predev. Academy

خريطة المسار

تطوير الـ Backend باستخدام PHP و Laravel

تبني REST API حقيقي مع لوحة تحكم Filament، وتكتب له اختبارات وتوثيقًا، وترفعه على سيرفر باستخدام Laravel 13 و PHP 8.4.

Web Backend

المدة	المحاضرات	المواعيد	طريقة الحضور	المستوى	اللغة
10 أسابيع	20 محاضرة مباشرة، 60 ساعة	مرتين في الأسبوع، 3 ساعات لكل محاضرة، مع ساعة office hours أونلاين كل أسبوع	حضوري أو أونلاين، لايف	مبتدئ، بعد Tech Compass	عامية مصرية، والمصطلحات التقنية بالإنجليزية

لمن هذا المسار

- طلاب حاسبات ونظم معلومات يريدون مهارة backend يشتغلون بها في وظيفة أو فريلانس
- خريجون جدد يعرفون أساسيات البرمجة لكن لم يبنوا API حقيقيًا ويرفعوه من قبل
- من يغيّرون مسارهم المهني وأنهوا Tech Compass واختاروا الـ web backend
- مطورو frontend أو mobile يريدون كتابة الـ API الذي تتعامل معه تطبيقاتهم

قبل اليوم الأول

- إنهاء Tech Compass، أو معرفة أساسية بالمتغيرات والشروط والـ loops في أي لغة
- لابتوب برام 8 GB على الأقل (16 GB أفضل)، ومساحة فارغة 20 GB، بنظام Windows 11 أو macOS أو Linux
- قبل أول يوم تكون مثبتًا: VS Code أو PhpStorm، و Git، و PHP 8.4 مع Composer (عن طريق Laravel Herd على Windows و macOS، أو php.new على Linux)، وبرنامج لقواعد البيانات مثل TablePlus أو DBeaver
- حسابات: GitHub، وأداة API مجانية مثل Postman أو Bruno
- قراءة documentation بسيطة بالإنجليزية، ولا مانع من استخدام مترجم

في نهاية المسار تستطيع

- تشرح ما يحدث أثناء الـ HTTP request وتختار الـ method والـ status code المناسبين لكل endpoint
- تكتب PHP 8.4 نظيفًا وtyped باستخدام classes و interfaces و traits و enums و namespaces، وتديره بـ Composer
- تصمم قاعدة بيانات علائقية، وتكتب SQL joins و aggregates، وتحوّل التصميم إلى Laravel migrations و Eloquent relations
- تبني REST API بإصدارات مع Form Request validation و API Resources و pagination و filtering وشكل موحد للأخطاء
- تؤمّن الـ API بـ Sanctum tokens و policies و rate limits، وتتجنب الثغرات المشهورة مثل mass assignment وضعف الـ authorization
- تنقل الشغل البطيء إلى Redis queues، وترسل mail و notifications، وتخزّن الملفات المرفوعة، وتجدول المهام المتكررة
- تكتب Pest feature tests لكل endpoint مهم ولكل قاعدة عمل أساسية

- تنشر توثيق OpenAPI وتبني Filament admin panel لفريق التشغيل
- تستخدم Claude Code أو Cursor لتسريع الشغل، ثم تراجع الناتج وتختبره قبل الـ merge
- ترفع تطبيق Laravel باستخدام Docker و Coolify على VPS مع Nginx و SSL و workers و scheduler و backups لقاعدة البيانات

التقنيات المستخدمة

اللغة والـ FRAMEWORK

PHP 8.4	لغة typed وobject oriented، وما زالت وراء جزء كبير من الويب ومن شركات كثيرة في مصر
Laravel 13	أكثر framework مستخدم في PHP، وفيه routing و ORM و queues و mail و testing جاهزة
Composer	يثبّت الـ packages ويعمل autoload للـ classes، وكل مشروع PHP يستخدمه

الـ API والـ AUTH

Laravel Sanctum	تسجيل دخول بالـ tokens لتطبيقات الموبايل والـ SPA بدون تعقيد OAuth
Form Requests و API Resources	يفصلان شكل الـ response والـ validation عن الـ controllers
Scramble باستخدام OpenAPI	يولّد توثيق الـ API من الكود نفسه ليستخدمه فريق الـ frontend والموبايل

الأدوات والـ DEPLOY

Git و GitHub	إدارة الإصدارات، والـ pull requests، والـ portfolio العام الخاص بك
Docker	يشغّل نفس البيئة على اللابتوب وعلى السيرفر
Coolify على VPS	deploy على سيرفر خاص بك مع Nginx و SSL وإدارة الـ env والـ backups، بنفس طريقة الفرق الصغيرة في مصر
Claude Code أو Cursor	مساعدات برمجة بالذكاء الاصطناعي، نستخدمها مع اختبارات ومراجعة للكود

البيانات

PostgreSQL 17 أو MySQL 8	قواعد البيانات العلائقية التي ستقابلها في أغلب وظائف Laravel
Eloquent ORM	يربط الجداول بـ models وrelations في PHP فتبقى الـ queries سهلة القراءة
Redis	يشغّل الـ queues والـ cache في تطبيقات Laravel على الـ production

