4.2 system architecture document

سند معماری سیستم - ProDecks
نسخه: 2.0
تاریخ: 1403/11/15
وضعیت: MVP - معماری پایه

فصل ۱: مرور کلی معماری

۱.۱. مقدمه
معماری ProDecks طراحی شده است که شامل لایه (Three-tier Architecture) بر پایه الگوی سه‌لایه presentation، business logic و لایه data access و مقیاس‌پذیری می‌باشد. این معماری به دلیل سادگی، قابلیت نگهداری و لایه مناسب برای استارتاپ‌ها انتخاب شده است.

۱.۲. اهداف معماری
• مقیاس‌پذیری افقی برای پشتیبانی از رشد کاربران
• قابلیت اطمینان بالا (High Availability)
• امنیت داده‌ها و تراکنش‌ها
• توسعه‌پذیری و قابلیت نگهداری آسان
• performance بهینه با کمترین latency

۱.۳. محدوده معماری
این سند معماری سیستم ProDecks را در نسخه MVP و شامل تصمیم‌های معماری، نمودارها، پوشش می‌دهد و انتخاب تکنولوژی‌ها و طرح‌های توسعه آینده است.

فصل ۲: نمودارهای معماری

۲.۱. نمودار کلی سیستم

[توضیح: این بخش در مستند واقعی شامل نمودار می‌شود]

سیستم ProDeccks از کامپوننت‌های زیر تشکیل شده است:
۱. کلاینت‌های وب (مرورگرها)
۲. لایه load balancer
۳. سرورهای وب (Apache/PHP)
۴. سرور پایگاه داده (MySQL)
۵. سرور فایل‌های استاتیک
۶. سرویس cache (Redis)

نمودار جریان داده ۲.۲.

جریان داده در سیستم به این صورت است:
۱. کاربر درخواست را از طریق مرورگر ارسال می‌کند.
۲. Load balancer درخواست را به یکی از سرورهای وب هدایت می‌کند.
۳. سرور وب درخواست را پردازش کرده و در صورت نیاز به پایگاه داده دسترسی پیدا می‌کند.
۴. داده‌های پرتکرار در Redis cache ذخیره می‌شوند.
۵. پاسخ به کاربر بازگردانده می‌شود.

فصل ۳: لایه‌های معماری

۳.۱. لایه Presentation (نمایش)
• تکنولوژی: HTML5, CSS3, JavaScript (Vanilla + jQuery)
• فریم‌ورک: Bootstrap 5 برای طراحی واکنش‌گرا
• کتابخانه‌های جانبی: SortableJS برای drag & drop، Toast برای نوتیفیکیشن‌ها
• ساختار فایل‌ها:
- فایل‌های PHP (ساده MVC با الگوی) برای رندر صفحات
- فایل‌های CSS در style.css
- فایل‌های JavaScript در script.js, project_decks.js, space_decks.js

۳.۲. لایه Business Logic (منطق کسب‌وکار)
• زبان: PHP 8.x
• الگوی طراحی: Model-View-Controller (MVC) به صورت ساده
• ماژول‌های اصلی:
- Authentication Module (auth.php): مدیریت احراز هویت، session، permission
- Space Management Module (spaces_manager.php): مدیریت فضاهای کاری
- Deck Management Module (project_decks.php, space_decks.php): مدیریت Decks
- Card Management Module (add_card.php, edit_card.php, delete_card.php): مدیریت کارت‌ها
- Gamification Module (add_experience.php, functions.php): مدیریت تجربه و سطح‌بندی

۳.۳. لایه Data Access (دسترسی به داده)
• پایگاه داده: MySQL 8.x
• واسط دسترسی: PDO (PHP Data Objects) با prepared statements
• ساختار: Redis برای session management و داده‌های پرتکرار cache
• محل ذخیره‌سازی فایل: Amazon S3 یا معادل برای فایل‌های آپلود شده

فصل ۴: پایگاه داده

۴.۱. طراحی پایگاه داده
پایگاه داده ProDecks شامل ۷ جدول اصلی است (همانطور که در schema.sql تعریف شده):

۱. users: اطلاعات کاربران

۲. spaces: فضاهای کاری

۳. space_members: اعضای فضاها

۴. decks: دسته‌بندی‌ها

۵. cards: کارت‌ها با قابلیت subcards

۶. projects: (برای سازگاری) پروژه‌ها

۷. user_achievements: دستاوردهای کاربران

۴.۲. ارتباط بین جداول

• هر user عضو می‌تواند multiple spaces باشد (از طریق space_members)

• هر space می‌تواند multiple decks داشته باشد

• هر deck می‌تواند multiple cards داشته باشد

• هر card می‌تواند multiple subcards داشته باشد (self-referential relationship)

