predev. Academy

خريطة المسار

تطوير الـ Backend باستخدام Node.js

تبني REST API جاهزة للإنتاج وتختبرها وتؤمّنها وتنشرها على خادم حقيقي باستخدام Node.js وTypeScript وNestJS وPostgreSQL.

Web Backend

المدة	الجلسات	المواعيد	المستوى	لغة الشرح	المكان
10 أسابيع	20 جلسة مباشرة، 60 ساعة	مرتين في الأسبوع، 3 ساعات لكل محاضرة، مع ساعة office hours أونلاين كل أسبوع	مبتدئ، بعد Tech Compass	العربية المصرية، مع المصطلحات التقنية بالإنجليزية	حضوري أو أونلاين، لايف

لمن هذا المسار

- طلاب علوم الحاسب ونظم المعلومات الذين يريدون العمل في الـ Backend قبل التخرج أو بعده مباشرة
- الخريجون الجدد الذين يعرفون أساسيات البرمجة لكنهم لم يسلّموا API حقيقية من قبل
- من يغيّرون مسارهم المهني وأنهوا Tech Compass واختاروا الـ Backend
- مطورو الـ Frontend أو الموبايل الذين يريدون كتابة الـ API التي تتصل بها تطبيقاتهم

قبل اليوم الأول

- إتمام Tech Compass، أو معرفة أساسيات البرمجة: المتغيرات والشروط والتكرار والدوال
- لابتوب بذاكرة 8 GB RAM على الأقل (يفضل 16 GB)، ومساحة فارغة 20 GB، ونظام Windows 10/11 مع WSL 2 أو macOS أو Linux
- حسابات جاهزة قبل أول يوم: GitHub، وتثبيت Postman أو Bruno، وحساب على أداة AI للبرمجة (Claude Code أو Cursor) للأسبوع 8
- برامج مثبتة قبل أول يوم: Node.js 24 LTS وVS Code وGit وDocker Desktop (أو Docker Engine على Linux)
- قراءة أساسية بالإنجليزية، لأن التوثيق ورسائل الأخطاء كلها بالإنجليزية

في نهاية المسار تستطيع

- تشرح ما يحدث في HTTP request من العميل حتى قاعدة البيانات ثم العودة
- تكتب كود TypeScript غير متزامن ونظيفًا، وتفهم الـ event loop في Node.js
- تصمم schema منظمة في PostgreSQL وتكتب SQL يحتوي على joins وindexes
- تبني REST API في NestJS 11 تحتوي على modules وتحقق من الـ DTOs وguards وردود أخطاء موحدة
- تصمم البيانات باستخدام Prisma، وتشغّل migrations، وتستخدم transactions للعمليات التي يجب أن تنجح معًا
- تنفذ التسجيل وتسجيل الدخول والـ refresh tokens وتشفير كلمات المرور والصلاحيات حسب الدور
- تضيف رفع الملفات إلى S3-compatible storage والـ caching باستخدام Redis والـ background jobs باستخدام BullMQ

- توثق الـ API باستخدام OpenAPI وتختبرها باختبارات unit وend to end
- تشغّل كل المكونات باستخدام docker compose وتنشرها على VPS عبر Coolify بدومين وHTTPS
- تستخدم أدوات الـ AI للبرمجة لتعمل أسرع، وتكتشف أخطاءها بالمراجعة والاختبارات

التقنيات المستخدمة

اللغة وبيئة التشغيل

JavaScript (ES2024) لغة Node.js. الكود غير المتزامن والـ modules هما أساس كل ما يأتي بعدهما.

TypeScript (strict mode) الـ types تكشف الأخطاء قبل التشغيل، وهي الاختيار الافتراضي لدى فرق الـ Backend في مصر وفي العمل عن بعد.

Node.js 24 LTS إصدار الدعم الطويل المستخدم في الإنتاج، ونستخدم منه fs وenv وstreams مباشرة.

pnpm / npm مديرو الحزم لتثبيت المكتبات وتثبيت إصداراتها وتشغيل scripts المشروع.

البيانات

PostgreSQL 17 قاعدة بيانات علائقية موثوقة تستخدمها أغلب المنتجات الجادة، ونصمم فيها الجداول ونكتب SQL يدويًا أولًا.

Prisma ORM وصول إلى قاعدة البيانات بـ types آمنة، وmigrations مبنية على الـ schema، وtransactions.

Redis 8 لـ caching وتخزين حدود الـ rate limit، وهو أساس الـ queues التي تعمل في الخلفية.

BullMQ مهام الخلفية مثل الرسائل البريدية والإشعارات والمهام المجدولة.

الجودة والتوثيق

Jest / Vitest + Supertest اختبارات unit وend to end تعمل على طبقة الـ HTTP الحقيقية.

OpenAPI / Swagger توثيق حي لـ API يقرؤه مطورو الـ Frontend والموبايل كل يوم.

Pino logging سجلات منظمة يمكن البحث فيها عندما يحدث عطل في الإنتاج.