لوحة التحكم والاختبارات

Filament 5	يبني admin panel كاملة فوق الـ models في ساعات
Pest 4	feature tests سهلة القراءة تحمي الـ API من الأخطاء الراجعة

مشروع المسار: Hagz: حجز، API للحجوزات مع لوحة تحكم

Hagz هو backend للحجوزات في العيادات والصالونات وملاعب البادل: العميل يبحث عن مكان، ويختار خدمة وموعدًا، ثم يحجز ويدفع، وصاحب المكان والأدمن يديرون كل شيء من لوحة Filament. المشروع يشبه أنظمة الحجز والمواعيد التي تبنيها predev. Solutions لعملائها، ويغطي كل موضوعات Laravel الأساسية: الـ relations، وأدوار المستخدمين، وقواعد العمل، والـ queues، والإشعارات، والدفع.

- تسجيل حساب ودخول العملاء بـ Sanctum tokens، مع أدوار للعميل وصاحب المكان والأدمن
- الأماكن والفروع والخدمات والموظفون ومواعيد العمل والـ time slots المولّدة تلقائيًا

- بحث وفلترة الأماكن حسب المنطقة والتصنيف والسعر والمواعيد المتاحة، مع pagination
- خطوات الحجز مع منع الحجز المزدوج باستخدام database transactions و row locks
- دورة حياة حالة الحجز (pending و confirmed و cancelled و completed) باستخدام PHP enum
- الدفع واستقبال الـ webhook من بوابة دفع تجريبية مثل Paymob
- إشعارات بالبريد وداخل التطبيق لتأكيد الحجز وإلغائه، تُرسل عبر Redis queues
- مهمة مجدولة ترسل تذكيرًا قبل كل حجز بيوم
- رفع صور الأماكن على storage محلي أو متوافق مع S3
- تقييمات ومراجعات لكل مكان، مع متوسط تقييم محفوظ في الـ cache
- API بإصدارات (/api/v1) مع توثيق OpenAPI يولّده Scramble
- لوحة Filament لإدارة الأماكن والحجوزات والمستخدمين مع dashboard للإيرادات
- مجموعة Pest feature tests تغطي الـ auth وقواعد الحجز والصلاحيات
- نسخة live على VPS عبر Coolify مع HTTPS و queue worker و scheduler و backups يومية

مراحل المسار

الأسابيع 1-3
الأساسيات
تفهم طريقة عمل الويب، وتكتب PHP بأسلوب object oriented، وتستخدم Git و Composer، وتصمم قاعدة بيانات علائقية وتكتب عليها queries.

الأسابيع 4-7
الـ Framework الأساسي
نبني Hagz bookings API ولوحة Filament الخاصة به خطوة بخطوة باستخدام Laravel 13، مع Pest tests تغطيه.

الأسابيع 8
البناء باستخدام الذكاء الاصطناعي
تستخدم Claude Code أو Cursor لتسريع العمل على كود حقيقي مع الحفاظ على الجودة عن طريق المواصفات والاختبارات والمراجعة.

الأسابيع 9
مشروع التخرج
تبني backend خاصًا بك من brief وترفعه live على VPS.

الأسابيع 10
أسبوع التوظيف
تعرض شغلك وتستعد لأول وظيفة backend أو أول عميل فريلانس.

الخطة أسبوعًا بأسبوع

الأساسيات · الأسابيع 1-3

الأسبوع 1 كيف يعمل الويب وأول خطوات في PHP

المحاضرة 1 الـ HTTP والـ APIs وتجهيز بيئة العمل

- الـ client والـ server، والـ DNS، والـ IP addresses، والـ ports بكلام بسيط
- مكونات الـ HTTP request والـ response: الـ method والـ URL والـ headers والـ body
- الـ HTTP methods (GET و POST و PUT و PATCH و DELETE) ومتى نستخدم كل واحدة
- الـ status codes: 2xx و 3xx و 4xx و 5xx، والأكواد التي ستستخدمها يوميًا
- JSON كلغة الـ APIs، وما الذي يجعل الـ API بأسلوب REST
- مراجعة تجهيز الأدوات: PHP 8.4 و Composer و extensions الـ VS Code وأساسيات الـ terminal

تطبيق عملي في المحاضرة: نفتح تبويب Network في DevTools على موقع حقيقي ونقرأ خمسة requests. بعدها نستدعي API عام باستخدام curl و Bruno أو Postman، ونكتب سكريبت PHP من 15 سطرًا يعمل على الـ built in server ويرجع JSON بالـ Content-Type والـ status code الصحيحين.

المحاضرة 2 أساسيات PHP 8.4 وحل المشكلات

- المتغيرات والـ scalar types والـ strings و declare(strict_types=1)
- الـ arrays (العادية والـ associative) وأكثر الـ array functions استخدامًا
- الشروط والـ loops و match expression
- الـ functions مع typed parameters و return types، والـ default والـ named arguments
- التعامل مع null: الـ nullable types والـ null coalescing والـ nullsafe operator
- طريقة لحل المشكلات: اقرأ، أعد الصياغة، مثال، خطة، كود، اختبار

تطبيق عملي في المحاضرة: نحل ست مسائل تمهيدية بـ PHP (نسخة من FizzBuzz، عدّاد تكرار الكلمات، التحقق من رقم موبايل مصري، حساب السعر مع ضريبة القيمة المضافة، تجميع عناصر array، فحص palindrome)، ولكل مسألة مجموعة صغيرة من حالات الاختبار اليدوية.

