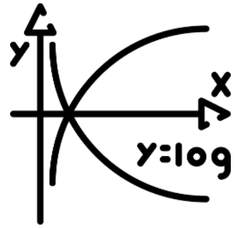
من الرياضيات إلى البرمجة الحديثة

محمد مهند الطحان

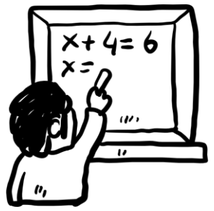
ERP-System

مطور برمجيات full stack developer
متخصص في أنظمة
(Enterprise-Resource-Planning)



mathematics

خريج جامعة دمشق قسم الرياضيات شعبة التحليل الرياضي



Mathematics teacher

مدرس رياضيات سابقاً في مدارس دمشق

ERP SYSTEM

Integrated Solutions for Your Business

حلول متكاملة لمؤسستك

HUMAN RESOURCES (HR)
- Personnel Management
- Time Tracking
- Employee Development

الموارد البشرية (HR)
- إدارة شؤون الموظفين
- تتبع الوقت والحضور
- تطوير الموظفين

FINANCE & ACCOUNTING
- General Ledger
- Cost Accounting
- Budgeting & Reporting

المالية والمحاسبة
- دفتر الأستاذ العام
- محاسبة التكاليف
- إعداد الميزانية والتقارير

SALES & CRM
- Customer Management
- Quotes & Orders
- Sales Control

المبيعات وإدارة علاقات العملاء
- إدارة العملاء
- عروض الأسعار والطلبات
- إدارة المبيعات

PRODUCTION
- Production Planning
- Resource Management
- Manufacturing Process

الإنتاج
- تخطيط الإنتاج
- إدارة الموارد
- عمليات التصنيع

INVENTORY & LOGISTICS
- Inventory Management
- Warehouse Operations
- Supply Chain Optimization

المخزون والخدمات اللوجستية
- إدارة المخزون
- عمليات المستودعات
- تحسين سلسلة التوريد

PURCHASING
- Supplier Management
- Purchase Orders
- Procurement Process

المشتريات
- إدارة الموردين
- أوامر الشراء
- عملية الشراء والتوريد

ERP

One System.
Centralized Data.
All Departments.

نظام واحد.
بيانات مركزية.
جميع الأقسام.

محاور المحاضرة

دليل متكامل يربط بين المهارات التقنية، الأدوات الحديثة، والآفاق المهنية

المنظومة التقنية المتكاملة

ربط تطوير الويب Front-end بالأنظمة الخلفية Backend، قواعد البيانات، وتطبيقات الهاتف وسطح المكتب

أدوات الإنتاجية والبيئات

دور Linux و Docker و Git في بناء وإدارة المشاريع البرمجية بكفاءة عالية وضمان استقرار البيئات

الأساس العلمي والمنطقي

أهمية الرياضيات في البرمجة وكيف تساهم في تطوير مهارات حل المشكلات المعقدة والمنطق البرمجي

الرؤية المهنية وسوق العمل

استكشاف الآفاق المهنية لطلاب الرياضيات والبرمجة وخارطة الطريق الفعالة للانتقال من التعلم إلى الاحتراف

من التأسيس إلى سوق العمل

البرمجة ليست مجرد كتابة كود؟

دليل متكامل يربط بين المهارات التقنية، الأدوات الحديثة

Systems

أنظمة مترابطة تعمل معًا.

Architecture

قبل كتابة الكود يجب تصميم هيكل النظام.

Teams

المشاريع تُبنى من خلال فرق عمل متعددة التخصصات.

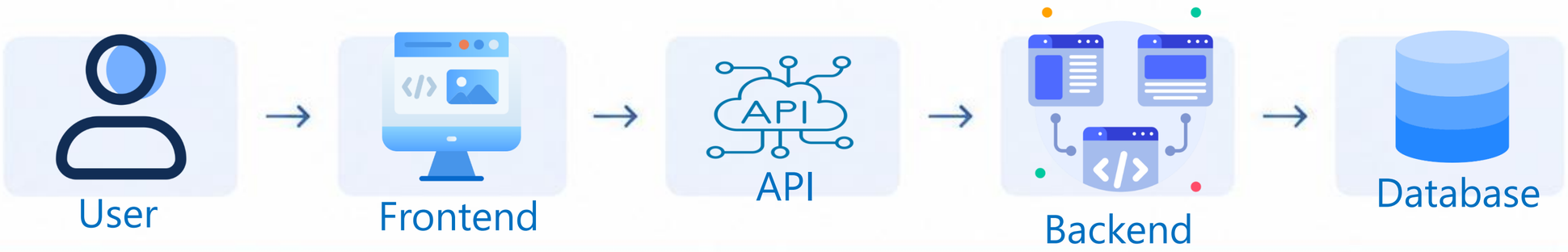
Deployment

التطبيق يجب أن يعمل على السيرفرات وليس فقط على جهاز المطور.

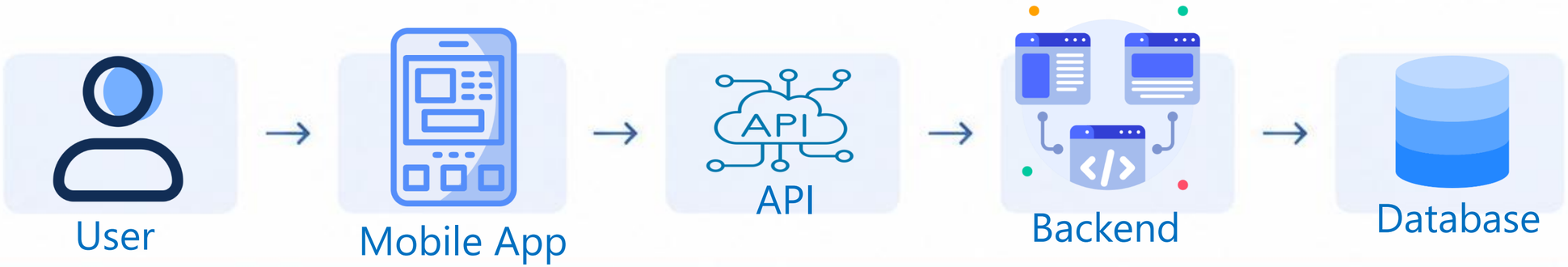
من التأسيس إلى سوق العمل

ماذا لديك خيارات للبرمجة

تطبيق ويب
Web app



تطبيق موبايل
mobile app



تطبيق سطح المكتب
Desktop app

تطبيقات سطح المكتب Desktop Apps

برامج تُثبّت على أجهزة المستخدم وتعمل بدون الحاجة لاتصال دائم بالإنترنت

الخصائص

- أداء أعلى
- تجربة مستقرة
- بدون إنترنت
- الوصول للملفات
- واجهة غنية

أدوات وتقنيات شائعة

Windows	• WinForms (.NET Framework) • WPF Windows Presentation Foundation (.NET) • UWP (Universal Windows Platform
macOS	• Swift (AppKit) • Objective-C (Cocoa) • Electron
Linux	• GTK (C / Python / Vala) • Qt (C++ / Python) • Electron

أمثلة على تطبيقات سطح المكتب

Microsoft Office (Word, Excel, PowerPoint)

Adobe Photoshop

Visual Studio Code

VLC Media Player

AutoCAD

Discord Desktop

... وغيرها الكثير

تطبيقات سطح المكتب Desktop Apps

أهم الخصائص والميزات

- أداء عالي وسرعة استجابة ممتازة
- إمكانية الوصول إلى جميع موارد الجهاز
- العمل بدون اتصال بالانترنت
- واجهة رسومية غنية وتجربة مستخدم أفضل
- أمان أكبر لأن البيانات تبقى على الجهاز
- إمكانية العمل على ملفات كبيرة و معقدة
- تخصيص كامل حسب احتياجات المستخدم

ملاحظات مهمة

- ✓ اختر التقنية المناسبة حسب نظام التشغيل والجمهور المستهدف .
- ✓ تعلّم أساسيات البرمجة وهياكل البيانات ضروري لأي تطبيق.
- ✓ اهتم بتجربة المستخدم (UI/UX) لأنها جزء أساسي من نجاح التطبيق.
- ✓ اختبر تطبيقك على أكثر من نظام تشغيل إن أمكن.

كيف تعمل تطبيقات سطح المكتب

01

تثبيت البرنامج

يتم تثبيت الملفات والمكتبات اللازمة على الجهاز.

02

تشغيل التطبيق

يعمل البرنامج داخل النظام.

03

معالجة البيانات محلياً

يتم تخزين ومعالجة البيانات داخل الجهاز.

04

التفاعل مع الأجهزة

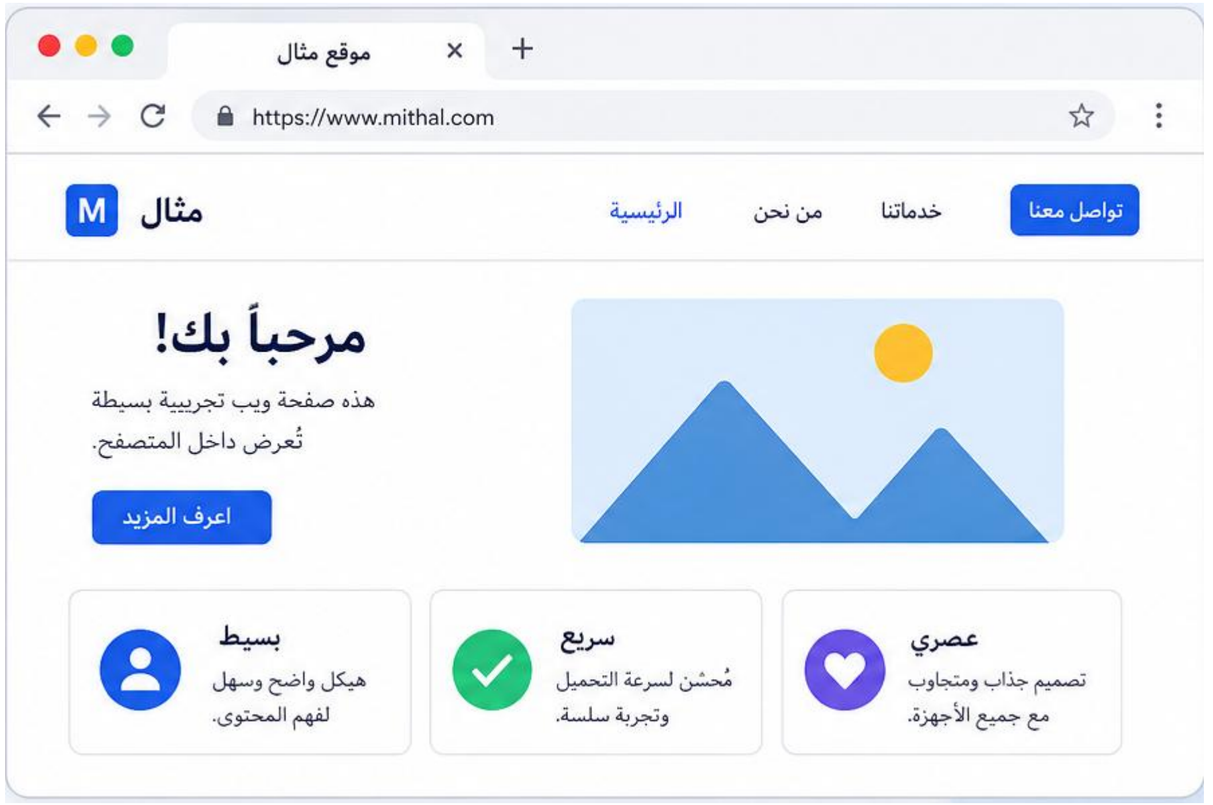
يمكن للتطبيق استخدام الطابعة، الكاميرا، الميكروفون، إلخ.