۴.۳. ایندکس‌گذاری بهینه

• ایندکس روی فیلدهای پرجستجو: users(email, username), spaces(invite_code)

• ایندکس روی فیلدهای رابطه‌ای: decks(space_id), cards(deck_id, parent_card_id)

• ایندکس ترکیبی برای کوئری‌های پرتکرار

۴.۴. استراتژی backup و recovery

• backup خودکار روزانه در ساعت ۲ بامداد

• نگهداری backup های ۷ روز گذشته

• امکان بازیابی نقطه‌ای (point-in-time recovery)

فصل ۵: طراحی API

۵.۱. معماری API

• نوع: RESTful API با JSON

• احراز هویت: Token-based authentication (در آینده JWT)

• نسخه‌بندی: API/v1/

۵.۲. endpoint های اصلی

• POST /api/v1/spaces - ایجاد Space جدید

• GET /api/v1/spaces/{id} - دریافت اطلاعات Space

• POST /api/v1/decks - ایجاد Deck جدید

• GET /api/v1/decks/{id}/cards - دریافت کارت‌های یک Deck

• POST /api/v1/cards - ایجاد کارت جدید

• PUT /api/v1/cards/{id} - آپدیت کارت

• DELETE /api/v1/cards/{id} - حذف کارت

۵.۳. مدیریت خطای API

• کدهای HTTP استاندارد (۲۰۰، ۴۰۰، ۴۰۱، ۴۰۳، ۴۰۴، ۵۰۰)

• پاسخ‌های خطا با ساختار یکسان: {"error": {"code": "xxx", "message": "..."}}
• لاگ تمام درخواست‌های ناموفق برای تحلیل

فصل ۶: امنیت

۶.۱. احراز هویت (Authentication)
• استفاده از session های PHP با تنظیمات امن
• هش کردن رمز عبور با الگوریتم bcrypt
• محدودیت تلاش برای ورود (max 5 attempts)
• امکان remember me با token ایمن

۶.۲. مجوزدهی (Authorization)
• سیستم Role-Based Access Control (RBAC)
• سه سطح دسترسی: مالک، مدیر، عضو
• بررسی permission در هر عمل حساس

۶.۳. محافظت در برابر حملات رایج
• CSRF Protection: توکن CSRF در تمام فرم‌ها
• SQL Injection Prevention: استفاده از PDO prepared statements
• XSS Prevention: escaping خروجی در نمایش داده‌ها
• Clickjacking Protection: استفاده از header X-Frame-Options

۶.۴. امنیت داده‌ها
• رمزگذاری داده‌های حساس در پایگاه داده
• استفاده از HTTPS برای تمام ارتباطات
• اعتبارسنجی ورودی‌ها در سمت سرور

فصل ۷: مقیاس‌پذیری

۷.۱. استراتژی مقیاس‌گذاری
• مقیاس‌گذاری افقی سرورهای وب
• استفاده از load balancer برای توزیع ترافیک
• جدا کردن سرور پایگاه داده از سرور وب

۷.۲. استراتژی caching
• Redis برای session storage
• Cache کردن داده‌های پرخواندن و کم‌تغییر
• استفاده از HTTP caching برای فایل‌های استاتیک

۷.۳. بهینه‌سازی پایگاه داده
• استفاده از replication برای خواندن‌های زیاد
• partitioning جداول بزرگ در آینده

• مداوم indexing و query optimization

فصل ۸: مانیتورینگ و observability

۸.۱. مانیتورینگ سلامت سیستم

• بررسی uptime سرورها

• مانیتورینگ مصرف CPU، RAM، Disk

• مانیتورینگ traffic شبکه

۸.۲. مانیتورینگ کارایی

• زمان پاسخ APIها

• زمان اجرای کوئری‌های پایگاه داده

• سرعت لود صفحات

۸.۳. لاگ‌گیری

• لاگ تمام خطاهای سیستم

• لاگ فعالیت‌های مهم کاربران

• لاگ تغییرات داده‌های حساس

۸.۴. هشدارها

• هشدار برای downtime

• هشدار برای خطاهای مکرر

• هشدار برای فعالیت‌های غیرعادی

فصل ۹: تصمیم‌های معماری (Architecture Decision Records - ADRs)

