Writing Integration Tests

Anna Filina
TESTING AND LEGACY EXPERT
@afilina afilina.com

Summary

Integration vs unit tests

Integration with the database

Integration with the filesystem

Minimize integration tests

Integration vs Unit Tests

DoctrineProductRepository.php

```php
use App\Catalog\Value\Product;
//...

public function findProducts(): array
{
    $productDtos = $this->entityManager
        ->createQueryBuilder()
        ->select('product', 'discount')
        ->from(\App\Entity\Product::class, 'product')
        ->leftJoin('product.discounts', 'discount')
        ->getQuery()
        ->execute();

    return array_map(function (\App\Entity\Product $productDto): Product {
        return new Product(
            $productDto->name,
            $this->getProductPrice($productDto)
        );
    }, $productDtos);
}
```

Doctrine
Entity

Map properties
to database
fields

Validate data
using business
rules

Value
Object

Validate data using business rules

Map properties to database columns

Map properties to CSV columns

Map properties to JSON fields

Value Object

DoctrineProductRepository.php

```php
use App\Catalog\Value\Product;
//...

public function findProducts(): array
{
    $productDtos = $this->entityManager
        ->createQueryBuilder()
        ->select('product', 'discount')
        ->from(\App\Entity\Product::class, 'product')
        ->leftJoin('product.discounts', 'discount')
        ->getQuery()
        ->execute();

    return array_map(function (\App\Entity\Product $productDto): Product {
        return new Product(
            $productDto->name,
            $this->getProductPrice($productDto)
        );
    }, $productDtos);
}
```