05

الحفظ والإغلاق

يحفظ التطبيق البيانات محلياً عند الحاجة.

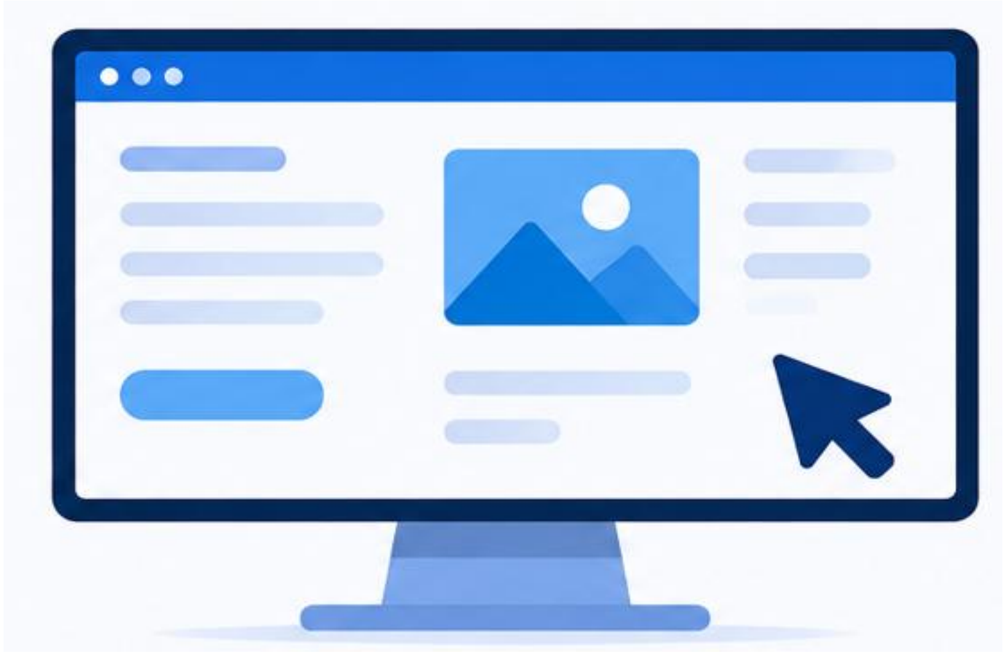
الواجهة الأمامية
(Frontend)



ما هو الـ Frontend؟

الـ Frontend هو الجزء الذي يتفاعل معه المستخدم مباشرة في المتصفح أو التطبيق.

هو المسؤول عن عرض البيانات، إرسال الطلبات، واستقبال الاستجابات من الـ Backend



واجهة المستخدم
User Interface (UI)

كل ما يراه المستخدم على الشاشة.

تفاعل المستخدم
User Experience (UX)

كيفية تفاعل المستخدم مع التطبيق بسهولة.

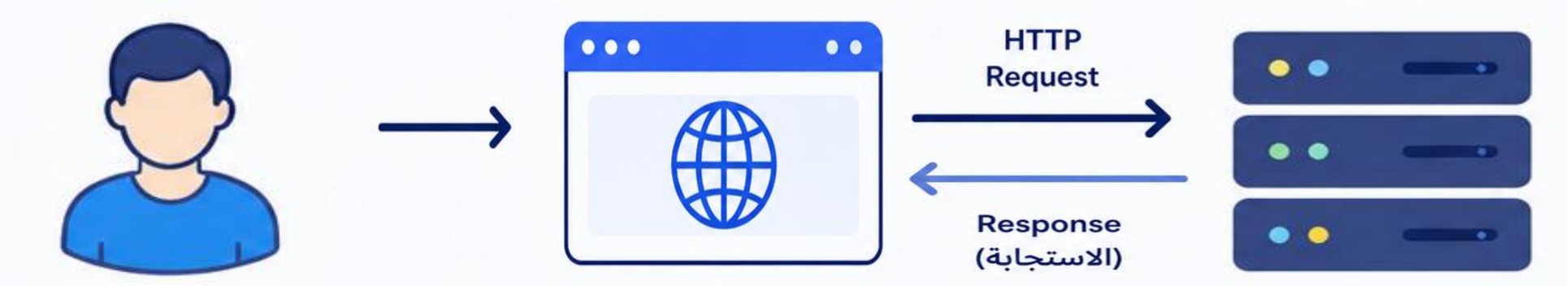
عرض البيانات

عرض المعلومات بشكل منظم وجذاب.

إرسال واستقبال الطلبات

التواصل مع الـ Backend لجلب أو إرسال البيانات.

كيف يعمل ال Frontend



فكرة أساسية

ال Frontend هو الوسيط الذي يتواصل بين المستخدم (الإنسان) والنظام (الخادم)

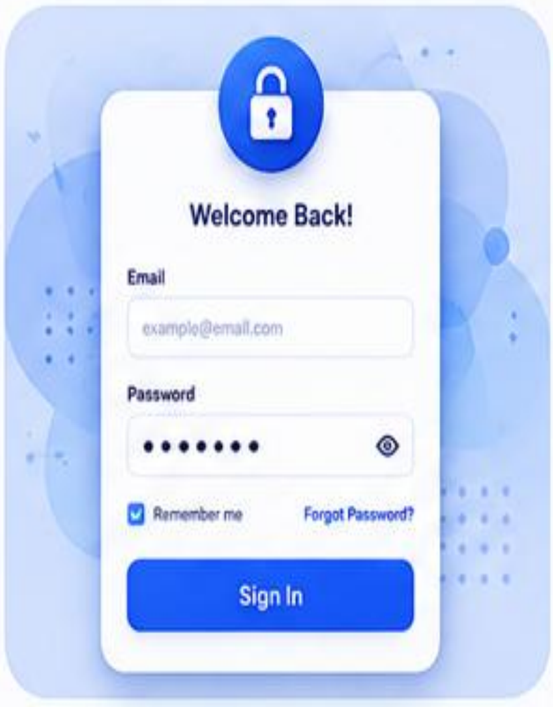
1. يتفاعل المستخدم مع الواجهة (يطلب أو ينقر على زر...).
2. يرسل الـ Frontend الطلب Request إلى الـ Backend.
3. الـ Backend يعالج الطلب ويوجه البيانات المناسبة.
4. الـ Frontend يعرض البيانات للمستخدم بطريقة منظمة وسهلة.

الهدف جعل التعامل مع البيانات سريعاً, واضحاً, وجميلاً للمستخدم

أمثلة حقيقية على الـ Frontend

صفحة تسجيل الدخول
Login page

واجهة تسمح للمستخدم بالدخول إلى النظام بأمان.

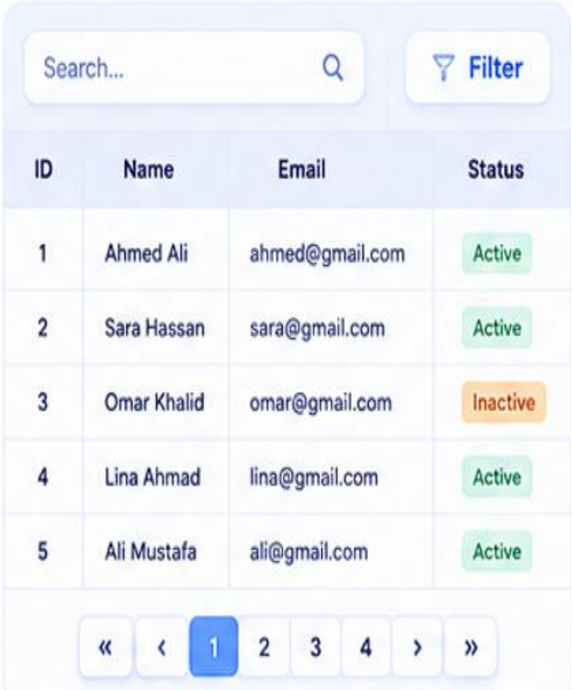


لوحة تحكم
Dashboard

تعرض البيانات والإحصائيات بشكل مري وسهل الفهم

عرض البيانات (جداول)
Data tables

عرض البيانات في جداول يمكن البحث, الفلترة والترتيب فيها



تطبيقات الهاتف
Mobile Apps

واجهات متجاوبة تعمل على مختلف أحجام الشاشات

تقنيات ال Frontend الأساسية

تستخدم هذه التقنيات لبناء واجهة المستخدم وجعلها أكثر تفاعلية وحذابة وسريعة الاستجابة

HTML

- لغة ترميز تُستخدم لإنشاء هيكل صفحات الويب.
- تحدد العناصر مثل العناوين والفقرات والصور والروابط.
- هي الأساس لأي صفحة ويب.

CSS

- تُستخدم لتنسيق وتصميم صفحات الويب.
- تتحكم في الألوان والخطوط والمسافات والتخطيط.
- تفصل بين المحتوى (HTML) والشكل (التصميم).

JavaScript

- لغة برمجة تجعل صفحات الويب تفاعلية وحيوية.
- تتحكم في العناصر والأحداث والتحقق من البيانات.
- أساس أي تطبيق ويب ديناميكي.

TypeScript

- نسخة مطورة من JavaScript تضيف الأمان والوضوح.
- تساعد على كتابة كود أكبر وأكثر تنظيماً.
- تقلل الأخطاء أثناء التطوير.

DOM

- مثل بنية صفحة الويب داخل المتصفح.
- يسمح لـ JavaScript بتعديل العناصر والمحتوى.
- أساس التفاعل الديناميكي في صفحات الويب.

Responsive Design التصميم المتجاوب

- يجعل الموقع مناسبًا لجميع أحجام الشاشات.
- يضمن ظهور الصفحة بشكل جيد على الحاسوب والموبايل والتابلت.
- يعتمد على CSS وتقنيات مثل Flexbox و Grid و Media Queries.

اتباع هذه الممارسات يمنحك منتجات أفضل، وتجربة مستخدم رائعة، وكوداً مستداماً وقابلاً للتطوير.

أشهر أطر عمل الـ Frontend

أطر العمل هي أدوات جاهزة تساعد المطور على بناء تطبيقات Frontendبسرعة وتنظيم أفضل، وتوفر مكونات وإمكانات كثيرة.

Vue.js

إطار عمل تدريجي لبناء واجهات مستخدم سهلة وفعّالة.

- ❑ **سهل التعلم:**
  بسيط في البداية وقوي عند الحاجة.
- ❑ **مرن وتدريجي:**
  يمكن استخدامه في أجزاء صغيرة من المشروع.
- ❑ **أداء ممتاز:**
  حجم صغير وسرعة عالية في التنفيذ.
- ❑ **مجتمع متنامٍ:**
  شعبيته في تزايد ودعمه ممتاز.

تُستخدم في ... Alibaba, Xiaomi, GitLab

Angular

إطار عمل متكامل من Google لبناء تطبيقات ويب قوية وكبيرة.

- ❑ **إطار عمل متكامل (Full Framework)**
  بسيط في البداية وقوي عند الحاجة.
- ❑ **يعتمد على TypeScript**
  يمكن استخدامه في أجزاء صغيرة من المشروع.
- ❑ **هيكلية قوية ومنظمة**
  حجم صغير وسرعة عالية في التنفيذ.
- ❑ **أمان عالٍ**
  شعبيته في تزايد ودعمه ممتاز.

