4.4 codebase & engineering quality pack

پک کیفیت کد و مهندسی - ProDecks
نسخه: 1.0
تاریخ: 1403/11/16
وضعیت: کدبس MVP

فصل ۱: مرور کلی کدبس

۱.۱. اطلاعات repository

• پلتفرم: GitHub
• آدرس: https://github.com/prodecks/mvp-core
• نوع: Private Repository (با دسترسی برای سرمایه‌گذاران)
• حجم: ۴۵۰ فایل، ۲۵,۰۰۰ خط کد
• زبان‌های اصلی: PHP (۶۵٪), JavaScript (۲۵٪), CSS (۸٪), SQL (۲٪)

۱.۲. ساختار پروژه

```
prodecks-mvp/
├── includes/                # فایل‌های core سیستم
│   ├── auth.php             # احراز هویت سیستم
│   ├── config.php           # تنظیمات
│   ├── functions.php        # توابع کمکی
│   └── database.php         # مدیریت دیتابیس
├── modules/                 # ماژول‌های کسب‌وکار
│   ├── spaces/              # Spaces سیستم
│   ├── decks/               # Decks سیستم
│   ├── cards/               # Cards سیستم
│   └── gamification/        # سیستم گیمیفیکیشن
├── assets/                  # فایل‌های استاتیک
│   ├── css/
│   ├── js/
│   └── images/
├── api/                     # endpoint های API
│   ├── v1/
│   └── middleware/
├── tests/                   # تست‌ها
│   ├── unit/
│   ├── integration/
│   └── e2e/
```

```
├── docs/          # مستندات
├── scripts/       # اسکریپت‌های کمکی
└── docker/        # کانفیگ Docker
```

فصل ۲: استانداردهای کدنویسی

۲.۱. استانداردهای PHP

- پیروی می‌کند PSR-12 و PSR-1 از
- نام‌گذاری:
  - کلاس‌ها: PascalCase (UserController)
  - متدها: camelCase (getUserData)
  - متغیرها: snake_case ($user_id)
  - ثابت‌ها: UPPER_SNAKE_CASE (MAX_LOGIN_ATTEMPTS)

- ساختار فایل‌ها:

```php
<?php
declare(strict_types=1);
namespace ProDecks\Modules;
class Example { ... }
```

۲.۲. استانداردهای JavaScript

- پیروی می‌کند Airbnb JavaScript Style Guide از
- استفاده از ES6+ features
- توضیحات JSDoc برای توابع مهم

۲.۳. استانداردهای CSS

- استفاده از BEM Methodology
- ساختار: .block__element--modifier
- ترتیب properties: Positioning > Box Model > Typography > Visual

۲.۴. استانداردهای پایگاه داده

- نام جداول: جمع انگلیسی (users, spaces, decks)
- نام ستون‌ها: snake_case
- استفاده از foreign key constraints
- ایندکس‌گذاری مناسب

فصل ۳: مدیریت وابستگی‌ها

۳.۱. وابستگی‌های PHP (composer.json)

```
{
```

```
"require": {
"php": "^8.0",
"ext-pdo": "*",
"ext-json": "*",
"vlucas/phpdotenv": "^5.4",
"monolog/monolog": "^2.3"
},
"require-dev": {
"phpunit/phpunit": "^9.5",
"squizlabs/php_codesniffer": "^3.6"
}
}
```


۳.۲. وابستگی‌های JavaScript (package.json)


```
{
"dependencies": {
"sortablejs": "^1.14.0",
"axios": "^0.27.0",
"chart.js": "^3.8.0"
},
"devDependencies": {
"jest": "^28.1.0",
"eslint": "^8.16.0"
}
}
```


۳.۳. مدیریت نسخه‌ها

- استفاده از Semantic Versioning (SemVer)
- نگهداری CHANGELOG.md
- منظم در Git Tagging

فصل ۴: سیستم تست

۴.۱. تست واحد (Unit Tests)

- پوشش: ۷۲٪ برای منطق کسب‌وکار
- فریم‌ورک: PHPUnit برای PHP، Jest برای JavaScript
- اجرای خودکار با GitHub Actions

مثال تست واحد:

```
<?php
class UserTest extends TestCase {
public function testUserCreation() {
$user = new User('test@example.com');
$this->assertEquals('test@example.com', $user->getEmail());
}
}
```

۴.۲. تست یکپارچگی (Integration Tests)

• تست ارتباط با دیتابیس
• تست API endpoints
• تست ماژول‌های مرتبط

۴.۳. تست end-to-end

• استفاده از Cypress
• سناریو اصلی کاربر ۱۵
• تست‌های cross-browser

۴.۴. گزارش پوشش کد (Code Coverage)

• کل پروژه: ۶۵٪
• Core modules: ۸۰٪
• گزارش‌های HTML در artifacts

فصل ۵: فرآیند توسعه

۵.۱. Workflow توسعه

۱. ایجاد issue در GitHub
۲. ایجاد branch از develop
```
git checkout -b feature/space-invitations
```
۳. توسعه و commit
```
git commit -m "feat: add space invitation system"
```
۴. ایجاد Pull Request
۵. بررسی کد (Code Review)
۶. merge پس از تأیید

۵.۲. قوانین commit

• فرمت: type(scope): description
• انواع: feat, fix, docs, style, refactor, test, chore
• مثال: feat(spaces): add bulk invite functionality

۵.۳. Code Review Checklist

✅ کد خوانا و قابل فهم است
✅ تست‌های مربوطه نوشته شده‌اند
✅ استانداردهای کدنویسی رعایت شده‌اند
✅ مستندات به‌روز شده‌اند
✅ security considerations بررسی شده‌اند

فصل ۶: مدیریت دانش

مستندات پروژه ۶.۱.