ESLint + Prettier أسلوب كود موحد للفريق كله، يُراجع تلقائيًا.

الـ FRAMEWORKS

Express 5 صغير وواضح، فيرى الطالب كيف يعمل الـ routing والـ middleware ومعالجة الأخطاء من الداخل.

NestJS 11 الـ framework الأساسي في المسار: modules وdependency injection وهيكل واضح تستخدمه الفرق في الـ APIs الكبيرة.

الأمان والتحقق من المدخلات

JWT + refresh tokens تسجيل دخول stateless يعمل مع تطبيقات الويب والموبايل.

argon2 / bcrypt تشفير كلمات المرور بالطريقة الصحيحة.

class-validator / Zod التحقق من كل request عند حدود النظام.

helmet + rate limiting HTTP headers آمنة وحماية من محاولات التخمين وإساءة الاستخدام.

التسليم والنشر

Git + GitHub الـ branches والـ pull requests ومراجعة الكود، بنفس طريقة العمل في predev. Solutions.

Docker + docker compose البيئة نفسها على كل لابتوب وعلى الخادم.

S3-compatible storage (MinIO محليًا) تخزين الملفات المرفوعة خارج الخادم، وهي الطريقة المعتادة للتعامل مع الصور والمستندات.

VPS + Coolify نشر حقيقي بدومين وHTTPS ومتغيرات بيئة، بتكلفة مناسبة للعمل الحر في مصر.

أدوات الـ AI

Cursor أو Claude Code	كتابة الكود بسرعة أكبر بمساعدة الـ AI، مع مراجعة كل تغيير والاعتماد على الاختبارات كشبكة أمان.

مشروع المسار: Khedma API: نظام Backend لحجز الخدمات المنزلية

Khedma هو الـ Backend لتطبيق يحجز فيه العملاء خدمات منزلية (سباكة، تنظيف، صيانة تكييف) من مقدمي خدمة موثقين. يشبه المشروع أنظمة الحجز والطلبات التي تبنيها predev. Solutions لعملائها: مستخدمون بأدوار مختلفة، وخدمات معروضة، ومواعيد متاحة، وحجوزات تتغير حالتها، وسجلات دفع، ورفع ملفات، وإشعارات، وAPI للوحة الإدارة. يبنيه الطلاب خطوة بخطوة من الأسبوع 4 إلى الأسبوع 7 وينشرونه على الإنترنت.

- تسجيل حساب ودخول باستخدام JWT access tokens وrefresh tokens، وتسجيل الخروج من كل الأجهزة
- 3 أدوار: عميل، ومقدم خدمة، ومسؤول، مع guards على كل route محمي
- تصنيفات الخدمات وملفات مقدمي الخدمة مع رفع الصور إلى S3-compatible storage
- مواعيد متاحة لكل مقدم خدمة، وإنشاء حجز يمنع الحجز المزدوج داخل transaction
- مراحل الحجز: pending وconfirmed وin progress وcompleted وcancelled، مع قواعد لكل دور
- تقييمات ومراجعات بعد اكتمال الحجز
- بحث وفلترة وترتيب وpagination على الخدمات ومقدمي الخدمة
- Redis caching للقوائم العامة وrate limits على routes تسجيل الدخول
- BullMQ jobs لرسائل تأكيد الحجز وتذكير المواعيد
- تحديثات لحظية لحالة الحجز عبر WebSockets
- endpoints للمسؤول: إدارة المستخدمين، واعتماد مقدمي الخدمة، وإحصائيات أساسية
- توثيق OpenAPI على /docs، وendpoint لفحص الحالة، وسجلات منظمة
- اختبارات unit وend to end، وDockerfile وdocker compose، ونشر على VPS عبر Coolify

مراحل المسار

الأسابيع 1-3
الأساسيات
تفهم كيف يعمل الويب، وتكتب TypeScript سليمًا على Node.js، مع Git وSQL وعادات حل المشكلات.

الأسابيع 4-7
الـ Framework الأساسي
تبني Khedma API باستخدام NestJS 11 مع Prisma والـ auth ورفع الملفات والـ caching والـ queues والاختبارات وDocker.

الأسابيع 8
البناء بالـ AI
تستخدم أدوات الـ AI للبرمجة لتعمل أسرع على كود حقيقي، مع الحفاظ على الجودة بالمراجعة والاختبارات.

الأسابيع 9
مشروع التخرج
يبني كل طالب API خاصة به جاهزة للإنتاج من وصف مشروع، وينشرها.

الأسابيع 10
أسبوع التوظيف
تحويل عمل البوتكامب إلى وظيفة أو أول عميل في العمل الحر.