الواجب: تنهي المسائل الست، ثم تحل خمس مسائل سهلة على موقع مثل مسار PHP في Exercism. وتكتب ملاحظة قصيرة فيها كل status code ظهر لك في تمرين تبويب Network ومعنى كل واحد.

الأسبوع 2 البرمجة الكائنية في PHP و Composer و Git

المحاضرة 1 الـ OOP في PHP الحديثة

- الـ classes والـ objects و constructor property promotion والـ readonly properties
- الـ visibility، ومنها asymmetric visibility و property hooks في PHP 8.4
- الـ interfaces والـ abstract classes ومتى يناسب كل منهما
- الـ traits لمشاركة السلوك بين الـ classes
- الـ backed enums مع methods، ونستخدمها للحالات والأدوار
- الـ exceptions: الـ throw والـ catch وكتابة exception classes خاصة بك

تطبيق عملي في المحاضرة: نبني نموذجًا صغيرًا للحجز بـ PHP فقط: classes لـ Venue و Service و TimeSlot و Booking، و enum اسمه BookingStatus، و interface اسمه PaymentGateway مع تنفيذ FakeGateway، و DoubleBookingException تظهر عند تداخل حجزين.

المحاضرة 2 الـ namespaces و Composer و PSR وطريقة العمل بـ Git

- الـ namespaces وجمل use
- Composer: ملف composer.json، والأمر require، والـ autoload بـ PSR-4، وملف composer.lock
- معايير PHP: PSR-4 للـ autoloading و PSR-12 لأسلوب الكود، وتطبيقها بـ Laravel Pint
- أساسيات Git: init و add و commit و log و diff والـ branches والـ merge
- GitHub: الـ remote repos والـ pull requests وتعليقات مراجعة الكود وكتابة README مفيد
- كتابة commit messages واضحة

تطبيق عملي في المحاضرة: نحوّل نموذج الحجز من يوم الجمعة إلى مشروع Composer بـ PSR-4 autoloading، ونضيف package مثل nesbot/carbon للتواريخ، وننسّق الكود بـ Pint، ونرفعه على GitHub، ونفتح pull request يراجعه زميل.

الواجب: تضيف class لـ Customer و class لـ Coupon مع enum لنوع الخصم، على feature branch، وتعمل لها merge عن طريق pull request بعد المراجعة. وتحل أربع مسائل تستخدم الـ classes والـ arrays.

الأسبوع 3 قواعد البيانات العلائقية و SQL

المحاضرة 1 تصميم قواعد البيانات

- الجداول والصفوف والأعمدة وأنواع البيانات في MySQL و PostgreSQL
- الـ primary keys والـ foreign keys وقيود unique و not null
- علاقات one to many و many to many والـ pivot tables
- الـ normalization حتى الـ third normal form، ومتى نخالفها
- الـ indexes: ماذا تسرّع وما تكلفتها
- رسم ERD قبل كتابة الكود

تطبيق عملي في المحاضرة: في مجموعات، نصمم قاعدة بيانات Hagz كـ ERD (users و venues و branches و services و staff و time slots و bookings و payments و reviews)، ثم ننشئها بـ SQL DDL على MySQL 8 أو PostgreSQL 17 ونضيف بيانات تجريبية.

المحاضرة 2 الاستعلام عن البيانات والتعامل معها من PHP

- SELECT مع WHERE و ORDER BY و LIMIT
- INNER JOIN و LEFT JOIN على أكثر من جدول
- GROUP BY و HAVING والـ aggregates مثل COUNT و SUM و AVG
- الـ transactions ولماذا يجب أن يكون حجز الموعد عملية واحدة لا تتجزأ
- PDO مع prepared statements، وكيف يحدث SQL injection
- قراءة خطة تنفيذ الـ query باستخدام EXPLAIN

تطبيق عملي في المحاضرة: نكتب 12 SQL query على قاعدة بيانات Hagz (عدد الحجوزات لكل مكان هذا الشهر، أعلى الأماكن تقييمًا، المواعيد المتاحة لخدمة في يوم معين، الإيراد لكل منطقة). ثم نكتب سكريبت PHP بـ PDO يحجز موعدًا داخل transaction ويرفض حجزًا ثانيًا لنفس الموعد.

الواجب: تضيف جدول reviews وتكتب خمس queries إضافية، منها واحدة تحتاج subquery. وتشرح في README قصير هجوم SQL injection على query غير آمنة عمدًا، ثم تصلحها بـ prepared statement.