تُستخدم في ... Google, YouTube, Microsoft

React

مكتبة JavaScript لبناء واجهات مستخدم تفاعلية.

- ❑ **تعتمد على المكونات (Components)**
  أقسّم الواجهة إلى مكونات قابلة لإعادة الاستخدام.
- ❑ **أداء عالٍ وسرعة**
  تستخدم Virtual DOMلتحسين الأداء.
- ❑ **مرونة كبيرة**
  يمكن استخدامها مع أي مكتبة أو مشروع.
- ❑ **دعم مجتمعي ضخم**
  لها مجتمع كبير وتحديثات مستمرة.

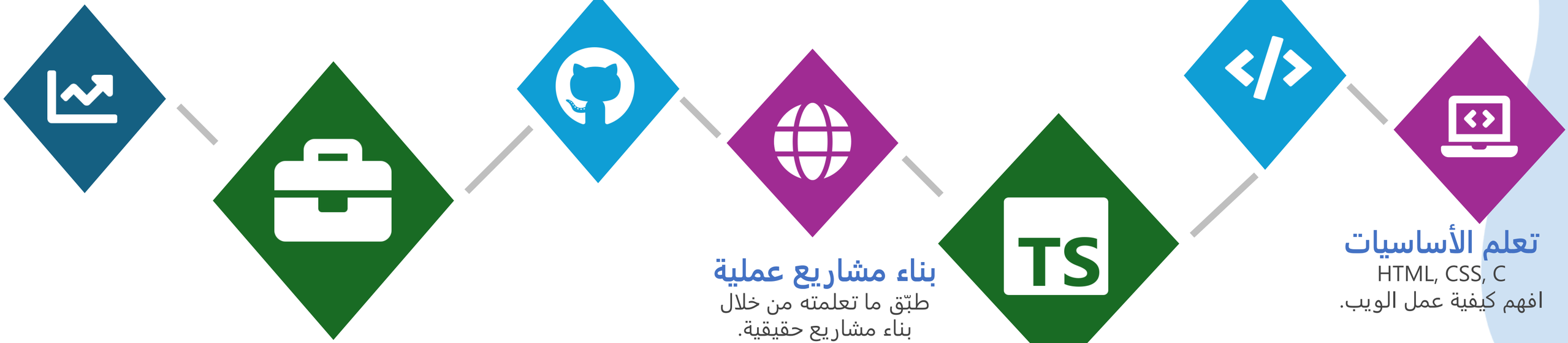
تُستخدم في Facebook, Instagram, Netflix

مسؤوليات الـ Frontend

- عرض البيانات بشكل واضح وجميل
- التأكد من سهولة الاستخدام (User Experience)
- التأكد من سرعة واستجابة التطبيق
- التفاعل مع ال Backend من خلال APIs
- التأكد أن التطبيق يعمل على كافة الأجهزة والمتصفحات

المسار المهني في مجال الـ Frontend

الـ Frontend مجال واسع ويوفر فرصاً مميزة للنمو والتطوير.
إليك مساراً مقترحاً لتطوير مهاراتك وبناء مستقبلك المهني.



تعلم الأساسيات
HTML, CSS, C
افهم كيفية عمل الويب.

تعلم المكتبات والأطر
اختر مكتبة كـ React أو Angular أو Vue.js.

تعميق المهارات
TypeScript، إدارة الحالة، التفاعل مع APIs.

بناء مشاريع عملية
طبّق ما تعلمته من خلال بناء مشاريع حقيقية.

بناء ملف أعمال احترافي
ارفع مشاريعك على GitHub وأنشئ موقعاً شخصياً.

الحصول على فرصة عمل
قدّم على الوظائف، استعدّ للمقابلات، وطوّر مهاراتك.

التطور والنمو
استمر في التعلم، تابع التقنيات الجديدة، وتخصص.

مجالات العمل المتاحة للـ Frontend Developer

الـ Frontend مجال واسع ويوفر فرصاً مميزة للنمو والتطوير.
إليك مساراً مقترحاً لتطوير مهاراتك وبناء مستقبلك المهني.

Frontend Developer
مطور واجهات مستخدم

Web & Mobile Dev
مطور تطبيقات الويب
والموبايل

Performance Dev
مطور تحسين الأداء

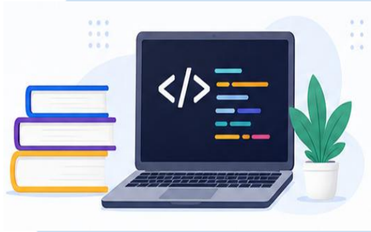
UI Developer
مطور واجهات تجربة المستخدم

Accessibility Dev
مطور إمكانية الوصول

Desktop App Dev
مطور تطبيقات مكتبية

المفتاح هو الاستمرارية في التعلم، والممارسة، وبناء مشاريع حقيقية.

مصادر مفيدة لتعلم وتطوير Frontend



مجموعة من أفضل المواقع والأدوات والمجتمعات لمساعدتك في تعلم وتطوير مهارات **Frontend** والبقاء على اطلاع دائم.

1 التوثيق (Documentation)

- MDN Web Docs — developer.mozilla.org
- W3Schools — w3schools.com
- CSS-Tricks — css-tricks.com
- JavaScript.info — javascript.info
- React Docs — react.dev

2 منصات التعلم (Learning Platforms)

- freeCodeCamp — freecodecamp.org
- Frontend Mentor — frontendmentor.io
- Coursera — coursera.org
- Udemy — udemy.com
- Scrimba — scrimba.com

3 الأدوات (Tools)

- VS Code — code.visualstudio.com
- CodePen — codepen.io
- JSFiddle — jsfiddle.net
- Postman — postman.com
- GitHub — github.com

4 المجتمعات (Communities)

- Stack Overflow — stackoverflow.com
- Reddit r/webdev — reddit.com/r/webdev
- Dev.to — dev.to
- Hashnode — hashnode.com
- Frontend Masters Community

5 التصميمية (Design Resources)

- Figma — figma.com
- UI8 — ui8.net
- Dribbble — dribbble.com
- Coolors — coolors.co
- Font Awesome — fontawesome.com

6 قنوات يوتيوب (YouTube Channels)

- Traversy Media
- Web Dev Simplified
- Net Ninja
- Academind
- JavaScript Mastery

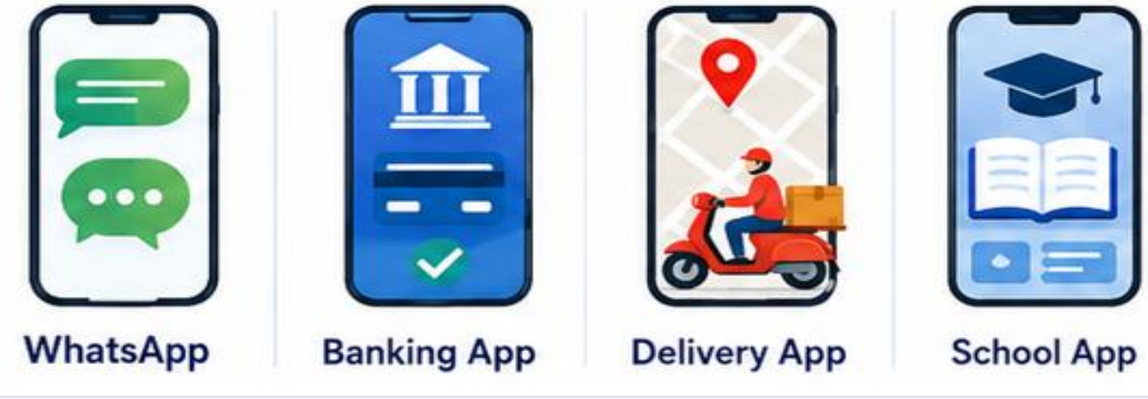
نصيحة: استمر في التعلم، وطبق ما تتعلمه عملياً، وتفاعل مع المجتمع، فالممارسة هي مفتاح الاحتراف.

ما هي تطبيقات الهاتف المحمول؟

تعريف What are mobile apps

- تطبيقات تعمل على الهواتف الذكية والأجهزة اللوحية
- تثبَّت على Androidو – iOS
- قد تعمل Online أو Offlineجزئياً

أمثلة



كيف تعمل؟

- غالباً تتصل مع Backendو API.
- يمكنها استخدام الكاميرا، الموقع GPS، والإشعارات.
- قد تحفظ بيانات محلياً Local Storage.

تطوير التطبيقات الأصلي Mobile Applications

التطوير الأصلي يعني بناء تطبيق مخصص لكل منصة باستخدام اللغة والأدوات الرسمية الخاصة بها

الأندرويد Android Development

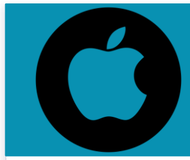
- ✓ اللغة الأساسية: Kotlin أو Java
- ✓ بيئة التطوير: Android Studio
- ✓ نشر التطبيق: Google Play Store
- ✓ خصائص النظام المتاحة

الوصول الكامل لخصائص الجهاز مثل: الكاميرا، الموقع GPS، جهات الاتصال، الإشعارات، البلوتوث، الحساسات، وغيرها

- ✓ الأداء

أفضل أداء لأن التطبيق مبني خصيصاً لنظام الأندرويد ويستفيد من إمكانيات الجهاز بالكامل

iOS Development iOS



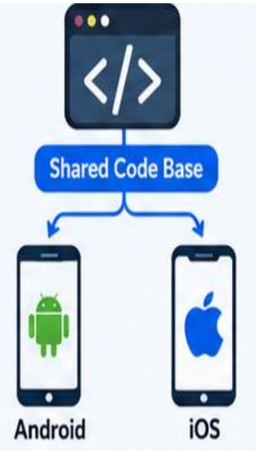
- ✓ اللغة الأساسية: Swift
- ✓ بيئة التطوير: Xcode
- ✓ نشر التطبيق: Apple App Store
- ✓ خصائص النظام المتاحة
- ✓ الوصول الكامل لخصائص الجهاز مثل: الكاميرا، الموقع GPS، جهات الاتصال، الإشعارات، Face ID / Touch ID، البلوتوث، الحساسات، وغيرها
- ✓ الأداء

أفضل أداء لأن التطبيق مبني خصيصاً لنظام iOS ويستفيد من إمكانيات أجهزة Appleبالكامل

ملاحظات مهمة: تطوير Androidلا يعمل على iOSوالعكس صحيح

تطوير التطبيقات متعددة المنصات
Cross-Platform Mobile Development

ما معنى Cross-Platform؟



- بناء تطبيق واحد يعمل على أكثر من منصة
- تقليل تكرار العمل بين Android وiOS
- مناسب للشركات التي تريد الوصول السريع إلى السوق

أشهر التقنيات

Xamarin (legacy)

من Microsoft سابقاً .كان شائعاً لبناء تطبيقات أصلية من C#

Ionic / Angular

تطوير باستخدام Angular مع مكونات UI قوية وأدوات جاهزة

.NET MAUI / C#

من Microsoft لبناء تطبيقات متعددة المنصات بكود C# واحد

React Native / JavaScript

من Meta لبناء تطبيقات باستخدام JavaScript و React

Flutter / Dart

إطار عمل من Google لبناء تطبيقات سريعة وجميلة

متى نختاره؟

- عند الحاجة إلى إطلاق سريع
- عند محدودية الميزانية
- عندما تكون متطلبات Android وiOS متشابهة
- عندما نريد MVP أو تطبيق أعمال/خدمات