الخطة أسبوعًا بأسبوع

الأساسيات · الأسابيع 1-3

الأسبوع 1 كيف يعمل الويب، وJavaScript وGit

المحاضرة 1 **HTTP وJSON ودورة حياة الـ request**

- العميل والخادم والـ DNS والـ IP والـ ports: ماذا يحدث عندما تفتح URL
- HTTP methods والـ headers والـ body والـ status codes (2xx وَ3xx وَ4xx وَ5xx)
- صيغة JSON وأساسيات REST: الموارد والـ URLs والأفعال
- فحص الـ requests باستخدام DevTools في المتصفح وcurl وPostman أو Bruno
- تجهيز بيئة التطوير: Node.js 24 LTS وVS Code وأساسيات الـ terminal

تطبيق عملي في المحاضرة: تتصل بـ API عامة باستخدام curl وPostman، وتقرأ كل header وstatus code، ثم تكتب ملاحظة قصيرة تشرح request وresponse كاملين.

المحاضرة 2 **أساسيات JavaScript وGit**

- المتغيرات والأنواع والدوال والـ arrow functions والـ scope
- الـ arrays والـ objects: map وfilter وreduce والـ destructuring والـ spread
- منهج حل المشكلات: افهم، خطط، تقدّم بخطوات صغيرة، اختبر بأمثلة
- أساسيات Git: init وadd وcommit وlog وdiff وgitignore.
- GitHub: إنشاء repo، وعمل push، وكتابة README جيد

تطبيق عملي في المحاضرة: تحل 6 مسائل صغيرة على الـ arrays والـ objects باستخدام JavaScript، وتعمل commit بعد كل مسألة برسالة واضحة، ثم ترفع الـ repo إلى GitHub.

الواجب: أكمل 10 مسائل JavaScript (strings وarrays وobjects) في الـ repo الخاص بك بـ commit لكل مسألة، واكتب README يشرح طريقة تشغيلها.

الأسبوع 2 JavaScript غير المتزامن وبيئة Node.js وTypeScript

المحاضرة 1 الكود غير المتزامن وبيئة تشغيل Node.js

- الـ callbacks والـ promises والـ async/await، ومعالجة الأخطاء باستخدام try/catch
- الـ event loop: الـ call stack والـ task queue والـ microtasks، ولماذا يضر الكود الذي يوقف التنفيذ
- ES modules: import وexport وخاصية type في package.json والـ scripts
- الـ core modules: fs/promises وpath وprocess.env وملفات env.
- أساسيات الـ streams: قراءة ملف كبير دون تحميله كاملًا في الذاكرة
- npm وpnpm: التثبيت وملفات الـ lock والـ semantic versioning

تطبيق عملي في المحاضرة: تبني أداة CLI تقرأ ملف CSV للطلبات باستخدام streams، وتحسب الإجمالي لكل عميل، وتكتب تقرير JSON، باستخدام async/await ومتغيرات بيئة لمسارات الملفات.

المحاضرة 2 TypeScript لمطوري الـ Backend

- لماذا TypeScript: الـ types والـ inference والـ compiler
- الأنواع الأساسية والـ arrays والـ union والـ literal types والـ type narrowing
- الـ interfaces والـ type aliases لوصف شكل بيانات الـ API
- الـ generics للدوال القابلة لإعادة الاستخدام ولأغلفة الـ response
- ملف tsconfig مع strict mode، وتشغيل TypeScript باستخدام tsx والبناء باستخدام tsc
- Git branches والـ pull requests: إنشاء branch وفتح PR والمراجعة والدمج

تطبيق عملي في المحاضرة: تحوّل أداة الـ CLI من الجلسة السابقة إلى TypeScript بـ strict mode، وتضيف interfaces لكل شكل بيانات، وتصلح كل أخطاء الـ compiler، وتدمج التغيير عبر pull request يراجعه زميلك.

الواجب: اكتب module لإدارة المهام في الذاكرة باستخدام الـ types (إضافة، عرض، إنهاء، حذف، فلترة حسب الحالة) مع generics لنوع النتيجة، وارفعه في pull request يحتوي على 5 commits صغيرة على الأقل.

الأسبوع 3 SQL وتصميم PostgreSQL وأول API باستخدام Express

المحاضرة 1 البيانات العلائقية وPostgreSQL 17

- الجداول والصفوف والـ primary keys والـ foreign keys والـ constraints
- العلاقات: one to many وmany to many وجداول الربط
- أساسيات الـ normalization ومتى تتوقف عندها
- SQL: SELECT وWHERE وORDER BY وJOIN وGROUP BY والـ aggregates
- الـ indexes: وظيفتها وكيف تقرأ نتيجة EXPLAIN بشكل مبسط
- تشغيل PostgreSQL داخل Docker والاتصال بها عبر psql أو برنامج GUI

تطبيق عملي في المحاضرة: تصمم قاعدة بيانات لمتجر إلكتروني صغير على الورق، ثم تنشئها في PostgreSQL داخل Docker، وتضيف بيانات تجريبية، وتكتب 10 queries منها joins وaggregates.