الأسبوع 4 أساسيات Laravel و Eloquent

المحاضرة 1 هيكل مشروع Laravel 13 والـ routing والـ controllers

- إنشاء مشروع بالـ Laravel installer وفهم هيكل الفولدرات
- دورة حياة الـ request: public/index.php ثم الـ HTTP kernel والـ middleware والـ router والـ controller
- أوامر Artisan التي ستستخدمها يوميًا
- الـ routes والـ route parameters والـ named routes والـ groups وملف api.php
- الـ controllers والـ single action controllers والـ resource controllers
- الـ configuration وملف .env، ولماذا لا تدخل الأسرار في Git أبدًا

تطبيق عملي في المحاضرة: ننشئ repository مشروع Hagz، ونضيف health check endpoint، ونبني endpoints للقراءة فقط للأماكن والخدمات ترجع JSON ثابتًا من خلال controllers تحت /api/v1.

المحاضرة 2 الـ migrations والـ models والـ relations والـ factories والـ seeders

- الـ migrations: إنشاء الجداول وتعديلها والـ foreign keys والـ indexes
- الـ Eloquent models والـ fillable attributes والـ casts (ومنها enum casts)
- الـ relations: belongsTo و hasMany و belongsToMany مع بيانات الـ pivot و hasManyThrough
- الـ query scopes والـ eager loading لتجنب مشكلة N+1
- الـ factories مع الـ states، والـ seeders لبيانات محلية واقعية
- الـ soft deletes والـ timestamps

تطبيق عملي في المحاضرة: نحوّل الـ ERD من الأسبوع 3 إلى Laravel migrations و models بكل الـ relations، ونكتب factories و seeder يملأ قاعدة البيانات بأماكن في مناطق مثل مدينة نصر ومصر الجديدة والمعادي والشيخ زايد، ونغيّر endpoints الأماكن لتستخدم Eloquent queries حقيقية.

الواجب: تضيف جداول staff ومواعيد العمل مع الـ relations الخاصة بها، وأمر Artisan يولّد الـ time slots لأول 14 يومًا قادمة من مواعيد العمل. وتستخدم Laravel Debugbar أو الـ query log لتثبت عدم وجود N+1 queries في قائمة الأماكن.

مرحلة في المشروع: مشروع Hagz على GitHub بقاعدة البيانات كاملة والـ relations والبيانات التجريبية، و endpoints قراءة تعمل للأماكن والخدمات.

الأسبوع 5 تصميم الـ API والـ validation والـ authentication

المحاضرة 1 تصميم REST API باستخدام Form Requests و API Resources

- قواعد REST: تسمية الـ resources والتداخل بينها والإصدارات بـ api/v1/
- الـ Form Requests للـ validation وللتحقق من صلاحية الإدخال
- الـ API Resources والـ Resource Collections للتحكم في شكل الـ response
- الـ pagination والـ filtering والـ sorting عن طريق الـ query parameters
- شكل موحد للأخطاء والتعامل مع الـ exceptions في bootstrap/app.php
- الـ route model binding

تطبيق عملي في المحاضرة: نبني CRUD كاملًا للأماكن والخدمات، و endpoint بحث عام يفلتر حسب المنطقة والتصنيف ونطاق السعر والمواعيد المتاحة، ويرجع API Resources بـ pagination، ويرجع أخطاء 422 بشكل موحد.

المحاضرة 2 الـ authentication والـ authorization باستخدام Sanctum والـ policies

- الـ hashing لكلمات المرور ولماذا لا نحفظها كنص عادي أبدًا
- الـ Sanctum API tokens: التسجيل والدخول والخروج والـ token abilities
- الـ middleware المسمى auth:sanctum وكتابة middleware خاص بك
- أدوار العميل وصاحب المكان والأدمن باستخدام enum
- الـ gates والـ policies للتحكم في الصلاحيات على مستوى كل سجل
- الـ rate limiting على تسجيل الدخول والـ endpoints العامة

تطبيق عملي في المحاضرة: نضيف endpoints للتسجيل والدخول والخروج والملف الشخصي باستخدام Sanctum. ونكتب policies بحيث لا يعدّل المكان إلا صاحبه، ولا يرى الحجز إلا العميل صاحب الحجز أو صاحب المكان، ثم نحاول كسرها بـ requests من مستخدم غير مسموح له.

الواجب: تضيف استعادة كلمة المرور بالبريد (تُكتب في الـ log driver حاليًا)، و endpoint لصاحب المكان يعرض أماكنه هو فقط. وتكتب قواعد الـ API الخاصة بـ Hagz في الـ README: شكل الروابط، وشكل الأخطاء، وشكل الـ pagination.

مرحلة في المشروع: API عليه authentication و validation وإصدارات، مع أدوار و policies. صاحب المكان يدير أماكنه وخدماته فقط.

