexing for catalog size.
- Security
  - Keep Shopware core and plugins up to date; configure HTTPS and proper headers at the web server.
- Maintainability
  - Encapsulate features in plugins; avoid core modifications; use events and decoration.
- Testability
  - Use Shopware testing framework (PHPUnit) and integration tests for plugins.
- Observability
  - Centralized logging; consider APM/metrics.

## 11. Risks and Technical Debt

- Plugin compatibility
  - Risk of breaking changes across Shopware upgrades; ensure plugins declare proper constraints.
- Search dependency
  - When enabling Elasticsearch/OpenSearch, ensure cluster availability; implement graceful degradation.
- Asset pipeline
  - Administration/Storefront assets must be rebuilt in CI/CD when customizing.
- Configuration drift
  - Keep environment-specific configs consistent across environments; document critical env vars.
- Repository hygiene
  - Ensure transient directories like `var/cache` are not persisted between deployments unless intended.

## 12. Glossary

- Administration/Storefront assets must be rebuilt in CI/CD when customizing.
- Configuration drift
  - Keep environment-specific configs consistent across environments; document critical env vars.
- Repository hygiene
  - Ensure transient directories like `var/cache` are not persisted between deployments unless intended.

## 12. Glossary

- Shopware 6: PHP e-commerce platform built on Symfony.
- Storefront: The customer-facing web frontend delivered by the `shopware/storefront` package.
- Administration: The back-office SPA for managing the shop.
- Plugin: A packaged extension that adds/overrides functionality; lives under `custom/plugins`.
- Kernel: The Symfony/Shopware application kernel created via `KernelFactory`.
- Sales Channel: Logical channel in Shopware that represents a storefront or API channel.

## Appendix

- Key files and directories in this repository
  - `public/index.php` – front controller and installer bootstrapping.
  - `config/bundles.php` – enabled Symfony/Shopware bundles (modules).
  - `config/services.yaml` – DI configuration for `App\` namespace and parameters (e.g., CDN strategy).
  - `composer.json` – packages, autoload, and plugin path repositories.
  - `custom/plugins` – local plugin development path (symlinked via Composer paths).
  - `src/` – application namespace for custom services/controllers.
  - `var/` – caches and generated artifacts.
  - `public/` – web root; contains compiled Administration/Storefront assets.
- Editing this document
  - Keep this file up to date with major changes to bundles, external integrations, or plugin architecture.

# Architecture Decision Records

- Status
- Context
- Decision
- Consequences

# Technical Debt Records

- Context
- Impact
- Symptoms
- Risks
- Proposed solution
- Cost of delay
- Effort to resolve
- Dependencies

# Agenda

Documentation

Guidelines

Testing

AI

## # Guidelines

### ## Development Process

- \* We use test-driven development in this project
- \* Always run existing tests before making any code changes
- \* Execute the full test suite after implementing new features
- \* Always verify test results before completing tasks
- \* All tests must pass before changes are considered complete
- \* Code coverage should be maintained or improved

## # Guidelines

### ## Development Process

- \* We use test-driven development in this project
- \* A human developer always writes the test code
- \* An AI coding agent may write production code
- \* An AI coding agent must not edit test code
- \* An AI coding agent must not edit configuration files for tools
- \* Always run existing tests before making any code changes
- \* Execute the full test suite after implementing new features
- \* If tests fail, fix them before presenting the final solution
- \* Always verify test results before completing tasks
- \* All tests must pass before changes are considered complete
- \* Code coverage should be maintained or improved

## ## Development Tools

- \* Use `./tools/composer dump-autoload` to generate an autoloader
- \* Use `./tools/php-cs-fixer fix` to format the code
- \* Use `./tools/phpstan` to perform static analysis
- \* Use `./tools/phpunit` to run the tests
- \* Use `./tools/infection` to perform mutation testing

## ## Programming Style

- \* Avoid inheritance
- \* A class must either be declared `abstract` or `final`
- \* Favour immutability (`readonly`) over mutability
- \* Favour lists over arrays
- \* In type annotations, use `list<type>` instead of `type[]`
- \* Use British English for names, comments, etc.