المحاضرة 2 Express 5: الـ routing والـ middleware والأخطاء

- إنشاء تطبيق Express 5 باستخدام TypeScript
- الـ routes والـ route params والـ query strings وقراءة الـ request body
- الـ middleware: كيف تعمل السلسلة، والـ logging، وأماكن مبدئية للـ auth
- معالجة مركزية للأخطاء وشكل JSON موحد لها
- الأخطاء غير المتزامنة في Express 5 وإرجاع status codes صحيحة
- الاتصال بـ PostgreSQL عبر pg driver واستخدام parameterized queries

تطبيق عملي في المحاضرة: تبني REST API للملاحظات باستخدام Express 5 تحتوي على CRUD كامل على PostgreSQL، وmiddleware للـ logging، ومعالج أخطاء مركزي، وstatus codes صحيحة لحالات not found والمدخلات الخاطئة.

الواجب: أضف البحث والـ pagination والتحقق من المدخلات إلى API الملاحظات، واكتب دليل API قصيرًا في الـ README مع أمثلة requests، وحل 3 تحديات SQL على الـ joins والـ grouping.

الأسبوع 4 هيكل NestJS 11 وطبقة البيانات باستخدام Prisma

المحاضرة 1 أساسيات NestJS

- لماذا ننتقل من Express إلى NestJS: هيكل مناسب لمشاريع الفرق
- الـ modules والـ controllers والـ providers
- الـ dependency injection وكيف تُشارك الـ services
- الـ DTOs والـ validation pipes باستخدام class-validator أو Zod pipes
- الـ exception filters وشكل موحد لردود الأخطاء
- تجهيز المشروع: Nest CLI وESLint وPrettier وmodule لإعدادات البيئة

تطبيق عملي في المحاضرة: تنشئ مشروع Khedma باستخدام Nest CLI، وتضيف config module، وتبني module للتصنيفات يحتوي على CRUD endpoints وتحقق من الـ DTOs وتخزين مؤقت في الذاكرة.

المحاضرة 2 Prisma ORM مع PostgreSQL

- Prisma schema: الـ models والحقول والعلاقات والـ enums
- الـ migrations: الإنشاء والتطبيق والـ reset، وحفظها في Git
- استخدام Prisma Client داخل Nest service، وqueries بـ types
- العلاقات عمليًا: include وselect والكتابة المتداخلة
- إضافة بيانات تجريبية (seeding) للتطوير
- تصميم schema الخاصة بـ Khedma: المستخدمون ومقدمو الخدمة والخدمات والتصنيفات والحجوزات والتقييمات

تطبيق عملي في المحاضرة: تكتب Prisma schema كاملة لـ Khedma، وتشغّل أول migration على PostgreSQL داخل Docker، وتكتب seed script، وتنقل modules التصنيفات والخدمات إلى Prisma.

الواجب: أكمل CRUD للخدمات وملفات مقدمي الخدمة على Prisma مع التحقق من كل مدخل، وأضف صورة ER diagram إلى الـ README، وافتح pull request لكل module.

مرحلة في المشروع: Khedma API تعمل محليًا على NestJS 11 + Prisma + PostgreSQL 17 ببيانات تجريبية وCRUD للتصنيفات والخدمات ومقدمي الخدمة.

الأسبوع 5 تسجيل الدخول والصلاحيات والحجوزات

المحاضرة 1 الـ Auth باستخدام JWT والـ refresh tokens

- تشفير كلمات المرور باستخدام argon2 أو bcrypt، وعدم تخزينها كنص صريح أبدًا
- خطوات التسجيل والدخول، وإرجاع access token وrefresh token
- تركيب الـ JWT ومدة صلاحيته ومفاتيح التوقيع من متغيرات البيئة
- تدوير الـ refresh tokens وتخزينها مشفرة، وتسجيل الخروج من كل الأجهزة
- Auth guards وcurrent-user decorator في NestJS

تطبيق عملي في المحاضرة: تبني auth module لـ Khedma: التسجيل والدخول والـ refresh وتسجيل الخروج وendpoint محمي /me، ثم تختبر دورة الـ tokens كاملة في Postman أو Bruno.

المحاضرة 2 الأدوار والصلاحيات ومسار الحجز

- صلاحيات حسب الدور باستخدام roles decorator وguard
- فحص الملكية: العميل يرى حجوزاته فقط
- قواعد حالة الحجز لكل دور في صورة state machine صغيرة
- Prisma transactions لمنع حجز الموعد نفسه مرتين
- أنماط الـ pagination والفلترة والترتيب في endpoints القوائم

تطبيق عملي في المحاضرة: تضيف أدوار العميل ومقدم الخدمة والمسؤول، وتبني المواعيد المتاحة وإنشاء الحجز داخل transaction، وتحدد من يملك صلاحية تأكيد الحجز أو بدئه أو إنهائه أو إلغائه.

الواجب: أضف التقييمات (بعد حجز مكتمل فقط، وتقييم واحد لكل حجز)، وendpoints للمسؤول لاعتماد مقدمي الخدمة، وبحثًا بفلاتر وpagination على الخدمات.

مرحلة في المشروع: يستطيع المستخدمون التسجيل والدخول وحجز مقدم خدمة، وقواعد الأدوار والملكية مطبقة على كل route.