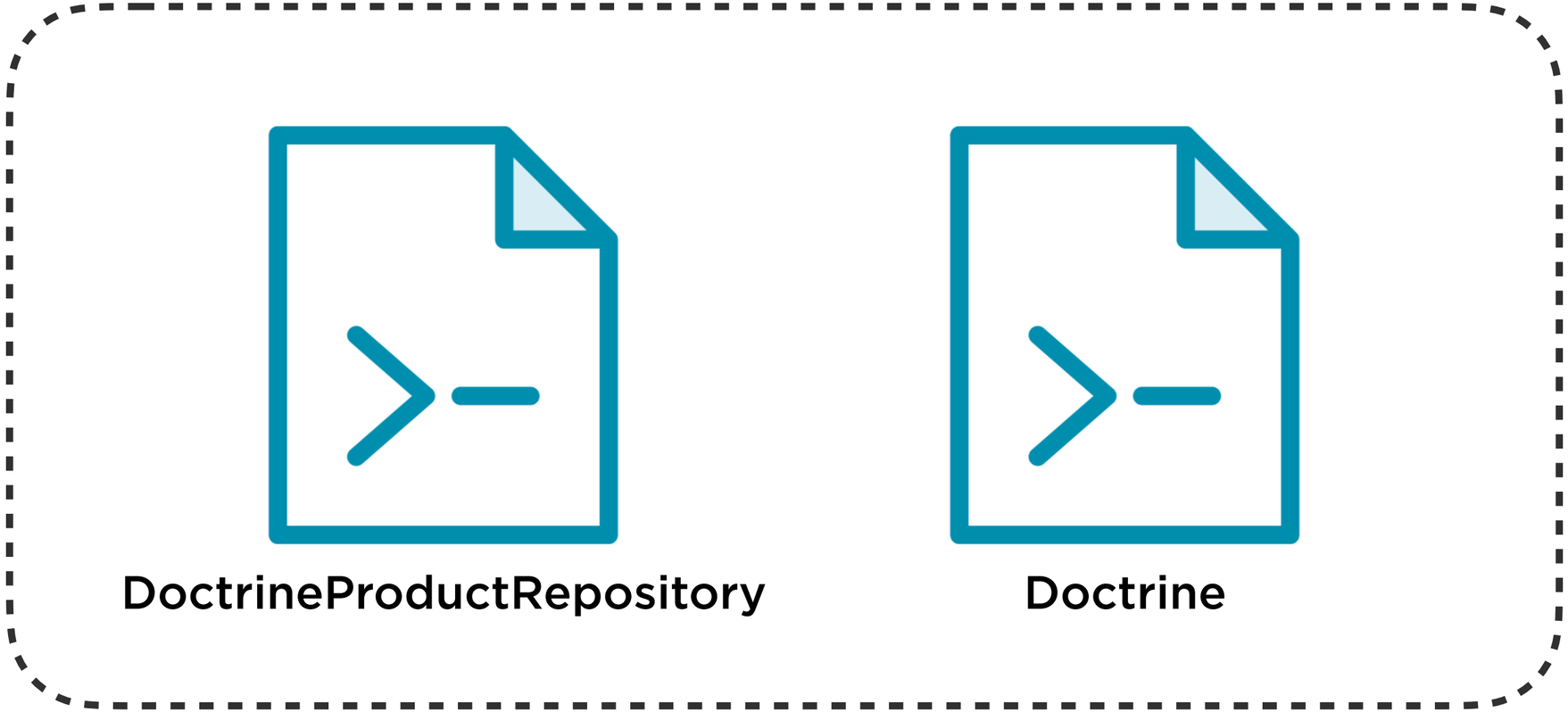
DoctrineProductRepositoryTest.php

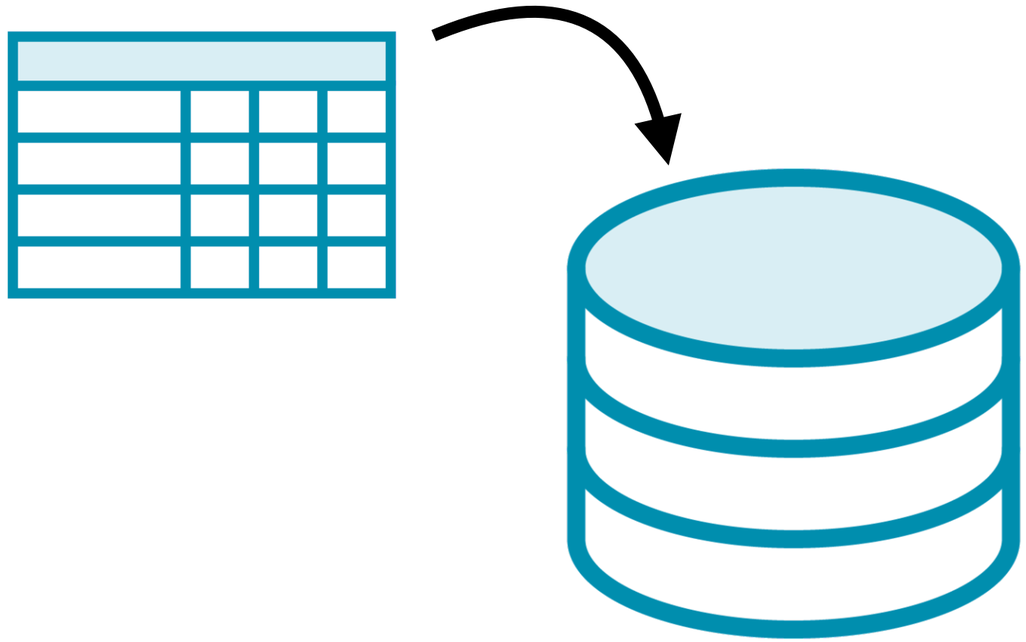
```php
$queryBuilder = $this->createMock(QueryBuilder::class);
$queryBuilder->expects($this->once())
    ->method('select')
    ->with('product', 'discount')
    ->willReturnSelf();
$queryBuilder->expects($this->once())
    ->method('from')
    ->with(Product::class, 'product')
    ->willReturnSelf();
$queryBuilder->expects($this->once())
    ->method('leftJoin')
    ->with('product.discounts', 'discount')
    ->willReturnSelf();
$queryBuilder->expects($this->once())
    ->method('getQuery')
    ->willReturn($query);
```

DoctrineProductRepositoryTest.php

```php
$repository = $this->diContainer->get(DoctrineProductRepository::class);
self::assertEquals(
    [
        new Product('Name 1', new Amount(1000)),
        new Product('Name 2', new Amount(1000)),
    ],
    $repository->findProducts()
);
```