۹.۱. ADR 001: انتخاب PHP به جای Node.js

• تاریخ: ۱۴۰۳/۰۸/۱۰

• وضعیت: پذیرفته شده

• زمینه: نیاز به توسعه سریع با تیم موجود

• تصمیم: استفاده از PHP به دلیل تجربه تیم و سرعت توسعه

• پیامدها: دسترسی به اکوسیستم گسترده PHP، اما performance کمتر نسبت به Node.js

۹.۲. ADR 002: انتخاب MySQL به جای NoSQL

• تاریخ: ۱۴۰۳/۰۸/۱۵

• وضعیت: پذیرفته شده

• زمینه: نیاز به داده‌های ساختاریافته با روابط پیچیده

• تصمیم: استفاده از MySQL به دلیل ACID compliance و consistency

• پیامدها: مقیاس‌پذیری عمودی آسان، اما مقیاس‌پذیری افقی سخت‌تر

۹.۳. ADR 003: استفاده از Vanilla JavaScript به جای فریم‌ورک‌های سنگین

• تاریخ: ۱۴۰۳/۰۸/۲۰

وضعیت: پذیرفته شده •
زمینه: نیاز به load time سریع و سادگی •
تصمیم: استفاده از JavaScript خالص به جای React/Vue •
پیامدها: حجم کمتر کد، اما توسعه‌پذیری کمتر در بلندمدت •

۹.۴. ADR 004: انتخاب مدل سه‌لایه به جای میکروسرویس
تاریخ: ۱۴۰۳/۰۹/۰۵ •
وضعیت: پذیرفته شده •
زمینه: مرحله MVP با تیم کوچک •
تصمیم: معماری سه‌لایه به دلیل سادگی •
پیامدها: توسعه سریع‌تر، اما پیچیدگی بیشتر در آینده •

فصل ۱۰: طرح استقرار (Deployment Plan)

۱۰.۱. محیط‌های استقرار

• محیط توسعه (Development): برای توسعه‌دهندگان
• محیط آزمایش (Staging): شبیه‌سازی production برای تست
• محیط عملیاتی (Production): برای کاربران واقعی

۱۰.۲. فرآیند استقرار

۱. توسعه در محیط development
۲. تست واحد (unit testing)
۳. استقرار در staging
۴. تست یکپارچگی (integration testing)
۵. استقرار در production با روش blue-green deployment

۱۰.۳. استقرار automation

• استفاده از script های bash برای استقرار
• استفاده از Git برای version control
• در آینده: استفاده از Docker و CI/CD pipeline

فصل ۱۱: طرح نگهداری و به‌روزرسانی

۱۱.۱. به‌روزرسانی‌های منظم

• به‌روزرسانی امنیتی: بلافاصله پس از انتشار patch
• به‌روزرسانی عملکردی: هر ۲ هفته یکبار
• به‌روزرسانی اصلی: هر ۳ ماه یکبار

۱۱.۲. مدیریت وابستگی‌ها

• استفاده از Composer برای وابستگی‌های PHP
• استفاده از CDN برای کتابخانه‌های frontend
• بررسی امنیتی وابستگی‌ها ماهانه

rollback طرح .۱۱.۳

• نگهداری backup قبل از هر استقرار

• امکان rollback سریع در صورت بروز مشکل

• حفظ compatibility با داده‌های قدیمی

فصل ۱۲: طرح توسعه آینده

مهاجرت به میکروسرویس .۱۲.۱

• پس از رسیدن به ۱۰,۰۰۰ کاربر فعال

• تفکیک سرویس‌های authentication, gamification, notification

real-time features افزودن .۱۲.۲

• استفاده از WebSocket برای collaboration real-time

• پیاده‌سازی presence system (مشاهده آنلاین کاربران)

بین‌المللی‌سازی .۱۲.۳

• پشتیبانی از زبان‌های بیشتر

• تطبیق با قوانین محلی کشورهای مختلف

بهبود معماری .۱۲.۴

• معرفی message queue برای عملیات زمان‌بر

• استفاده از CDN برای فایل‌های استاتیک در سطح جهانی

فصل ۱۳: نتیجه‌گیری

معماری فعلی ProDecks برای مرحله MVP طراحی شده و نیازهای اولیه را به خوبی پوشش می‌دهد.
این معماری به گونه‌ای انتخاب شده که در آینده بتواند به معماری پیچیده‌تر و مقیاس‌پذیرتر تبدیل شود. تصمیم‌های معماری گرفته شده بر اساس محدودیت‌های فعلی (تیم کوچک، زمان محدود، منابع کم) بوده و برای مرحله رشد مجدداً ارزیابی خواهند شد.

ضمیمه‌ها

ضمیمه الف: نمودار ERD پایگاه داده

ضمیمه ب: لیست کامل endpoint های API

ضمیمه ج: چک‌لیست امنیتی

ضمیمه د: نقشه راه فنی (Technical Roadmap)

تهیه‌کنندگان:

حامد کوهی (معمار فنی)

تیم توسعه ProDecks

مشاوران فنی (DeepSeek, Claude, Copilot)