الأسبوع 6 الملفات والـ caching والـ queues والتحديثات اللحظية

المحاضرة 1 رفع الملفات وRedis caching

- رفع الملفات بصيغة multipart في NestJS، وحدود الحجم والنوع
- S3-compatible storage باستخدام MinIO محليًا: الـ buckets والـ keys والملفات العامة والخاصة
- Presigned URLs للرفع المباشر والتحميل الخاص
- أساسيات Redis: الـ keys ومدة الصلاحية وأنواع البيانات
- نمط cache-aside للقوائم العامة، وإلغاء الـ cache عند التعديل

تطبيق عملي في المحاضرة: تضيف رفع صورة مقدم الخدمة إلى MinIO مع التحقق منها، وترجع presigned URLs، وتخزن قائمة الخدمات العامة في Redis وتلغي الـ cache عند تغيير أي خدمة.

المحاضرة 2 مهام الخلفية وWebSockets

- لماذا يجب أن يخرج بعض العمل من دورة الـ request: الرسائل البريدية والتذكيرات والتقارير
- BullMQ queues والـ workers وإعادة المحاولة والـ backoff
- مهام مجدولة ومؤجلة لتذكير الحجوزات
- أساسيات WebSockets باستخدام NestJS gateway
- التحقق من هوية اتصالات الـ socket وإرسال events لمستخدم واحد

تطبيق عملي في المحاضرة: ترسل رسائل تأكيد الحجز عبر BullMQ worker (باستخدام صندوق بريد تجريبي)، وتجدول job للتذكير قبل كل حجز، وترسل تغييرات حالة الحجز إلى العميل عبر WebSockets.

الواجب: أضف حدًا لمرات إعادة المحاولة وسجلًا للـ jobs الفاشلة، وخزّن ملفات مقدمي الخدمة في الـ cache مع إلغاء صحيح، واكتب قسمًا قصيرًا في الـ README يشرح تصميم الـ queue والـ cache.

مرحلة في المشروع: أصبح Khedma يدعم رفع الصور، والقوائم المخزنة في الـ cache، والرسائل والتذكيرات في الخلفية، والتحديثات اللحظية لحالة الحجز.

الأسبوع 7 الاختبارات والأمان والتوثيق وDocker

المحاضرة 1 الاختبارات وتوثيق الـ API

- ماذا نختبر: unit tests للـ services واختبارات end to end للـ routes
- إعداد Jest أو Vitest في NestJS، والـ mocks وبيانات الاختبار
- اختبارات end to end باستخدام Supertest على قاعدة بيانات للاختبار
- OpenAPI باستخدام Nest Swagger module: الـ decorators والـ auth في التوثيق والأمثلة
- Logging منظم باستخدام Pino مع request IDs

تطبيق عملي في المحاضرة: تكتب unit tests لقواعد خدمة الحجز واختبارات end to end لـ routes الـ auth والحجز، ثم تنشر توثيق OpenAPI على /docs/ مع bearer auth يعمل.

المحاضرة 2 الأمان والـ containers

- OWASP Top 10 على مستوى الـ API: الـ injection وكسر صلاحيات الوصول وأخطاء الـ auth
- helmet وإعدادات CORS والـ rate limiting على الدخول والتسجيل
- الـ secrets ومتغيرات البيئة، وعدم رفع ملف .env إلى Git أبدًا
- Dockerfile بـ multi-stage build لتطبيق NestJS
- docker compose للـ API وPostgreSQL وRedis وMinIO
- Health check endpoint والإيقاف الآمن للتطبيق

تطبيق عملي في المحاضرة: تطبق checklist أمان على Khedma وتصلح ما يفشل فيه، وتضيف helmet والـ rate limits، ثم تكتب Dockerfile بـ multi-stage وملف docker compose يشغّل كل المكونات بأمر واحد.

الواجب: غطِّ كل قواعد الحجز ومسارات الـ auth بالاختبارات، وأصلح كل مشكلة من checklist الأمان، وسجّل فيديو شاشة مدته 3 دقائق تعرض فيه الـ API من خلال توثيق Swagger.

مرحلة في المشروع: Khedma API مختبرة وموثقة ومؤمنة وتعمل بالكامل داخل docker compose، وجاهزة للنشر.

البناء بالـ AI · الأسابيع 8

الأسبوع 8 البناء بالـ AI بالطريقة الصحيحة

المحاضرة 1 أدوات الـ AI للبرمجة على كود حقيقي

- كيف يعمل Claude Code وCursor: الـ context والـ prompts والـ agents وحدودها
- كيف تعطي الأداة سياق المشروع: الـ README وملف الـ conventions ومهام صغيرة ومحددة
- التخطيط أولًا: اطلب خطة وراجعها، ثم اسمح للأداة بكتابة الكود
- قراءة الـ diffs التي ينتجها الـ AI سطرًا بسطر قبل قبولها
- أخطاء الـ AI المعتادة في كود الـ Backend: APIs غير موجودة، وفحوصات auth ناقصة، وqueries غير آمنة، وإصدارات مكتبات قديمة

تطبيق عملي في المحاضرة: تستخدم Claude Code أو Cursor لإضافة endpoint لتقرير أرباح مقدم الخدمة في Khedma، وتراجع الـ diff باستخدام checklist، وترفض أو تصلح كل تغيير خاطئ قبل الدمج.