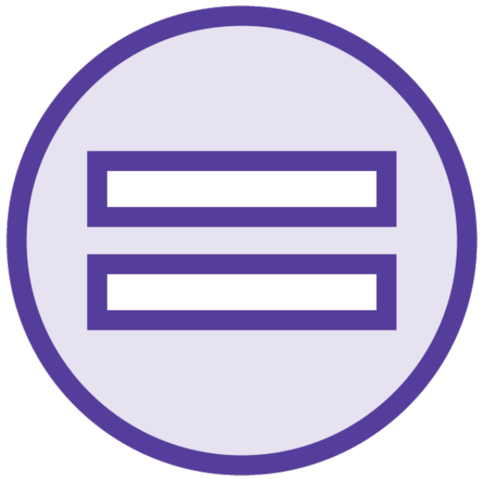
Integration With the Database


DoctrineProductRepository
Doctrine



Reset

Add record



Update record

IntegrationTestCase.php

```
abstract class IntegrationTestCase extends TestCase
{
}
```

```php
protected function resetDatabase(): void
{
    $this->importFixture(__DIR__ . '/fixtures/truncate.sql');
}
```

```sql
SET FOREIGN_KEY_CHECKS = 0;

TRUNCATE TABLE `product`;
TRUNCATE TABLE `discount`;

SET FOREIGN_KEY_CHECKS = 1;
```

IntegrationTestCase.php

```php
/**
 * @param array<string, mixed> $data
 */
protected function insertRecord(string $table, array $data): int
{
    return $this->getConnection()->insert($table, $data);
}
```

IntegrationTestCase.php

```php
/**
 * @param array<string, mixed> $data
 * @param array<string, mixed> $where
 */
protected function updateRecord(string $table, array $data, array $where): int
{
    return $this->getConnection()->update($table, $data, $where);
}
```

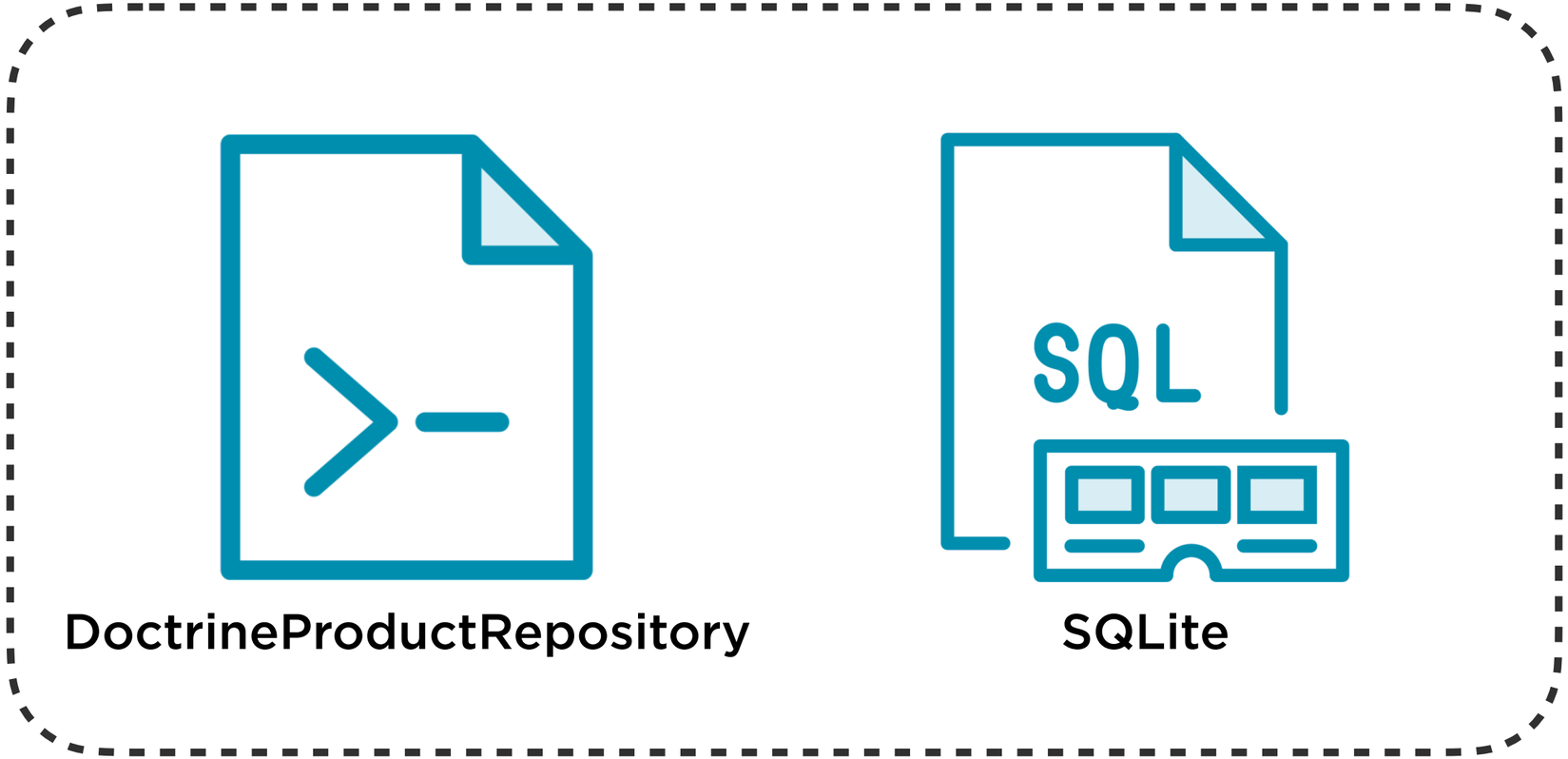
```php
protected function initializeContainer(): void
{
    $di = new DependencyInjection();
    $this->diContainer = $di->createContainer();
}
```