التحديات

- قد يكون الأداء أقل من Native في بعض الحالات
- بعض خصائص الجهاز تحتاج كوداً أصلياً
- الاعتماد على إطار العمل وتحديثاته
- مشكلات محتملة مع بعض المكتبات أو الأجهزة الخاصة

المزايا

- توفير الوقت والتكلفة
- فريق واحد يمكنه خدمة منصتين
- تحديثات أسرع
- إعادة استخدام جزء كبير من الكود
- مناسب للشركات الصغيرة والمتوسطة

مقارنة بين الأنواع الثلاثة Mobile App vs Web App vs Desktop App

وجه المقارنة	Mobile App تطبيق موبايل	Web App تطبيق ويب	Desktop App تطبيق سطح مكتب
منصة التشغيل	يعمل على الهواتف والأجهزة اللوحية (Android و iOS)	يعمل على المتصفحات في أي جهاز (Windows, Mac, Linux, Android, iOS)	يعمل على أجهزة الكمبيوتر (Windows, macOS, Linux)
الوصول والاستخدام	من خلال فتح التطبيق بعد تثبيته	من خلال المتصفح عبر رابط URL	من خلال فتح البرنامج بعد تثبيته
التثبيت	يحتاج تثبيت من المتجر (Google Play / App Store)	لا يحتاج تثبيت	يحتاج تثبيت على الجهاز
التحديثات	تحديث من المتجر	تحديث فوري على الخادم يصل للجميع مباشرة	تحديث عبر تحميل نسخة جديدة وتثبيتها
الأداء	أداء عالي ومُحسَّن للأجهزة المحمولة	يعتمد على المتصفح والإنترنت	أداء عالي؛ مناسب للمهام الثقيلة
الميزات والأجهزة	الوصول الكامل لميزات الهاتف (الكاميرا، GPS، الإشعارات، جهات الاتصال...)	إمكانيات محدودة حسب المتصفح	الوصول الكامل لأجهزة الكمبيوتر (الطابعة، الملفات، البلوتوث، ...)
التكلفة والوقت	تكلفة أعلى نسبياً وتطوير أطول خاصة في Native لكل منصة	أقل تكلفة وأسرع في التطوير والنشر	تكلفة متوسطة وتختلف حسب المنصة
الاستخدام المناسب	تطبيقات تحتاج لتفاعل مع الهاتف، العمل بدون إنترنت، وإشعارات	تطبيقات معلوماتية، مواقع، لوحات تحكم، أنظمة، إدارية	تطبيقات تحتاج أداء عالٍ أو العمل مع أجهزة طرفية

ما هو الـ Backend؟

الـ Backend هو الجزء الذي يعمل خلف الكواليس في التطبيقات والمواقع، يتعامل مع المنطق، قاعدة البيانات، المستخدمين، والأمان

ماذا يشمل الـ Backend

إدارة البيانات ومعالجتها
التعامل مع تخزين البيانات، تنظيمها، والاستعلام عنها

الخوادم
استقبال الطلبات من المستخدمين ومعالجتها

واجهات برمجية APIs
Application Programming Interface
توفير واجهات للتواصل بين ال Backendوال Frontendأو تطبيقات أخرى

الأداء والتوسع
تحسين سرعة التطبيق والتعامل مع عدد كبير من المستخدمين

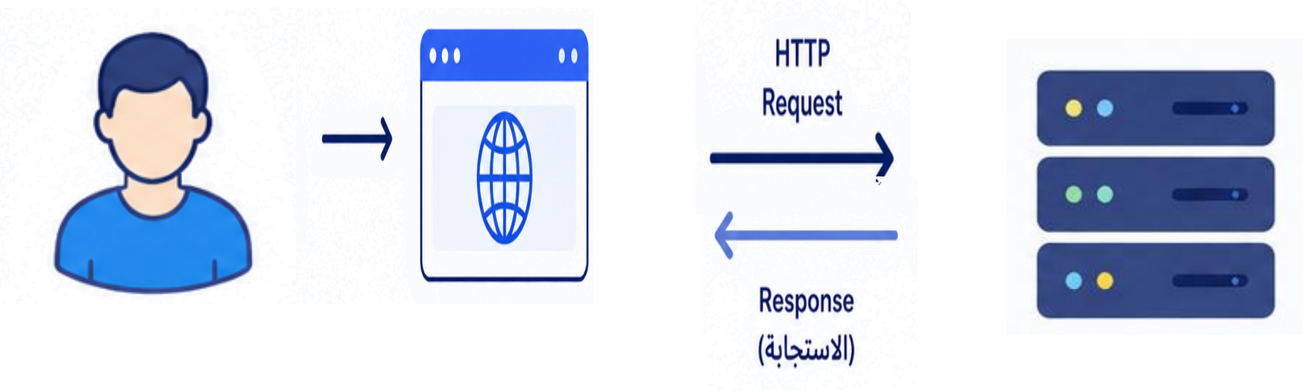
الأمان والمصادقة
حماية البيانات، تسجيل الدخول، الصلاحيات، والتشفير

منطق الأعمال
تنفيذ القواعد والعمليات التي تحقق أهداف التطبيق

فكرة أساسية
إذا كان ال Frontend هو ما يراه المستخدم ويتفاعل معه فإن ال Backend هو العقل الذي يدير كل شيء

الهدف
بناء نظام خلفي موثوق، آمن، وسريع يدعم التطبيق ويوفر تجربة مستخدم ممتازة

كيف يعمل ال Backend؟



ما الذي يحدث في الخلفية؟

استقبال الطلب
يستقبل الـ Backend الطلب القادم من Frontend، عبر بروتوكول HTTP مثل POST, GET

التحقق والمعالجة
يتحقق من صحة الطلب، وصلاحيات المستخدم، ثم يعالج البيانات باستخدام منطق الأعمال

الوصول إلى البيانات
يتفاعل مع قاعدة البياناتDatabase لقراءة أو إضافة أو تحديث أو حذف البيانات المطلوبة

إرسال الاستجابة
يرسل الـ Backend البيانات المناسبة إلى الـ Frontend عادة بصيغة JSON أو XML

عرض البيانات
يعرض الـ Frontend البيانات للمستخدم بطريقة منظمة وجذابة

1. يتفاعل المستخدم مع الواجهة يطلب أو ينقر على زر
2. يرسل الـ Frontend الطلب Requestإلى الـ Backend
3. الـ Backend يعالج الطلب ويوجه البيانات المناسبة
4. الـ Frontend يعرض البيانات للمستخدم بطريقة منظمة وسهلة

الخلاصة

الـ Backend يعمل كوسيط ذكي بين المستخدم والبيانات والخدمات، يضمن معالجة الطلبات بأمان وسرعة وتقديم المعلومات الصحيحة والمناسبة في كل مرة

الفرق بين ال Frontend وال Backend

Backend
الخادم والمنطق

ما لا يراه المستخدم

يعمل على الخادم ويتعامل مع البيانات والمنطق.

يعمل على الخادم (Server)

يستقبل الطلبات من الـ Frontend ويعالجها.

يُكتب باستخدام

Java, Python, PHP, Node.js, Django, Spring ...

المسؤولية الأساسية

إدارة البيانات، الأمان، المصادقة، وقاعدة البيانات.

VS

Frontend
واجهة المستخدم

ما يراه المستخدم

كل ما يتفاعل معه المستخدم على الشاشة.

يعمل في المتصفح

يتم تشغيله داخل المتصفح)مثل .(Chrome

يُكتب باستخدام

HTML, CSS, JavaScript, React, Angular ...

المسؤولية الأساسية

عرض البيانات بشكل جميل وسهل للمستخدم.

Frontendيطلب تنفيذ العملية
وال Backend يعالج الطلب ويعيد النتيجة

تقنيات الـ Backend الأساسية

لماذا توجد تقنيات متعددة؟
لا توجد تقنية واحدة تناسب كل شيء، لذلك نستخدم تقنيات مختلفة لكل جزء من أجزاء الـ Backendلتحقيق أفضل أداء ومرونة وأمان

01 Language

Python

لغة برمجة عالية المستوى وسهلة التعلم وقوية

تستخدم في

- تطوير الـ Backend
- تحليل البيانات
- الذكاء الاصطناعي

02 Framework

Django

إطار عمل لـ Python لتطوير تطبيقات ويب بسرعة وأمان

تستخدم في

- بناء تطبيقات ويب كاملة
- إدارة قاعدة البيانات
- نظام المصادقة

03 Database

PostgreSQL

قاعدة بيانات علاقية قوية، مفتوحة المصدر وداعمة للـ SQL

تستخدم في

- تخزين البيانات
- العلاقات المعقدة
- الاستعلامات المتقدمة

04 Server

Nginx

خادم ويب عالي الأداء يعمل كوسيط Reverse Proxy وموازن تحميل

تستخدم في

- استقبال الطلبات
- توجيهها للتطبيق
- تحسين الأداء والأمان

05 API

RESTful API

واجهة برمجية للسماح للتطبيقات بالتواصل مع الـ Backend

تستخدم في

- ربط Frontendبالـ Backend
- تطبيقات الجوال
- التكامل مع أنظمة أخرى

06 Authentication

JWT

آلية مصادقة آمنة تعتمد على JSON Web Token JSON Web Token

تستخدم في

- تسجيل الدخول
- إدارة الجلسات
- حماية الـ APIs

الخلاصة

تعتمد الـ Backendعلى مجموعة من التقنيات التي تعمل معًا لبناء أنظمة قوية، آمنة، وقابلة للتوسع اختيار التقنية المناسبة يعتمد على متطلبات المشروع وحجمه

قاعدة البيانات في ال Backend

ما هي قاعدة البيانات؟

- مجموعة منظمة من البيانات المرتبطة
- يتم تخزينها إلكترونيًا وتدار بواسطة نظام إدارة قواعد البيانات DBMS
- تتيح تخزين البيانات، استرجاعها، تحديثها وحذفها بكفاءة وأمان

وظائف قاعدة البيانات في الـ Backend

1 تخزين البيانات
حفظ بيانات المستخدمين، المنتجات، الطلبات إلخ

2 استرجاع البيانات
البحث وجلب البيانات عند الحاجة بسرعة

3 تحديث البيانات
تعديل البيانات عند حدوث تغييرات

4 حذف البيانات
إزالة البيانات غير المطلوبة بأمان

5 ضمان الأمان والسلامة
حماية البيانات، تسجيل الدخول، الصلاحيات، والتشفير

كيف تتفاعل الـ Backend مع قاعدة البيانات؟

قاعدة البيانات في ال Backend

أنواع قواعد البيانات

1 قواعد بيانات علائقية SQL

تخزن البيانات في جداول مترابطة صفوف وأعمدة

مثال:

PostgreSQL, SQL MySQL

id	name	email	age
1	Ali	ali@mail.com	22
2	Sara	sara@mail.com	20
3	Omar	omar@mail.com	25

مناسبة للبيانات المنظمة والعلاقات المعقدة

2 قواعد بيانات غير علائقية NoSQL

تخزن البيانات بطرق مختلفة مستندات، مفاتيح-قيم، رسائل

مثال:

MongoDB, Redis

مناسبة للبيانات غير المنظمة والتوسع الأفقي

أمثلة على أنظمة قواعد البيانات

ملاحظة مهمة

- اختيار نوع قاعدة البيانات يعتمد على طبيعة المشروع
- تصميم قاعدة بيانات جيدة = أداء أفضل للتطبيق
- يجب دائمًا عمل نسخة احتياطية للبيانات