# Agenda

Documentation

Guidelines

Testing

AI

# Testing

- Static Testing
- Dynamic Testing

```
<?php declare(strict_types=1);  
use PHPUnit\Framework\TestCase;  
  
final class BankAccountTest extends TestCase  
{  
  
}  
}
```

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class BankAccountTest extends TestCase
{
    public function testBalanceIsInitiallyZero(): void
    {
        $bankAccount = new BankAccount;

        $this->assertSame(0, $bankAccount->balanceInCents());
    }
}
```

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class BankAccountTest extends TestCase
{
    public function testBalanceIsInitiallyZero(): void
    {
        $bankAccount = new BankAccount;

        $this->assertSame(0, $bankAccount->balanceInCents());
    }

    public function testMoneyCanBeDeposited(): void
    {
        $bankAccount = new BankAccount;

        $bankAccount->deposit(1234);

        $this->assertSame(1234, $bankAccount->balanceInCents());
    }
}
```

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class BankAccountTest extends TestCase
{
    public function testBalanceIsInitiallyZero(): void
    {
        $bankAccount = new BankAccount;

        $this->assertSame(0, $bankAccount->balanceInCents());
    }

    public function testMoneyCanBeDeposited(): void
    {
        $bankAccount = new BankAccount;

        $bankAccount->deposit(1234);

        $this->assertSame(1234, $bankAccount->balanceInCents());
    }

    public function testMoneyCanBeWithdrawn(): void
    {
        $bankAccount = new BankAccount;

        $bankAccount->withdraw(1234);

        $this->assertSame(-1234, $bankAccount->balanceInCents());
    }
}
```

```
1  <?php
2  final class BankAccount {
3      private array $transactions = [];
4
5      public function balanceInCents() {
6          $balanceInCents = 0;
7
8          foreach ($this->transactions as $transaction) {
9              $balanceInCents += $transaction['amountInCents'];
10         }
11
12         return $balanceInCents;
13     }
14
15     public function deposit(int $amountInCents)
16     {
17         $this->transactions[] = [
18             'timestamp'    => new DateTimeImmutable('now'),
19             'amountInCents' => $amountInCents,
20             'description'  => 'Deposit',
21         ];
22     }
23
24     public function withdraw(int $amountInCents)
25     {
26         $this->transactions[] = [
27             'timestamp'    => new DateTimeImmutable('now'),
28             'amountInCents' => -1 * $amountInCents,
29             'description'  => 'Withdrawal',
30         ];
31     }
32 }
```

```
<?php
final class BankAccount {
    private $transactions;

    public function balanceInCents() {
        $balanceInCents = 0;

        foreach ($this->transactions as $transaction) {
            $balanceInCents += $transaction['amountInCents'];
        }

        return $balanceInCents;
    }

    public function deposit($amountInCents) {
        $this->transactions[] = [
            'timestamp'      => new DateTimeImmutable('now'),
            'amountInCents'  => $amountInCents,
            'description'   => 'Deposit',
        ];
    }

    public function withdraw($amountInCents) {
        $this->transactions[] = [
            'timestamp'      => new DateTimeImmutable('now'),
            'amountInCents'  => -1 * $amountInCents,
            'description'   => 'Withdrawal',
        ];
    }
}
```

```
<?php declare(strict_types=1);
final class BankAccount {
    private array $transactions = [];

    public function balanceInCents(): int {
        $balanceInCents = 0;

        foreach ($this->transactions as $transaction) {
            $balanceInCents += $transaction['amountInCents'];
        }

        return $balanceInCents;
    }

    public function deposit(int $amountInCents): void {
        $this->transactions[] = [
            'timestamp'      => new DateTimeImmutable('now'),
            'amountInCents'  => $amountInCents,
            'description'   => 'Deposit',
        ];
    }

    public function withdraw(int $amountInCents): void {
        $this->transactions[] = [
            'timestamp'      => new DateTimeImmutable('now'),
            'amountInCents'  => -1 * $amountInCents,
            'description'   => 'Withdrawal',
        ];
    }
}
```