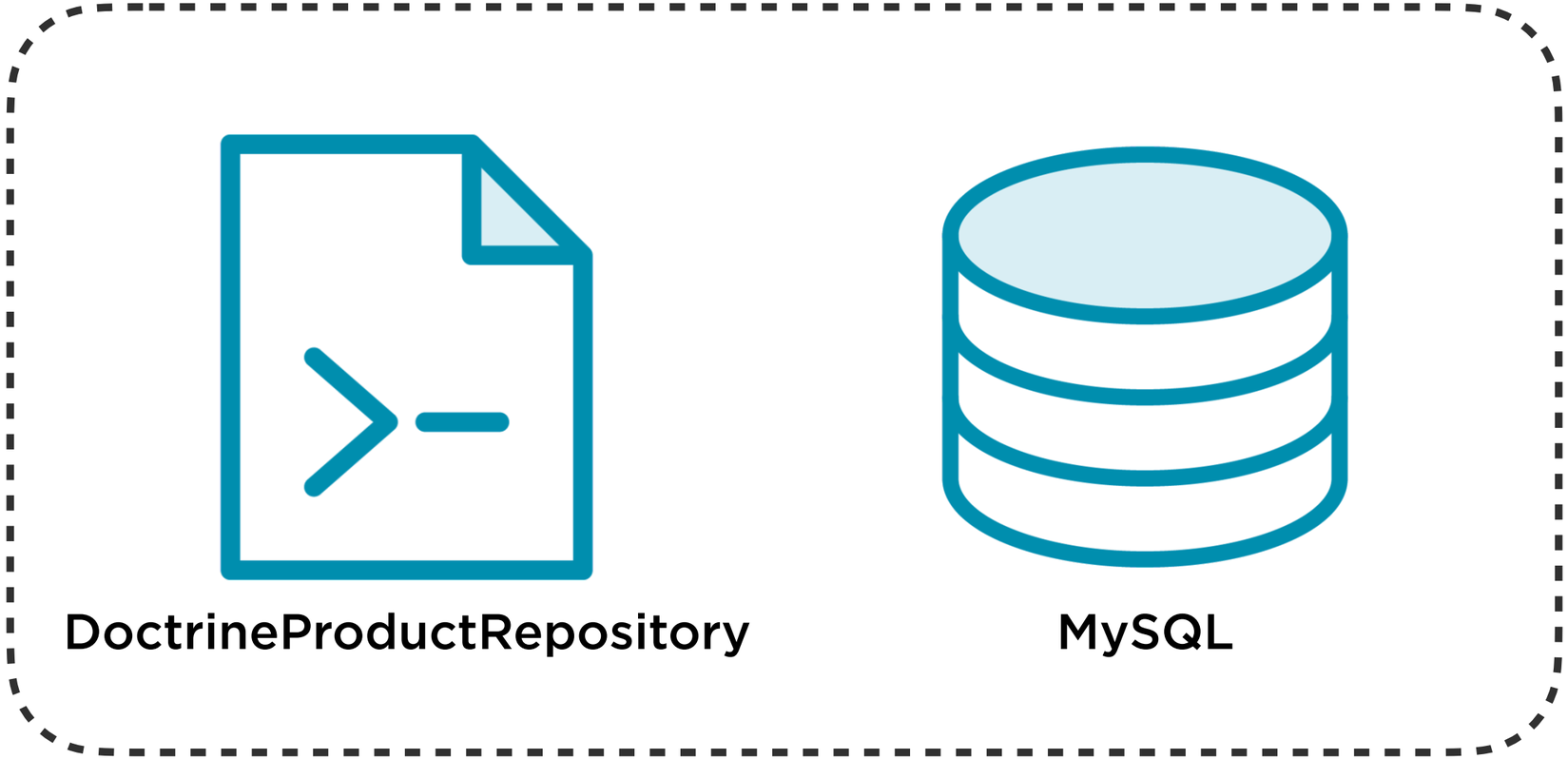
```php
$repository = $this->diContainer->get(DoctrineProductRepository::class);
```