معمارية الـ Backend

اختيار المعمارية المناسبة يعتمد على حجم المشروع، عدد المستخدمين، متطلبات الأداء، الميزانية، وفريق العمل

الأنماط المعمارية الشائعة

Monolithic Architecture

- تطبيق كامل في وحدة واحدة
- بسيط للبداية، لكنه يصعب توسيع وصيانته مع الكبر
- مناسب للتطبيقات الصغيرة

Microservices Architecture

- تطبيق مقسم إلى خدمات صغيرة
- مستقلة تواصل عبر APIs
- مرونة عالية وقابلية للتوسع
- مناسب للتطبيقات الكبيرة والمعقدة

Serverless Architecture

- تشغيل الكود دون إدارة خوادم مثل AWS Lambda
- يدفع حسب الاستخدام فقط
- يوفر في التكلفة وسهل التوسع

أفضل الممارسات في الـ Backend

الأمن
Security

- تحقق من صحة Validation
- استخدم المصادقة والتفويض وضمان Authorizationبشكل
- تشفير البيانات
- حماية من هجمات SQL Injectionو XSS

كتابة كود نظيف وقابل للصيانة
Clean Code

- اتبع مبادئ SOLID
- استخدام أسماء واضحة للدوال والمتغيرات
- تقسيم الكود إلى وحدات Modulesصغيرة

التعامل مع قاعدة البيانات بكفاءة

- استخدام Query Builderأو ORMحسب الحاجة
- تحسين الاستعلامات Indexes, Joins
- تجنب جلب البيانات الزائدة

استخدام التخزين المؤقت
Caching

- استخدم الكاش لتقليل العمل على قاعدة البيانات
- أدوات مثل Memcachedو Redis

تصميم API جيد

- RESTful API conventions
- استخدام أسماء موارد واضحة Nouns
- إرجاع رموز حالة HTTPمناسبة
- توثيق الـ API Swagger / OpenAPI

المراقبة والتسجيل
Logging & Monitoring

- سجل الأخطاء والمعلومات Logging
- استخدم أدوات المراقبة Monitoringمثل Prometheus, Grafana
- تنبيه عند حدوث مشاكل

قابلية التوسع
Scalability

- تصميم النظام ليتحمل النمو المستقبلي
- Load Balancingواستخدام تقنيات مثل Auto Scaling
- فصل الخدمات عند الحاجة

الاختبار
Testing

- كتابة اختبارات وحدات Unit Tests واختبارات تكامل Integration Tests
- اختبار الحالات المختلفة قبل النشر
- أدوات مثل Jest, PyTest, Postman

عملية تطوير الـ Backend

تحليل المتطلبات

- فهم متطلبات النظام
- تحديد الوظائف الأساسية
- تحديد القيود والمخاطر

تصميم النظام

- تصميم المعمارية
- تصميم قاعدة البيانات
- تصميم الـ API

التطوير

- إعداد البيئة
- كتابة الكود
- تطبيق أفضل الممارسات

الاختبار

- اختبار الوحدات
- اختبار التكامل
- اختبار الأداء والأمان

النشر

- إعداد بيئة الإنتاج
- نشر التطبيق
- مراقبة الأداء

الصيانة والتطوير المستمر

- إصلاح الأخطاء
- تحسين الأداء
- إضافة ميزات جديدة

أدوات وتقنيات شائعة في تطوير الـ Backend

اللغات

- Python
- Node.js
- Java
- Go

Frameworks

- Django
- Express.js
- Spring Boot
- FastAPI

قواعد البيانات

- PostgreSQL
- MySQL
- MongoDB
- Redis

أدوات التطوير

- Git
- Docker
- Postman
- Swagger

النشر والسحابة

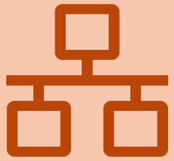
- AWS
- Google Cloud
- Microsoft Azure
- Heroku

عملية تطوير الـ Backend

تحليل المتطلبات

- فهم متطلبات النظام
- تحديد الوظائف الأساسية
- تحديد القيود والمخاطر

تصميم النظام



- تصميم المعمارية
- تصميم قاعدة البيانات
- تصميم الـ API

التطوير

- إعداد البيئة
- كتابة الكود
- تطبيق أفضل الممارسات

الاختبار

- اختبار الوحدات
- اختبار التكامل
- اختبار الأداء والأمان

النشر

- إعداد بيئة الإنتاج
- نشر التطبيق
- مراقبة الأداء

الصيانة والتطوير المستمر

- إصلاح الأخطاء
- تحسين الأداء
- إضافة ميزات جديدة

أدوات وتقنيات شائعة في تطوير الـ Backend

اللغات

Python

Node.js

Java

Go

Language C#

لغة برمجة قوية تُستخدم بكثرة في تطوير تطبيقات Backend، خصوصًا مع منصة .NET.

تُستخدم في:

- تطوير Backend
- بناء Web APIs
- تطبيقات المؤسسات والشركات
- التعامل مع قواعد البيانات

البيانات

Po

My

Mo

Re

Framework ASP.NET Core / .NET

إطار عمل من Microsoft لبناء تطبيقات ويب وواجهات API قوية، آمنة وسريعة.

تُستخدم في:

- بناء REST APIs
- أنظمة تسجيل الدخول والصلاحيات
- ربط Frontend مع Backend
- تطبيقات كبيرة وقابلة للتوسع

السحابة

aws A

Goo

Micr

He

عملية تطوير الـ Backend

مثال مبسط لسير العمل

فكرة
تحديد مشكلة أو حاجة معينة

متطلبات
جمع وتحليل المتطلبات وتوثيقها

تطوير
كتابة الكود وتطبيق الوظائف

اختبار
اختبار كافة الوظائف والتأكد من جودتها

نشر
نشر التطبيق على الخادم ليصبح متاحاً للمستخدمين

مراقبة وتحسين
متابعة الأداء وجمع الملاحظات والتحسين باستمرار

نصائح لنجاح عملية التطوير

وضوح المتطلبات
فهم واضح لما يحتاجه النظام يساعد على توفير الوقت والجهد

التعاون والتواصل
تواصل مستمر بين جميع أفراد الفريق

كتابة كود نظيف
كود منظم وسهل القراءة أسهل في الصيانة والتطوير

الاختبار المستمر
الاختبار المستمر يقلل الأخطاء ويضمن جودة أعلى

المراقبة والتحسين
مراقبة الأداء وجمع التحليلات يساعد على التحسين المستمر

أمن الـ Backend

أمن Backendيهدف إلى حماية النظام والبيانات من الهجمات والاختراقات، وضمان السرية، السلامة، والتوفر CIA Triad

أهم التهديدات الشائعة

ممارسات أمان الـ Backend

المصادقة والتفويض	حماية البيانات	التحقق من المدخلات	مراقبة وتسجيل	الحماية من الهجمات
- استخدام JWTأو OAuth 20 -تخزين كلمات المرور بطريقة آمنة bcrypt -تطبيق مبدأ أقل صلاحية Least Privilege	- تشفير البيانات الحساسة Encryption -استخدام HTTPSدائماً - إخفاء البيانات الحساسة في الاستجابات	- التحقق من صحة جميع المدخلات - استخدام Prepared Statements لاستعلامات SQL -تطهير المدخلات من الأكواد الضارة	- تسجيل جميع العمليات المهمة - مراقبة الأنشطة المشبوهة - تنبيهات فورية عند حدوث عطل أو هجوم	- Rate Limiting للدعم الطلبات - WAFلحماية من الهجمات - تحديث التبعيات باستمرار - إجراء اختبارات أمان دورية

أمن الـ Backend

مثال: استخدام JWT في المصادقة

```
const jwt = require('jsonwebtoken');

// إنشاء التوكن
const token = jwt.sign(
  { userId: 1, role: 'user' },
  process.env.SECRET_KEY,
  { expiresIn: '1h' }
);
```

التحقق من التوكن

```
jwt.verify(token,
 process.env.SECRET_KEY,
 (err, decoded) => {
 if (err) return res.status(401).send
   ('Unauthorized');
 req.user = decoded;
 next();
});
```

أدوات وتقنيات أمنية مهمة

WAF
جدار حماية Web Application Firewall

bcrypt
تشفير كلمات المرور

OAuth 20
تفويض الوصول Authorization

JWT
المصادقة Authentication

Rate Limiting
تحديد عدد الطلبات

Logging
تسجيل الأحداث

Helmetjs
أمان HTTP Headers في Nodejs

OWASP
معايير أمان التطبيقات

أمن الـ Backend

Defense in Depth

قائمة تحقق لأمان الـ Backend

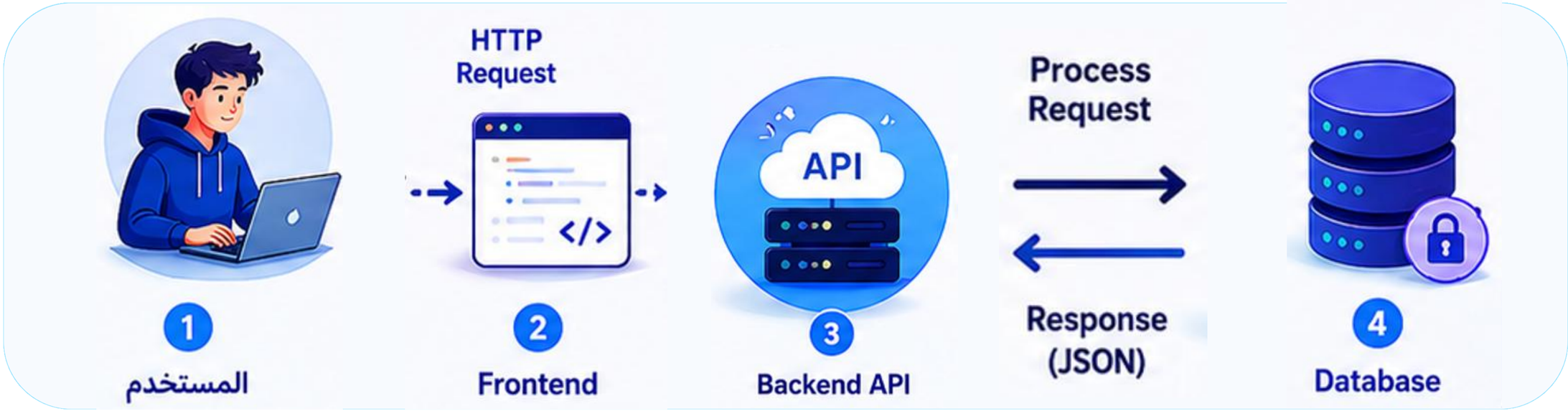
- ✓ تطبيق مبدأ أقل صلاحية
- ✓ تسجيل الأحداث والمراقبة المستمرة
- ✓ تحديد عدد الطلبات **Rate Limiting**
- ✓ تحديث التبعيات والـ **Libraries**
- ✓ إجراء اختبارات اختراق دورية
- ✓ استخدام **HTTPS** في جميع الاتصالات
- ✓ تشفير كلمات المرور **bcrypt**
- ✓ استخدام **JWT** أو **OAuth** للمصادقة
- ✓ التحقق من جميع المدخلات
- ✓ الاستخدام **Prepared Statements** في **SQL**