• README.md - راهنمای شروع
• ARCHITECTURE.md - معماری سیستم
• API_DOCS.md - مستندات API
• DEPLOYMENT.md - راهنمای استقرار
• CONTRIBUTING.md - راهنمای مشارکت

توسعه‌دهندگان جدید onboarding ۶.۲.

۱. دسترسی به repository
۲. مطالعه مستندات
۳. راه‌اندازی محیط توسعه
۴. اجرای تست‌ها
۵. کار روی issue ساده

انتقال دانش در صورت خروج عضو کلیدی ۶.۳.

• همه دانش در مستندات و کد کامنت‌شده موجود است
• جلسات knowledge sharing و pairing ماهانه
• ویدیوهای آموزشی ضبط شده

فصل ۷: کیفیت کد

معیارهای کیفیت ۷.۱.

• Complexity: میانگین Cyclomatic Complexity کمتر از ۱۰
• Duplication: کمتر از ۵٪ کد تکراری
• Issues: باز در هر زمان issue کمتر از ۱۰
• Debt: technical debt کمتر از ۵ روز

ابزارهای تحلیل کیفیت ۷.۲.

• SonarQube برای تحلیل استاتیک
• PHPStan برای type checking
• ESLint برای JavaScript
• PHP_CodeSniffer برای استانداردهای کد

بهبود مستمر ۷.۳.

• جلسات refactoring هفتگی
• metrics بررسی ماهانه
• technical debt اولویت‌دهی به

فصل ۸: قابلیت نگهداری

ساختار ماژولار ۸.۱.

• مشخص responsibility هر ماژول مستقل و با
• رابط‌های واضح بین ماژول‌ها
• حداقل وابستگی بین ماژول‌ها

طراحی برای تغییر ۸.۲.

• مناسب design patterns استفاده از
• dependency injection
• configuration externalization

مستندات کد ۸.۳.

• توضیحات برای کلاس‌ها و متدهای پیچیده
• usage مثال‌های
• برای جریان‌های مهم sequence نمودارهای

فصل ۹: مقیاس‌پذیری کدبس

طراحی برای رشد ۹.۱.

• امکان اضافه کردن ماژول‌های جدید بدون تغییر ماژول‌های موجود
• ساختار فایل‌بندی قابل گسترش
• separation of concerns

۹.۲. performance considerations

• ماژول‌ها lazy loading
• در سطح مناسب caching
• برای لیست‌های بزرگ pagination

آمادگی برای معماری جدید ۹.۳.

• به میکروسرویس extract کد آماده برای
• برای دیتابیس abstraction layer
• پایه event-driven architecture

فصل ۱۰: امنیت کد

امنیت در توسعه ۱۰.۱.

• آموزش‌های امنیتی برای توسعه‌دهندگان

• استفاده از linterهای امنیتی

• بررسی dependencyها برای vulnerability

۱۰.۲. secure coding practices

• input validation در تمام سطوح

• output escaping

• prepared statements برای تمام queries

۱۰.۳. بررسی‌های امنیتی

• بررسی کد قبل از merge

• scans دوره‌ای با ابزارهای امنیتی

• penetration testing هر ۶ ماه

فصل ۱۱: CI/CD Pipeline

۱۱.۱. GitHub Actions Workflow

```
name: CI Pipeline
on: [push, pull_request]
jobs:
test:
runs-on: ubuntu-latest
steps:
- uses: actions/checkout@v2
- uses: actions/setup-php@v2
- run: composer install
- run: php vendor/bin/phpunit
lint:
runs-on: ubuntu-latest
steps:
- uses: actions/checkout@v2
- run: php vendor/bin/phpcs
```

۱۱.۲. مراحل pipeline

۱. Lint و بررسی استانداردها

۲. اجرای تست واحد

۳. اجرای تست یکپارچگی

۴. build و packaging

۵. deployment به staging

۱۱.۳. کیفیت دروازه‌ای (Quality Gates)

• تمام تست‌ها باید pass شوند

• coverage نباید کاهش یابد

• هیچ vulnerability بحرانی نباید وجود داشته باشد

فصل ۱۲: Technical Debt مدیریت

۱۲.۱. شناسایی Debt

- ابزارها: SonarQube، CodeClimate
- بررسی دستی در code reviews
- بازخورد کاربران

۱۲.۲. ردیابی Debt

- ایجاد debt برای هر issue
- برچسب‌گذاری: tech-debt, refactor
- تخمین زمان و اولویت

۱۲.۳. پرداخت Debt

- اختصاص ۲۰٪ زمان توسعه به refactoring
- پرداخت debt قبل از اضافه کردن feature جدید
- بازبینی ماهانه debtها

فصل ۱۳: نتیجه‌گیری

کدبس ProDecks با استانداردهای بالا و با تمرکز بر قابلیت نگهداری و توسعه‌پذیری ایجاد شده است.
تیم متعهد به حفظ کیفیت کد و بهبود مستمر فرآیندهای مهندسی است.
سیستم‌های تست، بررسی کد و CI/CD تضمین‌کننده تحویل پایدار و قابل اعتماد هستند.

ضمیمه‌ها

ضمیمه ۱: گزارش کامل Code Quality
ضمیمه ۲: نمونه‌هایی از تست‌ها
ضمیمه ۳: قالب Code Review
ضمیمه ۴: نقشه فرآیندهای توسعه

تهیه‌کنندگان:
حامد کوهی (سرپرست فنی)
تیم توسعه ProDecks