DependencyInjection.php

```php
public function createContainer(): ContainerInterface
{
    $di = new Container();
    $di->set(Connection::class, DI\factory(function () {
        return new Connection(
            [
                'dbname' => 'pluralsight',
                'user' => 'pluralsight',
                'password' => 'pluralsight',
                'host' => 'mysql.phpunit.local',
                'port' => '3306',
            ],
            new Driver()
        );
    }));

    //...

    return $di;
}
```

```php
private function getProductPrice(\App\Entity\Product $productDto): Amount
{
    $cost = new Amount($productDto->cost);
    $priceOperations = [];
    $priceOperations[] = new MarkupOperation($productDto->markup / 100);

    foreach ($productDto->discounts as $discountDto) {
        $discount = $this->getProductDiscount($discountDto);
        $priceOperations[] = new DiscountOperation([$discount]);
    }

    return (new AmountCalculator())
        ->getResult($cost, $priceOperations);
}
```


SQL
DoctrineProductRepository
SQLite


DoctrineProductRepository
MySQL

Integration With the Filesystem

FilesystemSearchAnalytics.php

```php
public function track(array $searchFilters): void
{
    $csvLine = implode(',', [
        '"' . $this->clock->now()->format('c') . '",
        '"' . addslashes(json_encode($searchFilters)) . '"'
    ]);
    $this->filesystem->appendToFile(
        __DIR__ . '/../../../var/search_analytics.csv',
        $csvLine
    );
}
```

```php
/** @covers \App\Catalog\SearchAnalytics\FilesystemSearchAnalytics */
final class FilesystemSearchAnalyticsTest extends IntegrationTestCase
{
}
```

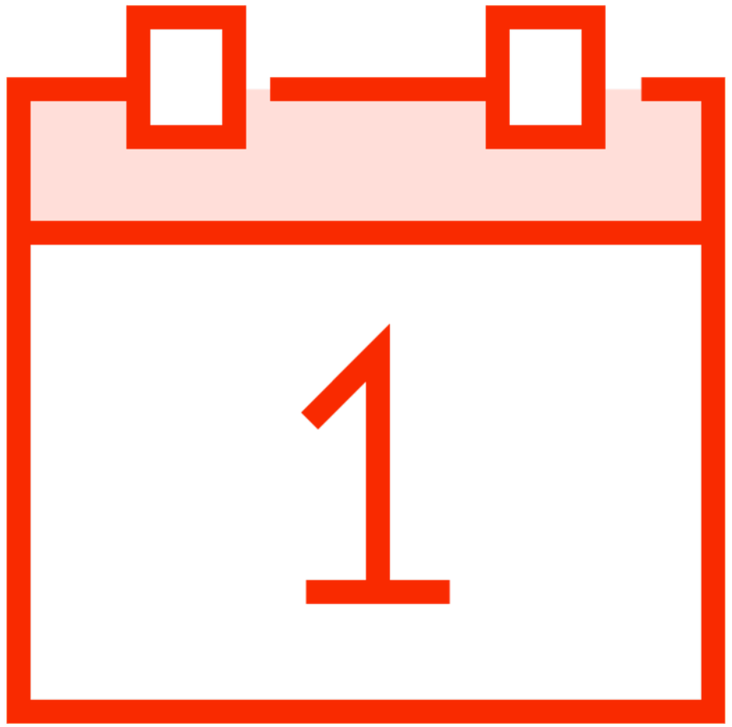
```php
/** @test */
public function track(): void
{
    $this->initializeContainer();
    $analytics = $this->diContainer->get(FilesystemSearchAnalytics::class);

    $analytics->track(['price' => null, 'name' => null]);

    self::assertEquals(
        '"2020-01-01T12:01:02+00:00","{\"price\":null,\"name\":null}"',
        file_get_contents(self::FILE_PATH)
    );
}
```

```php
const FILE_PATH = __DIR__ . '/../../../var/search_analytics.csv';

/** @test */
public function track(): void
{
    unlink(self::FILE_PATH);

    $this->initializeContainer();
    $analytics = $this->diContainer->get(FilesystemSearchAnalytics::class);

    $analytics->track(['price' => null, 'name' => null]);

    self::assertEquals(
        '"2020-01-01T12:01:02+00:00","{\"price\":null,\"name\":null}"',
        file_get_contents(self::FILE_PATH)
    );
}
```

```php
public function __construct(Filesystem $filesystem, Clock $clock)
{
    $this->filesystem = $filesystem;
    $this->clock = $clock;
}
```

```php
interface Clock
{
    public function now(): DateTimeImmutable;
}
```

```php
$this->initializeContainer();
$this->diContainer->set(Clock::class, DI\factory(function () {
    $clock = $this->createStub(Clock::class);
    $clock
        ->method('now')
        ->willReturn(new \DateTimeImmutable('2020-01-01 12:01:02', new
\DateTimeZone('UTC')));
    return $clock;
}));

$analytics = $this->diContainer->get(FilesystemSearchAnalytics::class);
```