API في Backend

ما هي الـ API (Application Programming Interface)؟

هي واجهة تسمح للتطبيقات المختلفة بالتواصل مع الـ Backend وطلب البيانات أو إرسال بيانات إليه بطريقة منظمة وآمنة

كيف تعمل الـ API؟



أنواع الـ API؟

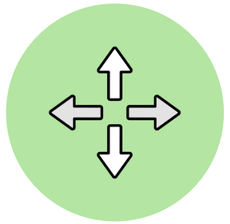
Public API
متاحة للجميع عبر الإنترنت

Private API
تستخدم داخل الشركة أو بين الأنظمة الداخلية

Third-party API
تتيح التكامل مع خدمات خارجية مثل Google, PayPal

فوائد استخدام الـ API

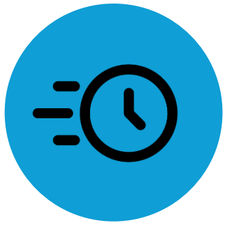
ربط تطبيقات مختلفة



مرونة وقابلية للتوسع

أمان وتحقق من الصلاحيات

إعادة استخدام الكود



سهولة وسرعة التكامل

أشهر طرق HTTP في الـ API

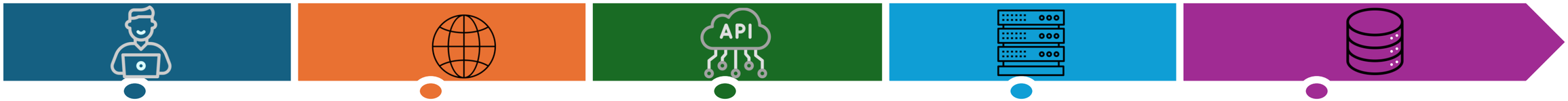
GET	/users	جلب البيانات قراءة
POST	/users	إرسال بيانات جديدة إنشاء
PUT	/users/1	تحديث بيانات موجودة بالكامل
PATCH	/users/1	تحديث جزء من البيانات
DELETE	/users/1	حذف بيانات

مثال استجابة JSON

```
{
  "id": 1,
  "name": "Ali",
  "email": "ali@mail.com",
  "role": "Student",
  "isActive": true
}
```

ما هو الـ API؟

هو مجموعة من القواعد والبروتوكولات التي تحدد كيف يمكن لتطبيق ما طلب بيانات أو خدمات من تطبيق آخر الـ API يعتبر جسر التواصل بين الأنظمة المختلفة



الاستجابة
يُعيد الـ API البيانات عادة بصيغة JSON إلى العميل

الـ Backend
ينفذ المنطق التجاري، يتعامل مع قاعدة البيانات أو الخدمات

الـ API
يستقبل الطلب، يتحقق من صحته، ويعالجه وفقًا للمنطق المحدد

الإنترنت
يتم إرسال الطلب عبر بروتوكول HTTP/HTTPS

العميل Client
واجهة المستخدم أو تطبيق خارجي يرسل طلبًا إلى الـ API

لماذا نستخدم الـ API؟

فصل الواجهة
فصل واجهة المستخدم عن منطق العمل Frontend / Backend

إعادة الاستخدام
نفس الـ API يمكن استخدامه من تطبيقات متعددة ويب، موبايل، طرف ثالث

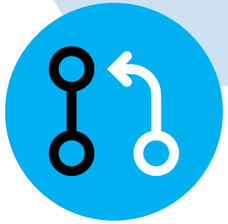
الأمان والتحكم
التحقق من الهوية وتقنين الصلاحيات للتحكم في الوصول إلى البيانات

المرونة والتوسع
تسهل إضافة ميزات جديدة وتطوير النظام دون تأثير على العملاء

توحيد البيانات
توفير بيانات موحدة ومتسقة لجميع التطبيقات والعملاء

ما هو الـ API؟

تصميم API الجديد



استخدم الإصدارات في الـ APIمثل: api/v1/

إرجاع رموز حالة HTTPمناسبة وواضحة

دعم الفلترة والترتيب والبحث والتقييد

استخدم أسماء موارد منطقية جمع مثل: users /

استخدم الطرق الصحيحة لكل عملية

أمن الـ API

التحقق من صحة المدخلات لمنع الحقن والهجمات

تحديد عدد الطلبات Rate Limiting للحماية من الإساءة

التحقق من الصلاحيات قبل تنفيذ أي عملية حساسة

التحقق من هوية المستخدم باستخدام JWT Tokenأو API Key

استخدام HTTPS دائمًا لتشفير الاتصالات

ما هو الـ Auth؟

01
إدخال البيانات
المستخدم يدخل البريد وكلمة المرور

02
إرسال الطلب
يُرسل الطلب إلى خادم المصادقة

03
التحقق
يتم التحقق من صحة البيانات

04
إصدار Token
إذا كانت البيانات صحيحة، يتم إصدار Tokenمثل JWT

05
الوصول
يُستخدم الـ Token للوصول إلى الموارد المحمية

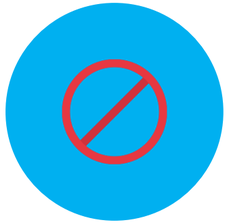
أفضل ممارسات الأمان في Auth

استخدام HTTPS دائمًا لتشفير الاتصالات

تخزين كلمات المرور مع Hash قوي مثل bcrypt

تحديد مدة صلاحية وتحديثه عند الحاجة

لتجديد الوصول بأمان



تسجيل الخروج يجب إبطال التوكن Logout / Invalidate

تطبيق مبدأ أقل صلاحية Least Privilege

ما هو الـ Auth؟

الفرق بين Authentication و Authorization

Authentication (المصادقة)

التحقق من هوية المستخدم
- سؤال: من أنت؟ - مثال: تسجيل الدخول بكلمة مرور

Authorization (التفويض)

- تحديد ما يسمح للمستخدم بفعله
- سؤال: ماذا يسمح لك أن تفعل؟
- مثال: مستخدم عادي لا يمكنه حذف المستخدمين

الأدوار والصلاحيات Roles & Permissions

Permissionالصلاحيات

تحديد الأفعال المسموح بها داخل النظام
مثال: create, read, update, delete

Rolesالأدوار

تجميع المستخدمين في مجموعات تملك صلاحيات محددة
مثال: Admin – Teacher – Student – User

تطبيقات سطح المكتب VS تطبيقات الويب

تطبيقات سطح المكتب (Desktop Applications)	وجه المقارنة ⚖	تطبيقات الويب(Web Applications)
تعمل على جهاز المستخدم مباشرة. يتم تثبيتها وتعمل كبرنامج مستقل.	مكان التشغيل	تعمل عبر المتصفح (Chrome, Edge, Firefox...). لا تحتاج تثبيت على الجهاز.
أداء قوي وسريع تستفيد من موارد الجهاز بالكامل)المعالج، الذاكرة، كرت الشاشة	الأداء	**يعتمد الأداء على المتصفح والسيرفر** قد يكون أبطأ في العمليات الثقيلة. يعتمد على جودة الاتصال بالإنترنت.
إمكانية العمل بدون إنترنت تعمل حتى في حالة عدم اتصال بالشبكة. مناسبة للبيئات التي لا يوجد فيها إنترنت.	الاتصال بالإنترنت	**تحتاج اتصال بالإنترنت** معظم التطبيقات تحتاج إنترنت للعمل.)باستثناء بعض التقنيات الحديثة مثل .(PWA
الوصول للأجهزة المحلية تستطيع التعامل مع الطابعة، الباركود، الكاميرا، ملفات الجهاز، وغيرها بسهولة.	الوصول للأجهزة	**الوصول للأجهزة محدود** الوصول للأجهزة المحلية محدود أو يحتاج إضافات خاصة)مثل .(Web USB
الأمان والبيانات البيانات تبقى محلياً على الجهاز. تحكم كامل في تخزين البيانات.	الأمان والبيانات	**الأمان والبيانات** البيانات غالباً مخزنة على السيرفر. تحتاج سياسات أمان قوية لحمايتها.
التثبيت والتحديث تحتاج إلى تثبيت على كل جهاز. التحديث يحتاج توزيع نسخة جديدة.	التثبيت والتحديث	**لا تحتاج تثبيت** تعمل مباشرة من المتصفح. التحديث يتم تلقائياً على السيرفر.
الاستخدام المناسب برامج تحتاج أداء عالٍ، أو تعمل بدون إنترنت، أو تتعامل مع أجهزة ومكونات محلية.	الاستخدام المثالي	**الاستخدام المناسب** أنظمة متعددة المستخدمين، الوصول من أي مكان، سهولة التحديث، الوصول من أي جهاز مختلفة.

أساسيات قواعد البيانات و SQL

SQL (Structured Query Language)هي لغة قياسية للتعامل مع قواعد البيانات العلائقية.
تُستخدم لإنشاء الجداول، إدخال البيانات، الاستعلام، التحديث، والحذف.

أنواع قواعد البيانات

RDBMS علائقية

تعتمد على الجداول والعلاقات (SQL)

MySQL • PostgreSQL • Oracle • SQL Server

NoSQL غير علائقية

مرنة لتخزين البيانات غير المنظمة

MongoDB • Cassandra • Redis

Columnar عمودية

تُحسن أداء التحليلات والقراءة الكبيرة

ClickHouse • Amazon Redshift

Key-Value

تخزن البيانات كمفتاح وقيمة بسيطة

Redis • Amazon DynamoDB

Graph DB رسوم بيانية

تخزن البيانات كعقد وروابط

Amazon Neptune

Document DB وثائقية

تخزن البيانات في مستندات JSON/BSON

MongoDB • CouchDB

مكونات نظام قاعدة البيانات

البيانات

حقائق ومعلومات تُخزن في الجداول
يجب أن تكون دقيقة ومتسقة

الجداول

تنظيم البيانات في صفوف وأعمدة
كل جدول يمثل كياناً واحداً

المفاتيح

PK: يحدد كل سجل بشكل فريد
FK: يربط بين الجداول

العلاقات

تربط الجداول ببعضها
واحد لواحد /متعدد لمتعدد

الفهارس

تُحسن سرعة البحث والاستعلام
تُنشأ تلقائياً على الأعمدة المستخدمة

القيود

ضمان صحة البيانات وتكاملها
مثل: NOT NULL, UNIQUE, CHECK

تطبيع قواعد البيانات — Normalization

تنظيم البيانات لتقليل التكرار والحفاظ على الاتساق

قبل التطبيع
Unnormalized

ID	Name	Phones
1	أحمد	0551111111, 0502222222
2	سارة	0563333333
3	محمد	0574444444, 0585555555

- تكرار البيانات
- صعوبة التعديل
- هدر في المساحة

1NF
القيمة الذرية

ID	Name	Phone
1	أحمد	0551111111
1	أحمد	0502222222
2	سارة	0563333333
3	محمد	0574444444

- إزالة التكرار داخل الخلية
- كل قيمة ذرية غير قابلة للتجزئة

2NF
الاعتماد الكامل

UserID	Name
1	أحمد
2	سارة
3	محمد

- فصل البيانات التي تعتمد جزئياً على المفتاح
- جداول منفصلة

3NF
لا يوجد اعتماد انتقالي

UserID	PhoneID
1	2
2	3
3	4,5

- فصل الاعتماد الانتقالي
- قاعدة بيانات منظمة وفعّالة

ORM (Object Relational Mapping)

تقنية تربط بين الكائنات في البرمجة والجداول في قاعدة البيانات.
تسهل عمليات CRUDباستخدام كائنات بدلاً من كتابة SQLمباشرة.