```
<?php declare(strict_types=1);
final class BankAccount {
    private array $transactions = [];

    public function balanceInCents(): int {
        $balanceInCents = 0;

        foreach ($this->transactions as $transaction) {
            $balanceInCents += $transaction['amountInCents'];
        }

        return $balanceInCents;
    }
    /**
     * @param positive-int $amountInCents
     */
    public function deposit(int $amountInCents): void {
        $this->transactions[] = [
            'timestamp'      => new DateTimeImmutable('now'),
            'amountInCents'  => $amountInCents,
            'description'   => 'Deposit',
        ];
    }
    /**
     * @param positive-int $amountInCents
     */
    public function withdraw(int $amountInCents): void {
        $this->transactions[] = [
            'timestamp'      => new DateTimeImmutable('now'),
            'amountInCents'  => -1 * $amountInCents,
            'description'   => 'Withdrawal',
        ];
    }
}
```

```
<?php declare(strict_types=1);
final class BankAccount {
    private array $transactions = [];

    public function balanceInCents(): int {
        $balanceInCents = 0;

        foreach ($this->transactions as $transaction) {
            $balanceInCents += $transaction['amountInCents'];
        }

        return $balanceInCents;
    }
    /**
     * @param positive-int $amountInCents
     */
    public function deposit(int $amountInCents): void {
        $this->transactions[] = [
            'timestamp'      => new DateTimeImmutable('now'),
            'amountInCents'  => $amountInCents,
            'description'   => 'Deposit',
        ];
    }
    /**
     * @param positive-int $amountInCents
     */
    public function withdraw(int $amountInCents): void {
        $this->transactions[] = [
            'timestamp'      => new DateTimeImmutable('now'),
            'amountInCents'  => -1 * $amountInCents,
            'description'   => 'Withdrawal',
        ];
    }
}
```

```
<?php declare(strict_types=1);
final class BankAccount {
    private array $transactions = [];

    public function balanceInCents(): int {
        $balanceInCents = 0;

        foreach ($this->transactions as $transaction) {
            $balanceInCents += $transaction['amountInCents'];
        }

        return $balanceInCents;
    }
    /**
     * @param positive-int $amountInCents
     */
    public function deposit(int $amountInCents): void {
        $this->transactions[] = [
            'timestamp'      => new DateTimeImmutable('now'),
            'amountInCents'  => $amountInCents,
            'description'   => 'Deposit',
        ];
    }
    /**
     * @param positive-int $amountInCents
     */
    public function withdraw(int $amountInCents): void {
        $this->transactions[] = [
            'timestamp'      => new DateTimeImmutable('now'),
            'amountInCents'  => -1 * $amountInCents,
            'description'   => 'Withdrawal',
        ];
    }
}
```

```
<?php declare(strict_types=1);
final class BankAccount {
    private array $transactions = [];