الأسبوع 6 قواعد العمل والاختبارات والشغل في الخلفية

المحاضرة 1 خطوات الحجز و feature tests باستخدام Pest

- وضع منطق العمل في action أو service classes بعيدًا عن الـ controllers
- منع الحجز المزدوج باستخدام database transactions و lockForUpdate
- انتقالات حالة الحجز باستخدام enum و guard methods
- تجهيز Pest وهيكل الاختبار و RefreshDatabase
- الـ feature tests مع الـ factories و Sanctum::actingAs و JSON assertions
- اختبار الحالات الفاشلة: 401 و 403 و 404 و 422

تطبيق عملي في المحاضرة: نبني endpoints إنشاء الحجز وتأكيده وإلغائه باستخدام action اسمه CreateBooking. ونكتب Pest feature tests أولًا للقواعد الأساسية: لا يمكن حجز نفس الموعد مرتين، ولا يلغي عميل حجز عميل آخر، ولا يُسمح بالإلغاء قبل الموعد بأقل من ساعتين.

المحاضرة 2 الـ queues والـ mail والـ notifications والـ storage والـ scheduling

- لماذا يذهب الشغل البطيء إلى queue، وتشغيل Redis محليًا
- الـ jobs والـ dispatching والـ retries والـ failed jobs والـ queue worker
- الـ Mailables و Markdown mail templates
- الـ notifications عبر البريد والـ database channel
- تخزين الملفات: disks محلية ومتوافقة مع S3، والتحقق من الملفات المرفوعة وحفظها
- الـ task scheduling للمهام المتكررة

تطبيق عملي في المحاضرة: نرسل بريد تأكيد الحجز و notification داخل قاعدة البيانات عبر Redis queue، ونضيف رفع صور الأماكن مع validation وتخزين، ونجدول job يومية ترسل تذكيرًا لكل حجز موعده غدًا. ونستخدم Mailpit لرؤية الرسائل.

الواجب: تضيف الدفع عبر بوابة تجريبية (Paymob في test mode أو fake gateway) مع webhook endpoint يتحقق من الـ signature ويحوّل الحجز إلى مدفوع. وتغطيه بأربعة Pest tests على الأقل، مع استخدام Http::fake للبوابة.

مرحلة في المشروع: خطوات حجز كاملة مع الدفع، وإشعارات عبر الـ queue، ورفع صور، وتذكيرات مجدولة، وكل Pest tests ناجحة.

الأسبوع 7 الأداء وتوثيق الـ API ولوحة التحكم

المحاضرة 1 الـ caching والأداء وتوثيق OpenAPI

- اكتشاف الـ queries البطيئة ومشكلات N+1 وإضافة الـ index المناسب
- الـ caching باستخدام Redis: الدالة remember، وأسماء الـ cache keys، وإبطال الـ cache مع أحداث الـ model
- الـ rate limiters لكل مستخدم ولكل IP
- قائمة مراجعة الأمان: mass assignment، والـ authorization على كل route، والـ CORS، والأسرار
- توليد توثيق OpenAPI باستخدام Scramble والحفاظ على دقته
- كتابة Postman أو Bruno collection لفريق الـ frontend والموبايل

تطبيق عملي في المحاضرة: نقيس أداء البحث عن الأماكن، ونصلح الأجزاء البطيئة بالـ eager loading والـ indexes، ونحفظ تقييمات الأماكن في الـ cache ونبطلها عند إضافة مراجعة، وننشر توثيق الـ API على /docs/api مع أمثلة لكل endpoint.

المحاضرة 2 لوحة التحكم باستخدام Filament

- تثبيت Filament panel وقصرها على الأدمن فقط
- الـ resources: الـ forms والـ tables والـ columns والـ filters والـ actions
- الـ relation managers لخدمات المكان وحجوزاته
- actions مخصصة مثل تأكيد الحجز أو استرداد الدفع
- الـ dashboard widgets بالأرقام والرسوم البيانية
- إعادة استخدام الـ policies لتلتزم اللوحة بنفس صلاحيات الـ API

تطبيق عملي في المحاضرة: نبني لوحة تحكم Hagz: إدارة المستخدمين والأماكن والخدمات والحجوزات، وفلترة الحجوزات حسب الحالة والتاريخ، و action للتأكيد والإلغاء يستخدم نفس الـ booking action class، و dashboard تعرض حجوزات اليوم وإيراد الشهر.

الواجب: تضيف المراجعات (بعد اكتمال الحجز فقط) مع endpoints و tests و Filament resource لمراجعتها من الأدمن. وتنظّم الـ README بحيث يستطيع مطور جديد أن يعمل clone و seed ويشغّل Hagz في أقل من 10 دقائق.

مرحلة في المشروع: Hagz مكتمل: API موثّق، ولوحة Filament، و caching، و tests لكل endpoint أساسي.

الأسبوع 8 البناء باستخدام الذكاء الاصطناعي بالطريقة الصحيحة

المحاضرة 1 مساعدات الذكاء الاصطناعي على مشروع Laravel حقيقي

- كيف تعمل مساعدات البرمجة ولماذا تخطئ وهي واثقة
- تجهيز Claude Code أو Cursor، مع ملف قواعد للمشروع (CLAUDE.md أو Cursor rules) يشرح أسلوب Hagz
- إعطاء المساعد سياق Laravel، مثلًا باستخدام Laravel Boost MCP server
- كتابة مواصفات قصيرة و acceptance tests قبل طلب الكود
- العمل بخطوات صغيرة على branch، و diff واحد قابل للمراجعة في كل مرة
- استخدام الذكاء الاصطناعي لشرح كود غير مألوف ورسائل الأخطاء

تطبيق عملي في المحاضرة: نضيف خاصية الكوبونات إلى Hagz بمساعدة الذكاء الاصطناعي: تكتب أنت المواصفات و Pest tests، ويكتب المساعد الـ migration والـ model والـ endpoints و Filament resource، ثم تشغّل الاختبارات وتصلح ما يفشل حتى تنجح كلها.