المحاضرة 2 الاختبارات كشبكة أمان، ومتى لا تثق في الـ AI

- اكتب الاختبارات أو راجعها أولًا، ثم اترك الـ AI ينفذ حتى تنجح
- استخدام الـ AI في كتابة الاختبارات، والتأكد من أن الاختبارات تختبر شيئًا فعلًا
- مراجعة أمان لكود الـ AI: الـ secrets والـ injection وصلاحيات الوصول
- متى لا تستخدم الـ AI: منطق auth لا تفهمه، وmigrations على بيانات حية، ونسخ كود لا تستطيع شرحه
- التراخيص والخصوصية، وعدم لصق secrets أو بيانات العملاء في أدوات الـ AI

تطبيق عملي في المحاضرة: مهمة ثنائية: يكتب طالب اختبارات فاشلة لخاصية رسوم إلغاء الحجز، ويستخدم الآخر الـ AI لتنفيذها، ثم يراجعان الكود معًا ويشرحان كل سطر.

الواجب: سلّم خاصية جديدة في Khedma (مثل أكواد الخصم أو ساعات عمل مقدم الخدمة) بمساعدة الـ AI، واكتب تقريرًا قصيرًا: ما الذي أصاب فيه الـ AI، وما الذي أخطأ فيه، وكيف كشفت الاختبارات الخطأ.

مرحلة في المشروع: أصبح في Khedma خاصية واحدة على الأقل منفذة بمساعدة الـ AI، ودُمجت فقط بعد مراجعة بشرية ونجاح الاختبارات.

مشروع التخرج · الأسابيع 9

الأسبوع 9 بناء مشروع التخرج والنشر الحقيقي

المحاضرة 1 تخطيط مشروع التخرج وجلسة بناء مكثفة

- قراءة وصف المنتج وتحويله إلى data model وendpoints
- تحديد النطاق: الأساسي والإضافي، وتقليص الخصائص للانتهاء في الوقت
- تجهيز الـ repo من قالب الدورة: lint واختبارات وDocker وإعدادات البيئة
- خطة يومية وGitHub issues وpull requests صغيرة
- متابعة مع المدرب لتصميم الـ schema والـ API

تطبيق عملي في المحاضرة: يختار كل طالب وصف مشروع، ويعتمد الـ schema وقائمة الـ endpoints مع المدرب، ويصل إلى auth وأول module أساسي يعملان قبل نهاية الجلسة.

المحاضرة 2 النشر على VPS باستخدام Coolify

- أساسيات الـ VPS: SSH والمستخدمون والـ firewall والتحديثات
- Coolify: المشاريع والتطبيقات من Git repo وقواعد البيانات والخدمات
- متغيرات البيئة والـ secrets في الإنتاج
- الدومين وسجلات الـ DNS وشهادات HTTPS
- تشغيل الـ migrations مع النشر، والـ health checks، وقراءة السجلات، والرجوع إلى إصدار سابق

تطبيق عملي في المحاضرة: ننشر Khedma API معًا كفصل أولًا، ثم ينشر كل طالب مشروع تخرجه مع PostgreSQL وRedis على Coolify، بدومين أو subdomain وHTTPS وتوثيق OpenAPI يعمل.

الواجب: أكمل خصائص مشروع التخرج، وحقق متطلبات الاختبارات والتوثيق، وأصلح مشكلات الإنتاج التي تظهر في السجلات، وجهّز عرضًا مدته 5 دقائق.

مرحلة في المشروع: API مشروع التخرج منشورة على URL عام بـ HTTPS، ومعها توثيق واختبارات وGit history نظيف.

الأسبوع 10 أسبوع التوظيف ويوم العرض

المحاضرة 1 الـ CV وLinkedIn وGitHub والمقابلات

- CV من صفحة واحدة لوظائف الـ Backend يتضمن المشاريع والروابط وتفاصيل بالأرقام
- ملف LinkedIn لوظائف الـ Backend في مصر وفي العمل عن بعد
- معرض أعمال على GitHub: repos مثبتة، وREADMEs تحتوي على diagrams، وروابط حية وتوثيق
- موضوعات مقابلات الـ Backend: HTTP وSQL والـ auth والـ event loop في Node.js وأساسيات تصميم الأنظمة
- مقابلات تقنية تجريبية تتضمن live coding وأسئلة تصميم API

تطبيق عملي في المحاضرة: يحدّث كل طالب الـ CV وLinkedIn وGitHub داخل الفصل بمراجعة المدرب، ثم يجري مقابلة تقنية تجريبية مدتها 20 دقيقة ويتلقى ملاحظات.