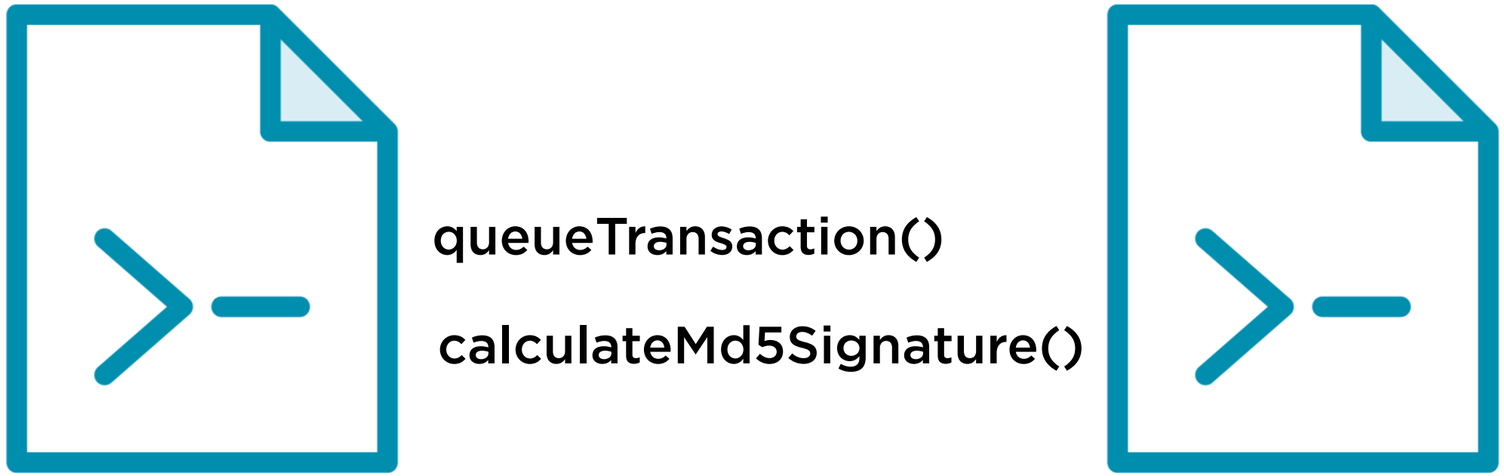


Mocking with today's date

Credit card expiration date

Minimize Integration Tests

```php
public function queueTransaction(array $data): void
{
    if ($this->requiresSignature) {
        $data['signature'] = $this->calculateMd5Signature(
            $data,
            $this->signatureKey
        );
    }
    $payload = json_encode($data);
    $this->queue->add($payload);
}
```