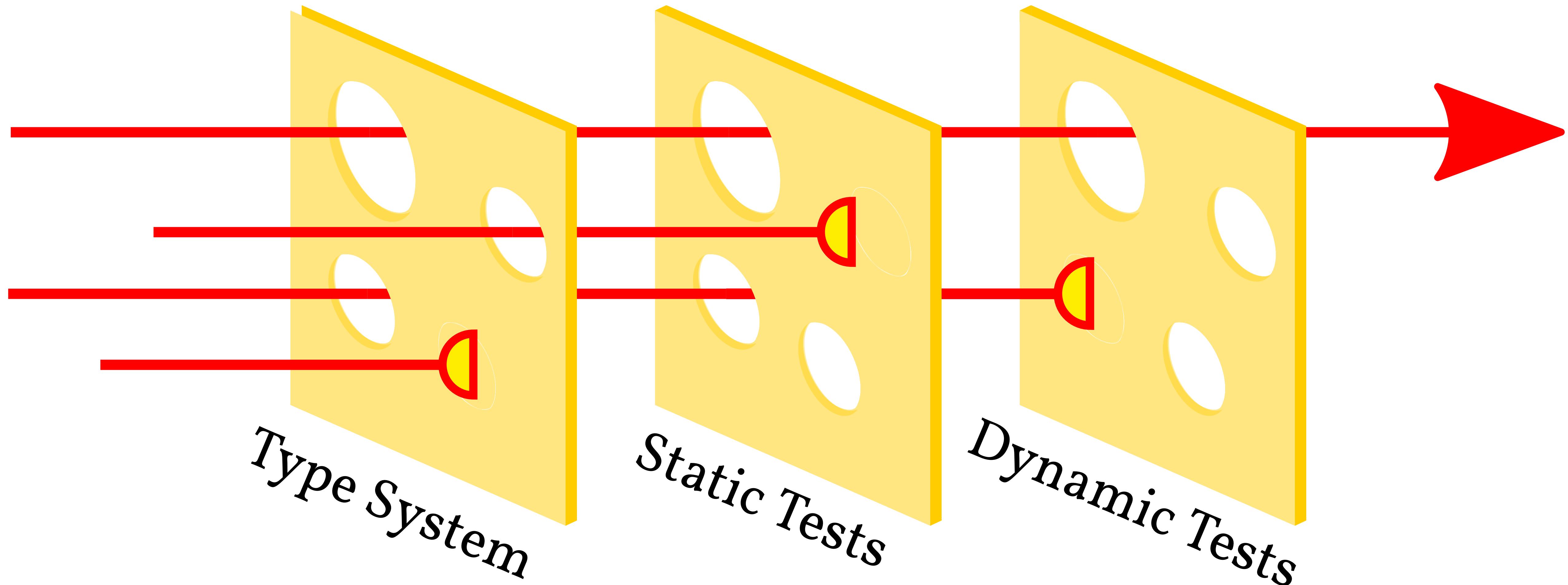
    public function balanceInCents(): int {
        $balanceInCents = 0;

        foreach ($this->transactions as $transaction) {
            $balanceInCents += $transaction['amountInCents'];
        }

        return $balanceInCents;
    }
    /**
     * @param positive-int $amountInCents
     */
    public function deposit(int $amountInCents): void {
        $this->transactions[] = [
            'timestamp'      => new DateTimeImmutable('now'),
            'amountInCents'  => $amountInCents,
            'description'   => 'Deposit',
        ];
    }
    /**
     * @param positive-int $amountInCents
     */
    public function withdraw(int $amountInCents): void {
        $this->transactions[] = [
            'timestamp'      => new DateTimeImmutable('now'),
            'amountInCents'  => -1 * $amountInCents,
            'description'   => 'Withdrawal',
        ];
    }
}
```

„[U]nit tests are not enough. Type systems are not enough. Contracts are not enough, formal specs are not enough, code review isn't enough, nothing is enough. We have to use everything we have to even hope of writing correct code, because there's only one way a program is right and infinite ways a program can be wrong, and we can't assume that any tool we use will prevent more than a narrow slice of all those wrong ways.“

Hillel Wayne



# Agenda

Documentation

Guidelines

Testing

AI

AppointmentListTest.php

```
14 final class AppointmentListTest extends TestCase
15     public static function provider(): array
16         new DateTimeImmutable('2025-07-31 11:00'),
17             ),
18             new Appointment(
19                 new DateTimeImmutable('2025-08-01 09:00'),
20                 new DateTimeImmutable('2025-08-01 11:00'),
21             ),
22             ),
23         ],
24         [
25             'August 25, 2025 and August 27, 2025 and August 29, 2025, each 09:00 to
26             new AppointmentList(
27                 new Appointment(
28                     new DateTimeImmutable('2025-08-25 09:00'),
29                     new DateTimeImmutable('2025-08-25 11:00'),
30                 ),
31                 new Appointment(
32                     new DateTimeImmutable('2025-08-27 09:00'),
33                     new DateTimeImmutable('2025-08-27 11:00'),
34                 ),
35                 new Appointment(
36                     new DateTimeImmutable('2025-08-29 09:00'),
37                     new DateTimeImmutable('2025-08-29 11:00'),
38                 ),
39             ),
40         ],
41     ];
42 }
43
44 #[DataProvider('provider')]
45 public function testCanBeRepresentedAsText(string $expected, AppointmentList $appointments)
46 {
47     $this->assertSame($expected, $appointments->asString());
48 }
```

Junie Ultimate

← Implement AppointmentList asString() Method

✓ Start Again Rollback All

☰ Implement the AppointmentList::asString() method

ⓘ Plan

- ✓ 1. Implement AppointmentList::asString() with sorting, grouping, and formatting logic.
- ✓ 2. Run ./tools/php-cs-fixer fix to ensure style compliance.
- ✓ 3. Run ./tools/phpstan for static analysis.
- ✓ 4. Run ./tools/phpunit to confirm all tests pass.
- ✓ 5. If any issues arise in steps 2-3, fix and rerun tests.
- ✓ 6. Provide final summary and submit.

ⓘ Get structure AppointmentList.php

The full content of a small source file was retrieved and displayed.

ⓘ Get structure Appointment.php

The entire content of the small Appointment.php file, consisting of 76 lines, has been displayed.

ⓘ Get structure AppointmentListWithDifferentDurationsException.php

The entire content of a small exception class file was retrieved and displayed.

☒ Terminal

Type your task here, press ↵

AppointmentListTest.php