المحاضرة 2 أولى خطوات العمل الحر ويوم العرض

- العمل الحر على Upwork ومستقل وخمسات: الملف الشخصي، وأول العروض، وتسعير مهام API صغيرة
- تقدير وقت مهمة Backend وكتابة عرض واضح
- التعامل مع العملاء: النطاق والمراحل والتسليم والتوثيق
- يوم العرض: تقديم مشروع التخرج للمدربين وضيوف من سوق العمل

تطبيق عملي في المحاضرة: تكتب عرض عمل حر حقيقيًا لإعلان وظيفة Backend، ثم تعرض مشروع تخرجك مباشرة في يوم العرض: المشكلة، والـ data model، والـ API الحية، والاختبارات، وخطوتك التالية.

الواجب: قدّم على 10 وظائف Backend أو junior أو مهام عمل حر على الأقل خلال أسبوعين من يوم العرض، وشارك تقدمك في مجموعة الخريجين.

مشروع التخرج

يبني كل طالب REST API جاهزة للإنتاج لمنتج حقيقي انطلاقًا من وصف قصير، بالأدوات والممارسات نفسها المستخدمة في مشروع Khedma، وينشرها على VPS باستخدام Coolify. العمل فردي، مع متابعة من المدرب والـ office hours.

المتطلبات

- NestJS 11 + TypeScript بـ strict mode + Prisma + PostgreSQL 17
- 5 data models مرتبطة على الأقل، مع migrations وseed script
- JWT auth مع refresh tokens وكلمات مرور مشفرة ودوران على الأقل محميان بـ guards
- التحقق من كل المدخلات وشكل موحد لردود الأخطاء
- واحد على الأقل مما يلي: رفع ملفات إلى S3-compatible storage، أو Redis caching، أو BullMQ background job
- Unit tests لقواعد العمل واختبارات end to end للمسارات الأساسية
- توثيق OpenAPI منشور على /docs
- Dockerfile وdocker compose للتطوير المحلي
- نشر حي على VPS عبر Coolify بـ HTTPS ومتغيرات بيئة
- README يتضمن خطوات التشغيل وملاحظات عن الهيكل وER diagram وروابط حية
- Git history نظيف مع pull requests

أفكار مقترحة

- API لمواعيد العيادات: الأطباء والمواعيد والحجوزات وسجلات المرضى
- API لطلبات المطاعم: قوائم الطعام والسلة والطلبات وتحديثات حالة المطبخ عبر WebSockets
- API لاشتراكات النوادي الرياضية: الباقات والاشتراكات وتسجيل الحضور وتذكير انتهاء الاشتراك عبر BullMQ
- API لإعلانات العقارات: الوحدات والصور على S3 وفلاتر البحث والاستفسارات
- API للدورات والدروس الخصوصية: المدرسون والدورات والاشتراكات ومتابعة التقدم
- API لتتبع التوصيل لمتجر صغير: الطلبات والمندوبون والحالات والتحديثات اللحظية

معايير تقييم مشروع التخرج

يعمل من البداية إلى النهاية ومنشور على الإنترنت بـ HTTPS	25%
جودة تصميم الـ API والـ data model	20%
الـ auth والصلاحيات وممارسات الأمان	15%
الاختبارات: تغطية unit وend to end ذات معنى	15%
جودة الكود والهيكل والـ Git history	10%
التوثيق: OpenAPI والـ README	5%
العرض في يوم العرض والإجابة عن الأسئلة	10%

طريقة التقييم

الواجب الأسبوعي يُقيّم عبر pull requests على GitHub: يعمل، وكود نظيف، وcommits واضحة، ومسلّم في موعده.	20%
مراحل مشروع Khedma تُراجع في نهاية الأسابيع 4 و5 و6 و7 وفق checklist كل مرحلة.	25%
الاختبارات القصيرة ومراجعات الكود اختبارات قصيرة في الأسابيع 3 و5 و7، وجودة المراجعات التي تقدمها لزملائك.	10%
مهمة البناء بالـ AI خاصية منفذة بمساعدة الـ AI ومدمجة مع اختبارات ناجحة وتقرير مراجعة مكتوب.	5%
مشروع التخرج يُقيّم وفق معايير تقييم مشروع التخرج.	30%
مخرجات أسبوع التوظيف CV محدّث، وملف LinkedIn، ومعرض أعمال على GitHub، ومقابلة تجريبية، وعرض عمل حر واحد.	10%

قائمة المهارات

الويب وHTTP

- [ ] تشرح دورة الـ request والـ response وتختار الـ status codes الصحيحة
- [ ] تصمم REST resources وURLs واضحة
- [ ] تتتبع مشكلات الـ requests باستخدام curl وPostman أو Bruno وDevTools

JavaScript وTypeScript وNode.js

- [ ] تكتب كود async/await مع معالجة صحيحة للأخطاء
- [ ] تشرح الـ event loop وتتجنب إيقافه
- [ ] تستخدم الـ interfaces والـ unions والـ generics في TypeScript بـ strict mode
- [ ] تستخدم fs وenv وstreams في Node.js