المحاضرة 2 مراجعة ناتج الذكاء الاصطناعي ومعرفة متى لا تثق فيه

- قائمة مراجعة لكود Laravel المولّد: الـ validation والـ authorization والـ mass assignment و N+1 والـ transactions
- اكتشاف methods مخترعة، وصياغة قديمة، و packages غير موجودة
- الاختبارات كشبكة أمان: ما الذي تكتشفه مجموعة الاختبارات وما الذي يفوتها
- مناطق تحتاج حذرًا: الـ auth والدفع والـ migrations على بيانات حقيقية وأي شيء يحذف بيانات
- إبعاد الأسرار وبيانات العملاء عن الـ prompts
- أن تكون قادرًا على شرح كل سطر تعمل له commit

تطبيق عملي في المحاضرة: نراجع pull request كتبه الذكاء الاصطناعي وفيه أخطاء مقصودة (policy check ناقص، وثغرة mass assignment، و N+1 query، وحجز غير atomic). تكتشف كل خطأ وتصلحه، وتضيف test كان سيكتشفه، وتكتب ملخص مراجعة قصيرًا.

الواجب: تختار فكرة الـ capstone، وتكتب مواصفات في صفحة واحدة (المستخدمون، والـ endpoints، ونموذج البيانات، وما هو خارج النطاق) و ERD، وتأخذ الموافقة عليها في الـ office hours. وتسجّل موقفًا واحدًا هذا الأسبوع أخطأ فيه الذكاء الاصطناعي وكيف اكتشفته.

مرحلة في المشروع: خاصية الكوبونات تدخل Hagz عبر pull request بعد المراجعة، مكتوبة بمساعدة الذكاء الاصطناعي ومغطاة باختبارات كتبتها أنت.

الأسبوع 9 بناء مشروع التخرج والـ deploy الحقيقي

المحاضرة 1 Docker والـ deploy باستخدام Coolify

- أساسيات Docker: الـ images والـ containers والـ volumes و Dockerfile لمشروع Laravel
- Docker Compose لتشغيل MySQL أو PostgreSQL و Redis و Mailpit محليًا
- Coolify على VPS: ربط GitHub، والـ build، والـ deploy مع كل push
- Nginx و PHP-FPM أمام Laravel، وملف .env الخاص بالـ production
- SSL باستخدام Let's Encrypt، و domain خاص، وتشغيل الـ migrations بأمان
- تشغيل الـ queue worker والـ scheduler على الـ production، والـ logs، و backups يومية لقاعدة البيانات

تطبيق عملي في المحاضرة: نرفع Hagz على VPS الكورس عبر Coolify مع domain و HTTPS، وخدمة Redis، و queue worker، والـ scheduler، و backups مجدولة. ثم تبدأ repository مشروع التخرج من المواصفات المعتمدة.

المحاضرة 2 يوم بناء مشروع التخرج

- تحويل المواصفات إلى قائمة مهام وتقليل النطاق ليناسب الأسبوع
- قاعدة البيانات والـ auth والـ endpoints الأساسية أولًا
- اختبارات لقاعدة العمل الرئيسية قبل التحسينات الشكلية
- مراجعة كود مع المدرب في منتصف اليوم
- رفع مشروع التخرج عبر Coolify
- كتابة الـ README وتوثيق الـ API

تطبيق عملي في المحاضرة: تبني الجزء الأساسي من مشروع التخرج في المحاضرة مع مراجعة من المدرب كل ساعة، وترفع أول نسخة تعمل على Coolify app خاص بك مع HTTPS قبل نهاية المحاضرة.

الواجب: تنهي مشروع التخرج: كل المتطلبات الأساسية، و tests ناجحة، وتوثيق OpenAPI، ولوحة Filament، ورابط live بـ HTTPS. وتجهّز عرضًا مدته 5 دقائق ليوم العرض.

مرحلة في المشروع: مشروع التخرج live على VPS مع HTTPS و tests وتوثيق ولوحة تحكم.

الأسبوع 10 الاستعداد لسوق العمل ويوم العرض

المحاضرة 1 الـ CV و LinkedIn و GitHub portfolio والفريلانس

- CV من صفحة واحدة لمطور backend مبني على المشروعات والنتائج
- بروفايل LinkedIn والـ headline والنشر عن ما بنيته
- GitHub portfolio: الـ repos المثبتة، و READMEs فيها روابط live وتوثيق API وصور
- كيف يتم التوظيف في Laravel في مصر والخليج والعمل عن بعد
- أول خطوات الفريلانس على Upwork و مستقل و خمسات: البروفايل وأول عروض وتحديد نطاق الشغل
- تقدير وتسعير شغل صغير لـ API أو لوحة تحكم

تطبيق عملي في المحاضرة: تعيد كتابة الـ CV وبروفايل LinkedIn مع ملاحظات المدرب، وتحسّن README مشروع Hagz ومشروع التخرج، وتكتب عرضًا تجريبيًا على إعلان شغل Laravel حقيقي.