```
14 final class AppointmentListTest extends TestCase
15     public static function provider(): array
16         new DateTimeImmutable('2025-07-31 11:00'),
17             ),
18             new Appointment(
19                 new DateTimeImmutable('2025-08-01 09:00'),
20                 new DateTimeImmutable('2025-08-01 11:00'),
21             ),
22         ),
23     ],
24     [
25         'August 25, 2025 and August 27, 2025 and August 29, 2025, each 09:00 to
26         new AppointmentList(
27             new Appointment(
28                 new DateTimeImmutable('2025-08-25 09:00'),
29                 new DateTimeImmutable('2025-08-25 11:00'),
30             ),
31             new Appointment(
32                 new DateTimeImmutable('2025-08-27 09:00'),
33                 new DateTimeImmutable('2025-08-27 11:00'),
34             ),
35             new Appointment(
36                 new DateTimeImmutable('2025-08-29 09:00'),
37                 new DateTimeImmutable('2025-08-29 11:00'),
38             ),
39         ),
40     ],
41 ];
42
43 #[DataProvider('provider')]
44 public function testCanBeRepresentedAsText(string $expected, AppointmentList $appointments)
45 {
46     $this->assertSame($expected, $appointments->asString());
47 }
```

Junie Ultimate

← Implement AppointmentList asString() Method

Start Again Rollback All Show Plan

Terminal Open

```
./tools/phpunit
```

The PHPUnit tests ran with 20 tests and 14 assertions, but 8 tests failed due to a call to an undefined method.

Update on task progress

Ran the full PHPUnit suite; 8 errors due to missing AppointmentList::asString(). Examined Appointment and AppointmentList classes and the tests to derive formatting rules. Key requirements: sort appointments by start time; format dates as 'F d, Y' and times as 'H:i' with timezone abbreviation; handle three cases: single appointment; multiple with identical time windows across days—if dates are consecutive, compress to 'startDate to endDate, from ...', else list dates joined with ' and ', followed by ', each ...'; otherwise, list each appointment with date and time, omitting date for subsequent appointments on the same day, joined by ' a...  
[Click to expand](#)

Edit AppointmentList.php +78

The AppointmentList class was updated to include a new method that formats the appointments as a string with detailed and formatted date and time information.

Terminal Open

```
./tools/phpunit
```

Type your task here, press ↲

AppointmentListTest.php x Junie Ultimate