قواعد البيانات

- [ ] تصمم schema منظمة بعلاقات وconstraints صحيحة
- [ ] تكتب SQL يتضمن joins وgrouping وindexes
- [ ] تستخدم Prisma migrations والعلاقات والـ transactions

NestJS وبناء الـ API

- [ ] تنظم التطبيق في modules وcontrollers وproviders
- [ ] تتحقق من المدخلات باستخدام DTOs وpipes
- [ ] تعالج الأخطاء باستخدام exception filters
- [ ] تضيف الـ pagination والفلترة والترتيب

الـ Auth والأمان

- [ ] تنفذ JWT access وrefresh tokens مع كلمات مرور مشفرة
- [ ] تحمي الـ routes باستخدام role guards وفحص الملكية
- [ ] تطبق helmet وCORS والـ rate limiting وأساسيات OWASP Top 10
- [ ] تُبقي الـ secrets خارج Git

الأداء ومهام الخلفية

- [ ] تستخدم Redis للـ caching وتلغي الـ cache بشكل صحيح
- [ ] تشغّل background jobs باستخدام BullMQ مع إعادة المحاولة والجدولة
- [ ] ترفع الملفات إلى S3-compatible storage باستخدام presigned URLs
- [ ] ترسل events لحظية عبر WebSockets

الجودة والتسليم

- [ ] تكتب unit tests واختبارات end to end باستخدام Jest أو Vitest وSupertest
- [ ] توثق الـ APIs باستخدام OpenAPI
- [ ] تضع التطبيق في containers باستخدام Docker وdocker compose
- [ ] تنشر على VPS عبر Coolify بدومين وHTTPS

العمل مع الـ AI

- [ ] تعطي أدوات الـ AI سياقًا واضحًا ومهام صغيرة
- [ ] تراجع الـ diffs التي ينتجها الـ AI وترفض الكود الخاطئ
- [ ] تستخدم الاختبارات للتحقق من عمل الـ AI

بعد التخرج

الأشهر 1-3 بعد التخرج

- التقديم على وظائف Backend للمبتدئين والتدريب كل أسبوع
- تحسين مشروع التخرج وإضافة خاصية كل أسبوع مع اختباراتها
- التدرب على مسائل SQL و3 data structures مرات أسبوعيًا
- تنفيذ أول مهمة API صغيرة في العمل الحر، حتى بسعر منخفض، لبناء التقييمات

الأشهر 4-6

- التعمق في PostgreSQL: الـ query plans واستراتيجية الـ indexes والـ views والـ full text search
- إضافة CI باستخدام GitHub Actions أو أي runner متاح، ومراقبة بـ uptime checks
- تعلم payment gateway مستخدمة في مصر مثل Paymob، والتعامل مع الـ webhooks
- المساهمة بإصلاحات صغيرة في مشروع Node.js مفتوح المصدر

الأشهر 7-12: نحو مستوى mid-level

- أساسيات تصميم الأنظمة: توسيع القراءة والـ queues والـ idempotency والـ rate limits
- Microservices مقارنة بالـ modular monolith، والـ message brokers والتصميم المبني على الأحداث
- GraphQL أو gRPC كأسلوب API ثانٍ
- أساسيات الـ cloud على AWS أو GCP، والمراقبة بالـ metrics والـ tracing
- توجيه طالب أحدث وقيادة مراجعات الكود

الوظائف التي يقود إليها المسار

Junior Backend Developer (Node.js) | Junior NestJS Developer | Node.js / TypeScript Developer

API Developer | Junior Full Stack Developer (مع مسار Frontend لاحقًا)

Backend Intern في شركات البرمجيات والـ startups | Freelance API Developer على Upwork ومستقل وخمسات

Remote Junior Backend Engineer لشركات في الخليج وأوروبا

مصادر

- توثيق Node.js: https://nodejs.org/docs/latest/api/
- دليل JavaScript على MDN: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide
- HTTP على MDN: https://developer.mozilla.org/en-US/docs/Web/HTTP
- TypeScript Handbook: https://www.typescriptlang.org/docs/handbook/intro.html
- توثيق Express: https://expressjs.com/
- توثيق NestJS: https://docs.nestjs.com/
- توثيق Prisma: https://www.prisma.io/docs
- توثيق PostgreSQL 17: https://www.postgresql.org/docs/17/
- توثيق Redis: https://redis.io/docs/latest/
- توثيق BullMQ: https://docs.bullmq.io/
- توثيق Jest: https://jestjs.io/docs/getting-started
- توثيق Vitest: https://vitest.dev/guide/
- Supertest: https://github.com/ladjs/supertest
- مواصفات OpenAPI: https://swagger.io/specification/
- OWASP Top 10: https://owasp.org/www-project-top-ten/
- توثيق Docker: https://docs.docker.com/
- توثيق Coolify: https://coolify.io/docs/
- كتاب Pro Git: https://git-scm.com/book/en/v2
- GitHub Docs: https://docs.github.com/
- توثيق pnpm: https://pnpm.io/motivation