المحاضرة 2 مقابلات تجريبية ويوم العرض

- أسئلة مقابلات PHP و Laravel الشائعة: دورة حياة الـ request و Eloquent والـ queues والـ service container
- أسئلة SQL وتصميم قواعد البيانات
- مسألة live coding تحلها وأنت تشرح بصوت عالٍ
- شرح قرارات التصميم في مشروعك والمفاضلة بين الخيارات
- يوم العرض: عرض مشروع التخرج في 5 دقائق أمام ضيوف وشركاء توظيف
- ملاحظات وخطة شخصية للأشهر الثلاثة القادمة

تطبيق عملي في المحاضرة: كل طالب يخوض مقابلة تقنية تجريبية مدتها 20 دقيقة مع ملاحظات، ثم يعرض مشروع التخرج live في يوم العرض أمام ضيوف من الصناعة.

الواجب: تقدّم على خمس وظائف Laravel أو مشروعات فريلانس على الأقل خلال الأسبوعين القادمين، وتعمل بملاحظات المقابلة، وتنشر بوست عن مشروع التخرج على LinkedIn.

مشروع التخرج

كل طالب يبني ويرفع backend خاصًا به من brief: REST API لحالة استخدام محلية حقيقية، فيه authentication، وقاعدة عمل واحدة على الأقل غير بسيطة، وشغل في الخلفية، ولوحة Filament، و Pest tests، وتوثيق OpenAPI، ونسخة live على VPS عبر Coolify.

المتطلبات

- مواصفات من صفحة واحدة و ERD معتمدان قبل بدء الكود
- 5 جداول مترابطة على الأقل مع migrations و factories و seeders
- Sanctum authentication بدورين على الأقل و policies على كل resource محمي
- Form Request validation و API Resources على كل endpoint، تحت /api/v1
- قاعدة عمل واحدة على الأقل محمية بـ transaction أو lock
- queued job واحدة على الأقل و notification واحدة (بريد أو database)
- لوحة Filament للـ resources الأساسية
- Pest feature tests للـ auth ولقاعدة العمل الرئيسية وللصلاحيات، وكلها ناجحة
- توثيق OpenAPI متاح من الرابط الـ live
- نسخة live على VPS عبر Coolify مع HTTPS و queue worker و scheduler شغالين
- README فيه خطوات التشغيل والرابط الـ live ورابط التوثيق وجزء قصير عن قرارات التصميم

أفكار مقترحة

- backend لطلبات مطعم: منيو وسلة طلبات ومناطق توصيل وتتبع حالة الطلب
- منصة لسنتر دروس: مجموعات وحصص وحضور ومتابعة المصاريف الشهرية للمدرسين
- نظام اشتراكات جيم: باقات وحجز كلاسات وتسجيل حضور وتذكير بالتجديد
- API لإعلانات العقارات: بحث بالكمبوند والمنطقة، وطلبات معاينة، و dashboard للوكيل
- مخزون وطلبات صيدلية صغيرة مع تنبيهات نقص المخزون وأوامر شراء من الموردين
- حجز تذاكر فعاليات للأسر الطلابية في الجامعات مع تذاكر QR وحد أقصى للعدد وتسجيل دخول

معايير تقييم مشروع التخرج

يعمل من أوله لآخره و live على الـ VPS بـ HTTPS	25%
تصميم الـ API والـ validation وتوحيد شكل الـ responses	15%
نموذج البيانات والـ migrations واستخدام Eloquent	15%
الـ authentication والـ authorization والأمان	15%
Pest tests تغطي القواعد الأساسية	15%
جودة الكود وتاريخ Git والـ README وتوثيق الـ API	10%
العرض في يوم العرض والإجابة عن الأسئلة	5%

طريقة التقييم

الواجبات الأسبوعية تُسلَّم كـ pull request على GitHub وتُراجع كل أسبوع بقائمة مراجعة قصيرة	20%
مراحل المشروع المستمر مراحل Hagz في الأسابيع 4 إلى 7، تُراجع live في الـ office hours: الـ endpoints تعمل، والـ tests ناجحة، والكود مقروء	25%
تمرين المراجعة في أسبوع الذكاء الاصطناعي الأخطاء التي اكتشفتها وأصلحتها في الـ pull request المولّد، وجودة الاختبارات التي أضفتها	10%
مشروع التخرج يُقيَّم بمعايير مشروع التخرج	30%
المقابلة التجريبية ويوم العرض الإجابات التقنية، ووضوح شرحك لكودك، والعرض الـ live	10%
الحضور والمشاركة حضور المحاضرات، والعمل في ثنائيات، ومراجعة pull requests الزملاء	5%