```
14 final class AppointmentListTest extends TestCase
15     public static function provider(): array
16         new DateTimeImmutable('2025-07-31 11:00'),
17             ),
18             new Appointment(
19                 new DateTimeImmutable('2025-08-01 09:00'),
20                 new DateTimeImmutable('2025-08-01 11:00'),
21             ),
22         ),
23     ],
24     [
25         'August 25, 2025 and August 27, 2025 and August 29, 2025, each 09:00 to
26         new AppointmentList(
27             new Appointment(
28                 new DateTimeImmutable('2025-08-25 09:00'),
29                 new DateTimeImmutable('2025-08-25 11:00'),
30             ),
31             new Appointment(
32                 new DateTimeImmutable('2025-08-27 09:00'),
33                 new DateTimeImmutable('2025-08-27 11:00'),
34             ),
35             new Appointment(
36                 new DateTimeImmutable('2025-08-29 09:00'),
37                 new DateTimeImmutable('2025-08-29 11:00'),
38             ),
39         ),
40     ],
41 ];
42
43 #[DataProvider('provider')]
44 public function testCanBeRepresentedAsText(string $expected, AppointmentList $appointments)
45 {
46     $this->assertSame($expected, $appointments->asString());
47 }
```

← Implement AppointmentList asString() Method

✓ Start Again Rollback All Show Plan

Terminal Open

```
./tools/phpunit
```

All 20 tests with 22 assertions passed successfully using PHPUnit.

ⓘ Update on task progress

Implemented AppointmentList::asString() and ran the test suite; all 20 tests now pass. The method handles sorting, grouping, same-window detection, consecutive date ranges, and formatting per test expectations.

Terminal Open

```
./tools/php-cs-fixer fix
```

PHP CS Fixer analyzed and fixed one file out of ten, completing the process successfully.

Terminal Open

```
./tools/phpunit
```

All 20 PHPUnit tests passed successfully with 22 assertions.

Terminal Open

```
./tools/phpstan
```

Type your task here, press ↵

AppointmentListTest.php

```
14 final class AppointmentListTest extends TestCase
15     public static function provider(): array
16         new DateTimeImmutable('2025-07-31 11:00'),
17             ),
18             new Appointment(
19                 new DateTimeImmutable('2025-08-01 09:00'),
20                 new DateTimeImmutable('2025-08-01 11:00'),
21             ),
22         ),
23     ],
24     [
25         'August 25, 2025 and August 27, 2025 and August 29, 2025, each 09:00 to
26         new AppointmentList(
27             new Appointment(
28                 new DateTimeImmutable('2025-08-25 09:00'),
29                 new DateTimeImmutable('2025-08-25 11:00'),
30             ),
31             new Appointment(
32                 new DateTimeImmutable('2025-08-27 09:00'),
33                 new DateTimeImmutable('2025-08-27 11:00'),
34             ),
35             new Appointment(
36                 new DateTimeImmutable('2025-08-29 09:00'),
37                 new DateTimeImmutable('2025-08-29 11:00'),
38             ),
39         ),
40     ],
41 ];
42
43 #[DataProvider('provider')]
44 public function testCanBeRepresentedAsText(string $expected, AppointmentList $appoint
45 {
46     $this->assertSame($expected, $appointments->asString());
47 }
```

Junie Ultimate

← Implement AppointmentList asString() Method

✓ Start Again Rollback All Show Plan

Terminal Open

```
./tools/phpstan
```

PHPStan analysis found one error regarding a potential invalid offset access in the code.

ⓘ Open AppointmentList.php

The AppointmentList.php file was opened successfully, revealing the class definition with a method that throws an exception if appointment durations differ.

ⓘ Update on task progress

PHPStan reported one error due to indexing with \$count - 1 for the last element. Will adjust the code to fetch the last appointment using end() with a type assertion to satisfy PHPStan, then rerun static analysis and tests.

>Edit AppointmentList.php +1

The file was updated to include an additional imported function in the list of used functions.

Edit AppointmentList.php +3 -1

A code section was modified to safely handle the end appointment value by checking its type before formatting the date.

Terminal

Type your task here, press ↵

AppointmentListTest.php