أمثلة على أدوات الـ ORMالشائعة

ميزات ORM:

- ✓ تسهيل التعامل مع قاعدة البيانات .
- ✓ تقليل كتابة SQLالمتكرر .
- ✓ زيادة الإنتاجية وسرعة التطوير
- ✓ دعم الكائنات والعلاقات والتجريد .

عيوب ORM

- ✓ قد يؤثر على الأداء في بعض الحالات.
- ✓ يصعب تنفيذ استعلامات معقدة جداً.
- ✓ يعتمد على أداة خارجية (.Lock-in)
- ✓ يحتاج لفهم جيد للأداة وإعدادها.

أنواع عمليات SQL

DDL
Data Definition Language

إنشاء وتعديل وحذف الكائنات (الجداول، الفهارس)

الأوامر

CREATE, ALTER
DROP, TRUNCATE

DML
Data Manipulation Language

التعامل مع البيانات داخل الجداول

الأوامر

INSERT, UPDATE
DELETE

DQL
Data Query Language

استعلام واسترجاع البيانات

الأوامر

SELECT

DCL
Data Control Language

التحكم في الصلاحيات

الأوامر

GRANT, REVOKE

TCL
Transaction Control Language

التحكم في المعاملات

الأوامر

COMMIT, ROLLBACK
SAVEPOINT

بنية جملة Select مع اكثر الأوامر استخداماً

SELECT — استرجاع البيانات من جدول

```
SELECT id, name, email
FROM users
WHERE age > 18
ORDER BY name ASC;
```

INSERT — إدخال بيانات جديدة

```
INSERT INTO users (name, email,
age)
VALUES ('Ali', 'ali@ex.com', 25);
```

UPDATE — تحديث بيانات موجودة

```
UPDATE users
SET age = 26
WHERE id = 1;
```

DELETE — حذف بيانات من جدول

```
DELETE FROM users
WHERE id = 1;
```

CREATE — إنشاء جدول جديد

```
CREATE TABLE users (
  id INT PRIMARY KEY,
  name VARCHAR(100)
);
```

ALTER — تعديل بنية جدول موجود

```
ALTER TABLE users
ADD COLUMN phone
VARCHAR(20);
```

بنية جملة SELECT

SELECT	تحديد الأعمدة المراد عرضها
FROM	تحديد الجدول
WHERE	تصفية الصفوف بناءً على شرط
GROUP BY	تجميع النتائج حسب عمود
HAVING	تصفية النتائج بعد التجميع
ORDER BY	ترتيب النتائج تصاعدياً أو تنازلياً
LIMIT	تحديد عدد الصفوف المعروضة

أنواع عمليات SQL

أنواع الربط بين الجداول JOINs

INNER JOIN	إرجاع السجلات المتطابقة فقط في كلا الجدولين
LEFT JOIN	كل سجلات الجدول الأيسر والمتطابقة من الأيمن
RIGHT JOIN	كل سجلات الجدول الأيمن والمتطابقة من الأيسر
FULL JOIN	جميع السجلات من الجدولين

مثال على INNER JOIN

```
SELECT u.id, u.name, o.order_date, o.total
FROM users u
INNER JOIN orders o ON u.id = o.user_id
WHERE o.total > 100
ORDER BY o.order_date DESC
LIMIT 10;
```

Aggregate Functions

COUNT() — عدد الصفوف

```
SELECT COUNT(*)
FROM users;
```

SUM() — مجموع القيم العددية

```
SELECT SUM(salary)
FROM employees;
```

AVG() — متوسط القيم العددية

```
SELECT AVG(age)
FROM users;
```

MIN() — أصغر قيمة

```
SELECT MIN(price)
FROM products;
```

MAX() — أكبر قيمة

```
SELECT MAX(price)
FROM products;
```

التحكم في الإصدارات والتعاون Git & GitHub

هو نظام للتحكم في الإصدارات **Version Control System**, يسمح بتتبع التغييرات في الكود، والرجوع لأي نسخة سابقة, والتعاون مع فريق العمل بكفاءة وأمان.

لماذا نستخدم Git؟

التعاون الفعال

الرجوع لأي نسخة سابقة

حماية الكود من الفقدان

تجربة ميزات جديدة

سرعة وأداء عالي جداً

كيف يعمل Git؟

جلب التحديثات

رفع للمستودع

إنشاء Commit

إضافة للتجهيز

تعديل الملفات

التحكم في الإصدارات والتعاون Git & GitHub

مستويات العمل في Git

Working Directory	Staging Area	Local Repository	Remote Repository
منطقة العمل	منطقة التجهيز	المستودع المحلي	المستودع البعيد
تعديلات على الملفات محلياً	التغييرات التي تم تجهيزها للحفظ	التغييرات التي تم حفظها (Commits)	مثل GitHub لتخزين ومشاركة المشروع

أفضل الممارسات

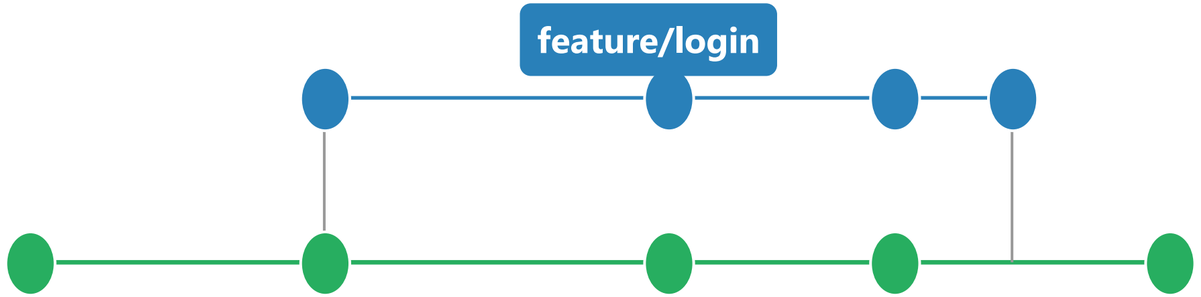
- اكتب رسائل Commit واضحة ومعبرة.
- راجع الكود قبل دمجه .(Code Review)
- استخدم فروع لكل ميزة جديدة.
- احذف الفروع غير المستخدمة.
- قم بـ Pull قبل Push لتجنب التعارضات.

التحكم في الإصدارات والتعاون Git & GitHub

أهم الأوامر الأساسية في Git

`git init`	إنشاء مستودع جديد
`git clone <url>`	استنساخ مستودع موجود
`git status`	عرض حالة التغييرات
`git add <file>`	إضافة ملف للتجهيز
`git add .`	إضافة جميع الملفات للتجهيز
`git commit -m "msg"`	حفظ التغييرات مع رسالة
`git push origin<branch>`	رفع التغييرات إلى المستودع البعيد
`git pull origin <branch>`	جلب التغييرات من المستودع البعيد
`git branch`	عرض الفروع
`git checkout <branch>`	الانتقال إلى فرع آخر

التعامل مع الفروع (Branches)



- **main**: الفرع الرئيسي (main)النسخة المستقرة من المشروع.
- إنشاء فرع جديد للتطوير على ميزة جديدة.
- دمج الفرع بعد الانتهاء من تطوير الميزة.
- تحديث الفرع الرئيسي بآخر التغييرات.

الاستخدام يومياً, والممارسة المستمرة هي أفضل طريقة لتصبح محترفاً فيه.

التحكم في الإصدارات والتعاون Git & GitHub

ما هو "المستودع" Repository؟

هو مجلد أو مساحة تخزين مخصصة لمشروع برمجي معين يحتوي هذا المجلد على كل ما يخص المشروع: ملفات الكود، وسجل كامل بالتعديلات التي حدثت عليه، وملاحظات الفريق، وأكثر

المستودعات بشكل عام تنقسم إلى نوعين رئيسيين

المستودع المحلي Local Repository

طبيعته: هو مجلد عادي على القرص الصلب لجهازك، لكنه يحتوي على مجلد مخفي اسمه gitويقوم Git من خلاله بتتبع كل التغييرات

ميزاته: تتيح لك هذه النسخة المحلية العمل في بيئة معزولة تماماً يمكنك إضافة كود جديد، تجربة أفكار، أو الرجوع إلى نسخة قديمة من المشروع دون الحاجة إلى اتصال بالإنترنت

المستودع البعيد Remote Repository

طبيعته: هذا هو المستودع الموجود على خادم سيرفر بعيد، ويمكن أن يكون إما على السحابة مستأجر أو على خادم خاص بك داخل الشركة

ميزاته: هو أداة المشاركة والتعاون مع فريقك بعد أن تنتهي من العمل على نسختك المحلية، يمكنك "رفع" أو "سحب" التعديلات لتحديث نسختك المحلية

التحكم في الإصدارات والتعاون Git & GitHub

الميزة	المستودع على خادم الزبون سيرفر خاص بك	المستودع على خادم مستأجر مثل GitHub
مكان الاستضافة	خوادم شركتك داخل مبناك	خوادم شركة خارجية
التحكم والخصوصية	تحكم كامل 100%، مثالي للمشاريع شديدة الحساسية	خصوصية محكومة بسياسة الشركة
التكلفة	تحتاج إلى شراء وصيانة خادم (سيرفر) وتوفير طاقم فني للادارة	مجاني للمشاريع مفتوحة المصدر، خطط مدفوعة للميزات المتقدمة في الفرق والشركات
الصيانة	أنت من يقوم بعمل backups، ويحل المشاكل	لا شيء عليك، GitHub يتولى كل شيء
سهولة الاستخدام	يحتاج إلى إعداد البنية التحتية وتثبيت وإدارة برامج	سهل جداً بواجهة مستخدم رسومية وأدوات جاهزة مثل للاختبار الآلي والنشر

خلاصة

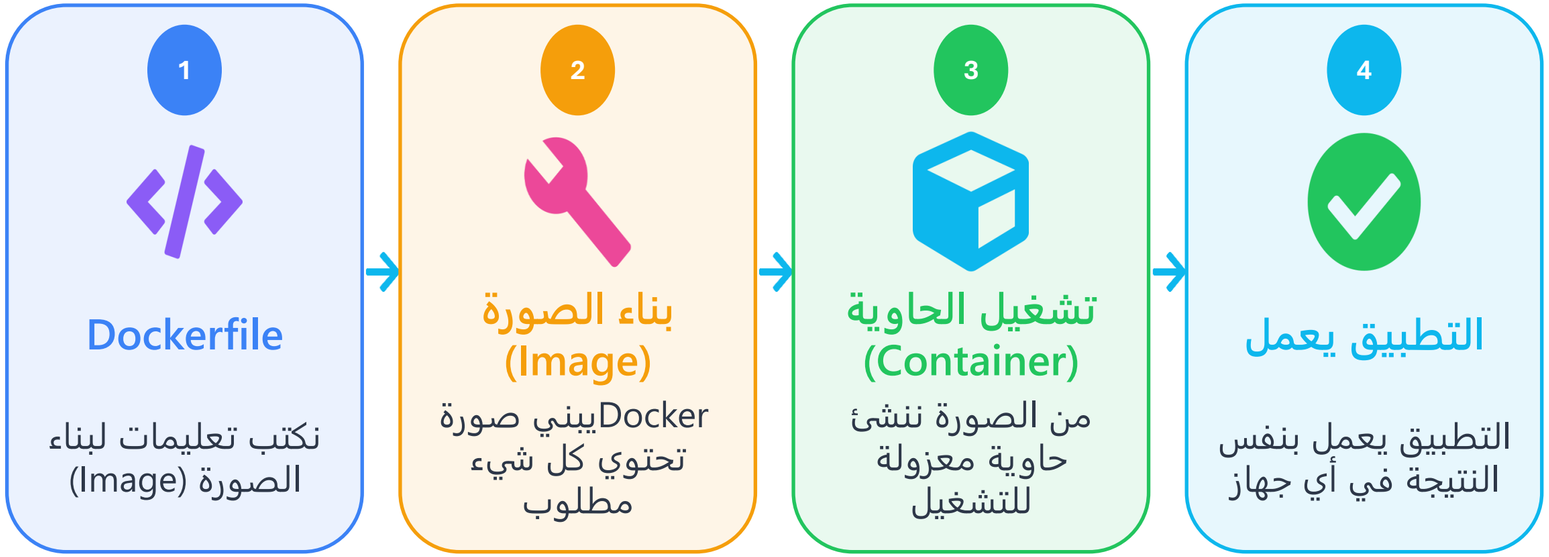
GitHub هو الخيار الأسهل والأشهر والأرخص (مجاني في الغالب)، مناسب لـ 99% من الاستخدامات اليومية، من المشاريع الشخصية إلى الشركات الكبيرة

المستودع على خادم خاص بك هو خيار السيطرة المطلقة، وتختاره عندما تكون البيانات حساسة جداً أو عندما تريد الاستقلال التام عن خدمات الطرف الثالث

الحاويات ونشر التطبيقات Docker

هي منصة مفتوحة المصدر تستخدم تقنية الحاويات لحزم التطبيقات مع جميع تبعياتها في وحدة واحدة وخفيفة وقابلة للتشغيل في أي بيئة.