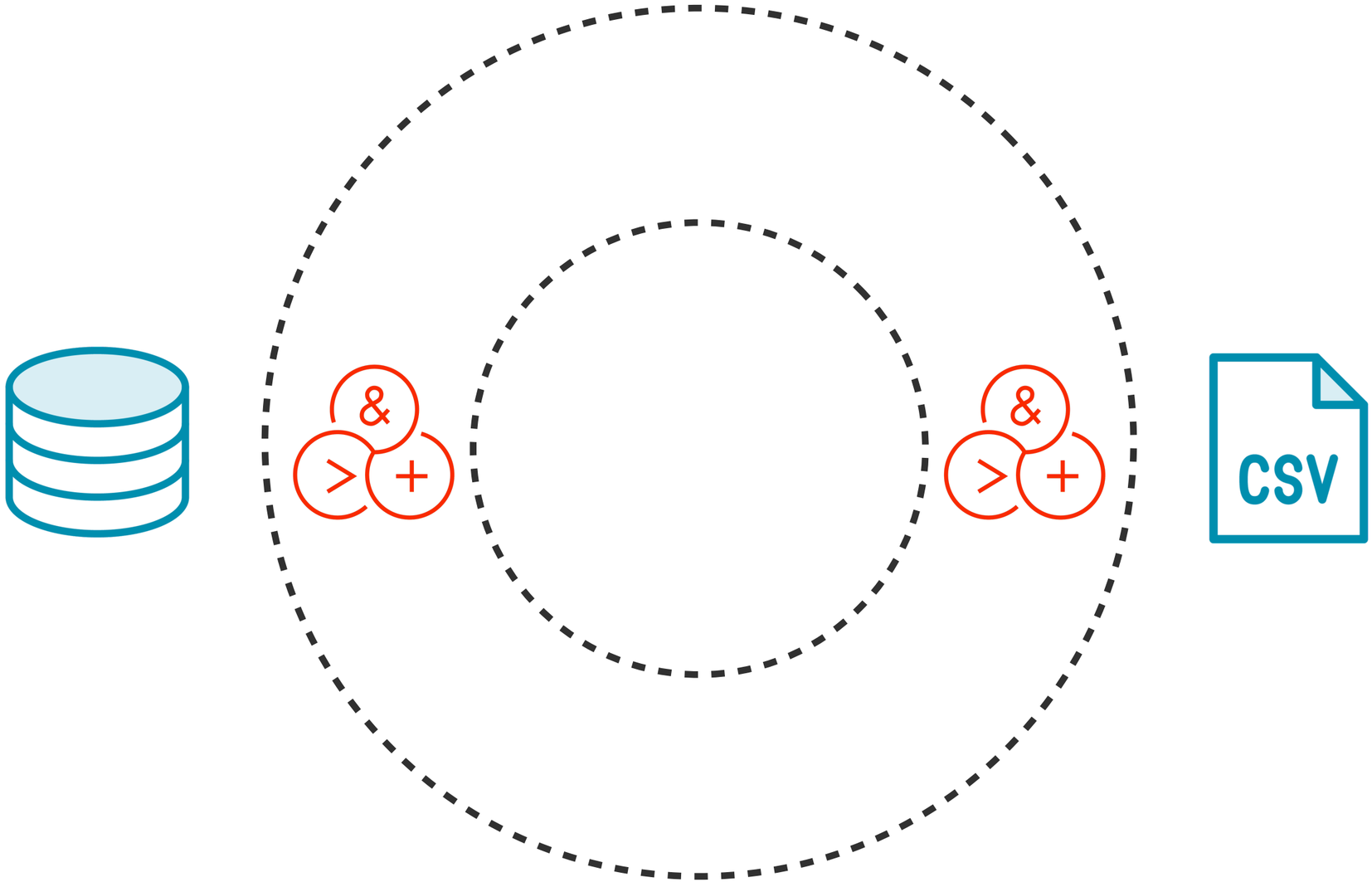

queueTransaction()
calculateMd5Signature()
TransactionQueue

queueTransaction()

TransactionQueue

calculateMd5Signature()

```php
public function queueTransaction(array $data): void
{
    $payload = json_encode($data);
    $this->queue->queue($payload);
}
```


&
>
+
&
>
+
CSV

Up Next:
Continuous Integration