```
14 final class AppointmentListTest extends TestCase
15     public static function provider(): array
16         new DateTimeImmutable('2025-07-31 11:00'),
17             ),
18             new Appointment(
19                 new DateTimeImmutable('2025-08-01 09:00'),
20                 new DateTimeImmutable('2025-08-01 11:00'),
21             ),
22         ),
23     ],
24     [
25         'August 25, 2025 and August 27, 2025 and August 29, 2025, each 09:00 to
26         new AppointmentList(
27             new Appointment(
28                 new DateTimeImmutable('2025-08-25 09:00'),
29                 new DateTimeImmutable('2025-08-25 11:00'),
30             ),
31             new Appointment(
32                 new DateTimeImmutable('2025-08-27 09:00'),
33                 new DateTimeImmutable('2025-08-27 11:00'),
34             ),
35             new Appointment(
36                 new DateTimeImmutable('2025-08-29 09:00'),
37                 new DateTimeImmutable('2025-08-29 11:00'),
38             ),
39         ),
40     ],
41 ];
42
43 #[DataProvider('provider')]
44 public function testCanBeRepresentedAsText(string $expected, AppointmentList $appoint
45 {
46     $this->assertSame($expected, $appointments->asString());
47 }
```

Junie Ultimate

← Implement AppointmentList asString() Method

Start Again Rollback All Show Plan

Terminal Open

```
./tools/phpstan
```

PHPStan analysis completed with one error reported regarding an always true instanceof check in the code.

Update on task progress

PHPStan flagged an always-true instanceof due to the previous fix. I'll switch to using array\_key\_last() with a null guard to select the last element, which should satisfy static analysis cleanly.

Edit AppointmentList.php +1 -1

A function import statement was replaced in the file, changing one of the used functions to array\_key\_last.

Edit AppointmentList.php +3 -3

The code was modified to replace the method of obtaining the last appointment with a safer approach using the last array key.

Terminal Open

```
./tools/phpstan
```

The PHPStan analysis completed successfully with all checks passing and no errors reported.

Type your task here, press ↵

AppointmentListTest.php

```
14 final class AppointmentListTest extends TestCase
15     public static function provider(): array
16         new DateTimeImmutable('2025-07-31 11:00'),
17             ),
18             new Appointment(
19                 new DateTimeImmutable('2025-08-01 09:00'),
20                 new DateTimeImmutable('2025-08-01 11:00'),
21             ),
22         ),
23     ],
24     [
25         'August 25, 2025 and August 27, 2025 and August 29, 2025, each 09:00 to
26         new AppointmentList(
27             new Appointment(
28                 new DateTimeImmutable('2025-08-25 09:00'),
29                 new DateTimeImmutable('2025-08-25 11:00'),
30             ),
31             new Appointment(
32                 new DateTimeImmutable('2025-08-27 09:00'),
33                 new DateTimeImmutable('2025-08-27 11:00'),
34             ),
35             new Appointment(
36                 new DateTimeImmutable('2025-08-29 09:00'),
37                 new DateTimeImmutable('2025-08-29 11:00'),
38             ),
39         ),
40     ],
41 ];
42
43 #[DataProvider('provider')]
44 public function testCanBeRepresentedAsText(string $expected, AppointmentList $appointments)
45 {
46     $this->assertSame($expected, $appointments->asString());
47 }
```

Junie Ultimate

← Implement AppointmentList asString() Method

✓ Start Again Rollback All Show Plan

Terminal Open

```
./tools/phpstan
```

The PHPStan analysis completed successfully with all checks passing and no errors reported.

Terminal Open

```
./tools/phpunit
```

All 20 tests passed successfully with 22 assertions and no errors.

ⓘ Update on task progress

All changes implemented. Static analysis and code style verified. Test suite passes fully. Ready to finalize.

Done

Changed Files

AppointmentList.php +91

Implemented AppointmentList::asString() to create formatted appointment lists. The method sorts, groups, and formats appointments with handling of shared and mixed time windows. All tests passed after running code style, static analysis, and unit tests.

Rollback

Type your task here, press ↲