قائمة المهارات

الويب و HTTP

- [ ] تقرأ وتشرح أي HTTP request و response
- [ ] تختار الـ method والـ status code المناسبين لكل endpoint
- [ ] تختبر الـ APIs باستخدام curl و Postman أو Bruno

PHP

- [ ] تكتب PHP 8.4 typed مع strict types
- [ ] تستخدم الـ classes والـ interfaces والـ traits والـ enums في مكانها الصحيح
- [ ] تدير الـ dependencies والـ autoloading باستخدام Composer
- [ ] تلتزم بأسلوب PSR-12 باستخدام Laravel Pint

قواعد البيانات

- [ ] تصمم قاعدة بيانات normalized وترسم الـ ERD الخاص بها
- [ ] تكتب joins و aggregates و subqueries
- [ ] تستخدم الـ transactions والـ indexes بشكل صحيح
- [ ] تمنع SQL injection

Laravel

- [ ] الـ routing والـ controllers والـ middleware
- [ ] الـ migrations و Eloquent relations والـ factories والـ seeders
- [ ] الـ Form Requests والـ API Resources
- [ ] Sanctum auth والـ gates والـ policies
- [ ] الـ queues والـ jobs والـ mail والـ notifications والـ storage والـ scheduling
- [ ] الـ caching باستخدام Redis وتجنب N+1 queries

الجودة

- [ ] تكتب Pest feature tests للحالات الناجحة والفاشلة
- [ ] تراجع الكود بحثًا عن مشكلات الأمان والأداء
- [ ] تستخدم مساعدات الذكاء الاصطناعي مع مواصفات واختبارات ومراجعة

التسليم

- [ ] تعمل بـ Git branches و pull requests
- [ ] تكتب توثيق OpenAPI و README واضحًا
- [ ] تبني لوحة تحكم بـ Filament
- [ ] ترفع المشروع باستخدام Docker و Coolify مع SSL و workers و backups

بعد التخرج

الأشهر 1-3 بعد التخرج

- تقدّم على وظائف junior Laravel و PHP backend والتدريبات كل أسبوع
- تأخذ شغلًا أو اثنين صغيرين على مستقل أو خمسات أو Upwork لتجمع تقييمات
- تضيف مشروعًا آخر مرفوعًا live على GitHub، ويُفضّل أن يكون لنشاط تجاري صغير حقيقي
- تقرأ أجزاء Laravel docs التي لم نغطها: الـ events والـ broadcasting و Octane

الأشهر 4-6

- تتعلم frontend للـ APIs الخاصة بك: Livewire، أو Inertia مع Vue أو React، أو تطبيق Flutter
- تتعمق في قواعد البيانات: تحسين الـ queries والـ indexes ومميزات PostgreSQL
- تضيف static analysis باستخدام Larastan و CI باستخدام GitHub Actions
- تساهم بإصلاح صغير أو تحسين في توثيق package مفتوحة المصدر في Laravel

الأشهر 7-12: نحو مستوى mid-level

- تصميم البرمجيات: SOLID، وتنظيم المشروعات الكبيرة بأسلوب domain driven، والـ design patterns
- خصائص real time باستخدام Laravel Reverb والـ broadcasting
- تشغيل الـ production: المراقبة والـ logging وتوسيع الـ queues بـ Horizon و zero downtime deploys
- مهارة backend ثانية لتوسيع خياراتك: Node.js مع NestJS، أو أساسيات Go
- توجيه طلاب أحدث منك وقيادة خاصية من المواصفات حتى الـ deploy

الوظائف التي يقود إليها المسار

Junior Backend Developer (PHP / Laravel) | Laravel Developer في شركات البرمجيات والـ startups في مصر | Junior Full Stack Developer (Laravel مع Livewire أو Vue أو React) | API Developer لفرق تطبيقات الموبايل | Remote Laravel Developer لشركات في الخليج وأوروبا وأمريكا | Freelance Laravel Developer على Upwork و مستقل و خمسات | Junior DevOps أو Deployment Engineer لفرق PHP

مصادر

- توثيق Laravel: https://laravel.com/docs
- Laravel Bootcamp: https://bootcamp.laravel.com
- Laracasts: https://laracasts.com
- دليل PHP الرسمي: https://www.php.net/manual/en/
- PHP: The Right Way: https://phptherightway.com

- توثيق Composer: https://getcomposer.org/doc/
- معايير PSR من PHP-FIG: https://www.php-fig.org/psr/
- MDN: HTTP: https://developer.mozilla.org/en-US/docs/Web/HTTP
- توثيق MySQL: https://dev.mysql.com/doc/
- توثيق PostgreSQL: https://www.postgresql.org/docs/
- توثيق Pest: https://pestphp.com/docs
- توثيق Filament: https://filamentphp.com/docs
- Scramble (Laravel J OpenAPI): https://scramble.dedoc.co
- مواصفات OpenAPI: https://spec.openapis.org/oas/latest.html
- توثيق Redis: https://redis.io/docs/latest/
- توثيق Docker: https://docs.docker.com
- توثيق Coolify: https://coolify.io/docs
- توثيق Git: https://git-scm.com/doc
- مسار PHP على Exercism: https://exercism.org/tracks/php