```
10  final readonly class AppointmentList
11
12      public function asString(): string
13  {
14
15      $appointments = $this->appointments;
16
17
18      // Sort by start time to ensure consistent output order
19      usort($appointments, static fn (Appointment $a, Appointment $b): int => $a->startTime() <=> $b->startTime());
20
21
22      $formatDate  = static fn (Appointment $a): string => $a->startTime()->format('F d, Y');
23      $formatStart = static fn (Appointment $a): string => $a->startTime()->format('H:i');
24      $formatEnd   = static fn (Appointment $a): string => $a->endTime()->format('H:i');
25      $formatTz    = static fn (Appointment $a): string => $a->startTime()->format('T');
26
27
28      $count = count($appointments);
29
30
31      if ($count === 1) {
32          $a = $appointments[0];
33
34
35          return $formatDate($a) . ' from ' . $formatStart($a) . ' to ' . $formatEnd($a) . ' ' . $formatTz($a);
36      }
37
38
39
40
41      // Check if all appointments have the same time window and timezone
42      $first      = $appointments[0];
43      $sameWindow = true;
44      $firstStart = $formatStart($first);
45      $firstEnd   = $formatEnd($first);
46      $firstTz    = $formatTz($first);
47
48
49      foreach ($appointments as $a) {
50          if ($formatStart($a) !== $firstStart || $formatEnd($a) !== $firstEnd || $formatTz($a) !== $firstTz) {
51              $sameWindow = false;
52
53              break;
54          }
55      }
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
```



AppointmentList.php x

10 final readonly class AppointmentList

37 public function asString(): string

72 if (\$sameWindow) {

73 // Check if all dates are consecutive (each next day is exactly +1 day)

74 \$consecutive = true;

75

76 for (\$i = 1; \$i < \$count; \$i++) {

77 \$prev = \$appointments[\$i - 1];

78 \$curr = \$appointments[\$i];

79

80 if (\$prev->startTime()->modify('+1 day')->format('Y-m-d') !== \$curr->startTime()->format('Y-m-d')) {

81 \$consecutive = false;

82

83 break;

84 }

85 }

86

87 if (\$consecutive) {

88 \$startDate = \$formatDate(\$appointments[0]);

89 \$lastKey = array\_key\_last(\$appointments); // int|null

90 \$lastKey = \$lastKey ?? 0;

91 \$endDate = \$formatDate(\$appointments[\$lastKey]);

92

93 return \$startDate . ' to ' . \$endDate . ', from ' . \$firstStart . ' to ' . \$firstEnd . ' ' . \$firstTz;

94 }

95

96 // Non-consecutive dates but same time window: list dates and add ", each ..."

97 \$dates = [];

98

99 foreach (\$appointments as \$a) {

100 \$dates[] = \$formatDate(\$a);

101 }

102

103 return implode(' and ', \$dates) . ', each ' . \$firstStart . ' to ' . \$firstEnd . ' ' . \$firstTz;

104

AppointmentList.php x

10 final readonly class AppointmentList

37 public function asString(): string

106 // Different time windows: build segments, omitting duplicate date for subsequent appointments on the same day

107 \$segments = [];

108 \$previousDate = null;

109

110 foreach (\$appointments as \$a) {

111 \$date = \$formatDate(\$a);

112 \$segmentTime = 'from ' . \$formatStart(\$a) . ' to ' . \$formatEnd(\$a) . ' ' . \$formatTz(\$a);

113

114 if (\$date === \$previousDate) {

115 \$segments[] = \$segmentTime;

116 } else {

117 \$segments[] = \$date . ' ' . \$segmentTime;

118 \$previousDate = \$date;

119 }

120 }

121

122 return implode(' and ', \$segments);

123 }

124

125 *\*\**

126 \* param array<Appointment> \$appointments

127 \*

128 \* throws EmptyAppointmentListException

129 \*/

130 private function ensureNotEmpty(array \$appointments): void

131 {

132 if (empty(\$appointments)) {

133 throw new EmptyAppointmentListException('Appointment list needs at least one appointment');

134 }

135 }

136

137 *\*\**

**This presentation has a home on the web:**



# Thank you!

- ➡ <https://phpunit.expert>
- ✉ [sebastian@thephp.cc](mailto:sebastian@thephp.cc)
- Ⓜ [@sebastian@phpc.social](https://@sebastian@phpc.social)



# Image Credits

---

- <https://www.pexels.com/de-de/foto/zeitrafferfotografie-wahrend-der-nacht-1103969>
- [https://commons.wikimedia.org/wiki/File:Swiss\\_cheese\\_model.svg](https://commons.wikimedia.org/wiki/File:Swiss_cheese_model.svg)