كيف يعمل Docker؟



لماذا نستخدم Docker

سرعة النشر والتشغيل

نشر التطبيقات في ثوانٍ بدلاً من ساعات مع بيئات جاهزة

توافق بيئي

يعمل في أي مكان —محلياً، في السحابة، أو على أي خادم

عزل التطبيقات

كل تطبيق معزول تماماً عن الآخر بدون تعارضات

تقليل المشاكل

"يعمل على جهازي "لم تعد مشكلة —البيئة متطابقة دائماً

سهولة التوسع

توسيع التطبيقات أفقياً بكل سهولة مع docker-compose وKubernetes

مكونات Docker الأساسية

Dockerfile

ملف نصي ثابت يحتوي على تعليمات لبناء الصورة .يحدد كل شيء من نظام التشغيل إلى الحزم والأوامر.

Image (الصورة)

قالب ثابت Read-only يحتوي على كل ما يحتاجه التطبيق .يُنشأ من Dockerfile ويُخزَّن لإعادة الاستخدام.

Container (الحاوية)

نسخة تشغيل من الصورة —قابل للقراءة والكتابة Read-Write. يعمل بعزل تام وهو المكان الفعلي لتشغيل التطبيق.

Docker Hub

مستودع سحابي لمشاركة الصور الجاهزة مثل nginx, redis, node.يمكن سحب صور رسمية أو نشر صورك الخاصة.

استخدامات Docker

أوامر Dockerالأساسية

```
docker --version
```

عرض إصدار Docker

```
docker build -t <name> .
```

بناء صورة من Dockerfile

```
docker images
```

عرض الصور المحلية

```
docker run -p <port>:<port>
<image>
```

تشغيل حاوية من صورة وربط منفذ

```
docker ps
```

عرض الحاويات قيد التشغيل

```
docker stop <container_id>
```

إيقاف حاوية

```
docker rm <container_id>
```

حذف حاوية

```
docker rmi <image_id>
```

حذف صورة

```
docker pull <image>
```

تحميل صورة من Docker Hub

Docker vs Virtual Machine

Virtual Machine

App 1 | App 2 | App 3

Bins/Libs | Bins/Libs | Bins/Libs

Guest OS | Guest OS | Guest OS

Hypervisor

Infrastructure

VS

Docker Containers

App 1 | App 2 | App 3

Bins/Libs | Bins/Libs | Bins/Libs

Docker Engine

Infrastructure

- ✓ خفيفة وسريعة
- ✓ تستهلك موارد أقل
- ✓ إقلاع سريع جداً
- ✓ قابلة للنقل بسهولة

الخلاصة

Dockerيجعل التطبيقات تعمل في أي مكان بنفس الطريقة،
بسرعة وكفاءة، ويُسهّل النشر والتوسع
"Build once, Run anywhere"

بيئة العمل والخادم Linux

🚀 النظام الذي يشغّل الإنترنت، السيرفرات، السحابة... وكل ما هو مهم في عالم البرمجة اليوم!

هو نظام تشغيل مفتوح المصدر، قوي وآمن جداً، يُستخدم لتشغيل السيرفرات، السحابة، أدوات المطورين، والملايين من الأجهزة.

⭐ لماذا Linux مهم؟

🛡️ آمن جداً
مستهدف أقل للبرمجيات الخبيثة

🚀 مستقر وسريع
يعمل لسنوات بدون إعادة تشغيل

🔧 قابل للتخصيص
يمكن تعديل كل شيء فيه

💰 مجاني ومفتوح المصدر
الكود متاح للجميع

👥 مجتمع ضخم
مطورون من حول العالم

تحديات Linux

- ❑ حل بعض المشاكل يحتاج بحث
- ❑ دعم بعض البرامج/الألعاب أقل
- ❑ يتطلب بعض التعلم

هل هو مفتوح المصدر؟

- ✓ الكود متاح للدراسة والتعديل
- ✓ يمكن تحسينه وبناء توزيعات جديدة
- ✓ يساهم فيه مطورون من كل العالم
- ✓ شفافية +أمان +تطور مستمر

توزيعات Linux الشهيرة

Ubuntu	Debian	Fedora	CentOS Stream Rocky Linux	Arch Linux
أسهل للمبتدئين مناسبة للاستخدام اليومي والسيرفرات	مستقر جداً أساس كثير من التوزيعات	أحدث التقنيات للمطورين	موجهة للسيرفرات والشركات	مرنة وقوية للمستخدمين المتقدمين

بيئة العمل والخادم Linux

🚀 النظام الذي يشغّل الإنترنت، السيرفرات، السحابة... وكل ما هو مهم في عالم البرمجة اليوم!

لماذا منتشر في السيرفرات وليس بين العامة؟

منتشر في السيرفرات

- ❑ مستقر ويعمل بدون إعادة تشغيل
- ❑ آمن وسهل التحكم
- ❑ يستهلك موارد قليلة
- ❑ يحمل أعباء ضخمة بكفاءة
- ❑ يدعمه مزودو السحابة و DevOps

ليس منتشر بين العامة

- ❑ واجهة أقل جاذبية
- ❑ يحتاج بعض التعلم
- ❑ برامج أقل جاهزية مقارنة بـ Windows
- ❑ ليس موجهاً للاستخدام المنزلي

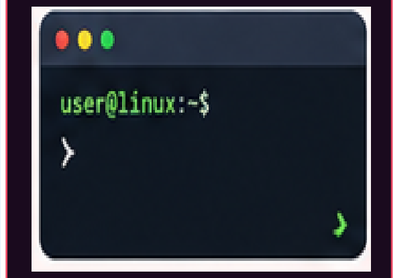
🏆 الأساس في السيرفرات، السحابة، DocOps، الأمن السيبراني، الذكاء الاصطناعي، والأنظمة المدمجة.

واجهات المستخدم

مع واجهة (GUI)

- أسهل للمبتدئين
- مناسب للديسكتوب
- يستهلك موارد أكثر

بدون واجهة (CLI)



- أخف وأسرع
- مناسب للسيرفرات
- لا يستهلك موارد

لماذا هو مهم للبرمجة والخوادم؟Linux

لماذا ينتشر في السيرفرات؟

⏱️ مستقر لفترات طويلة

يعمل لفترات طويلة بدون إعادة تشغيل.

🔒 آمن وقابل للإدارة عن بعد

صلاحيات دقيقة وإدارة عن بعد بسهولة.

💾 يستهلك موارد أقل

أداء عالي باستهلاك أقل للمعالج والذاكرة.

☁️ أساس شائع في Cloud و Hosting

يدعم البنية التحتية الحديثة والمرنة.

أين نراه في الواقع؟ 🌍

Web Servers	Databases	Containers	Cloud VMs	CI/CD Runners	Networking
Nginx, Apache	MySQL, PostgreSQL	Docker, Kubernetes	AWS, GCP, Azure	GitHub Actions, GitLab CI	Routers & Firewalls

لماذا يحبه المطورون؟

مفتوح المصدر

مجاني، شفاف، وقابل للتخصيص.

سهولة الأتمتة والسكريبتات

مثالي للتكرار، الأتمتة، وبناء الأدوات.

أدوات قوية لسطر الأوامر

تحكم كامل وسرعة في إنجاز المهام.

بيئة ممتازة لـ Git, Docker, DevOps

تكامل قوي مع أدوات التطوير والنشر الحديثة.

أوامر وأدوات أساسية في Linux

أوامر Linuxالأساسية

الأمر	الاستخدام	مثال
ls	عرض الملفات والمجلدات	ls -la
cd	التنقل بين المجلدات	cd /var/log
pwd	عرض المسار الحالي	pwd
mkdir	إنشاء مجلد	mkdir my-folder
touch	إنشاء ملف فارغ	touch file.txt
cp	نسخ ملف أو مجلد	cp file.txt /home/
mv	نقل أو إعادة تسمية	mv old.txt new.txt
rm	حذف ملف أو مجلد	rm file.txt
cat	عرض محتوى ملف	cat file.txt
chmod	تغيير الصلاحيات	chmod 755 script.sh
ps	عرض العمليات	ps aux
df	عرض مساحة الأقراص	df -h

هيكل نظام الملفات في Linux

/

/bin	/etc	/home	/var	/usr	/tmp
أوامر النظام	إعدادات النظام	مجلدات المستخدمين	ملفات متغيرة	البرامج والمكتبات	ملفات مؤقتة

أنواع الـ Shell

bash | zsh | sh | fish

مدراء الحزم (أمثلة)

```
sudo apt install <pkg>
sudo dnf install <pkg>
sudo pacman -S <pkg>
```

لتشغيل أمر بصلاحيات المدير

sudo <command>

أدوات مفيدة

- grep 🔍 البحث داخل الملفات
- find 🔎 البحث عن الملفات
- tar 📦 ضغط وفك الملفات
- nano/vim 🖍️ محررات النصوص
- top/htop 🖥️ عرض العمليات
- curl/wget 🌐 جلب البيانات

نصائح مهمة 💡

- تجنب العمل كـ root دائماً
- اقرأ رسائل الخطأ، فهي تساعدك
- استخدم Tab للإكمال التلقائي
- استخدم history لمراجعة الأوامر
- تعلم man <command> للمساعدة

أشهر التوزيعات واستخداماتها في Linux

توزيعات **Linux** هي أنظمة جاهزة مبنية على نواة **Linux** تختلف في الأدوات، وسهولة الاستخدام، والجمهور المستهدف.

Ubuntu / Debian

- سهل للمبتدئين
- شائع في الخوادم والتطوير
- apt / deb

```
$ apt install pkg
```

Fedora / RHEL / Rocky

- مناسب للشركات والمؤسسات
- تركيز على الاستقرار والدعم
- dnf / rpm

```
$ dnf install pkg
```

Arch Linux

- مرن وقابل للتخصيص
- مناسب للمستخدمين المتقدمين
- pacman

```
$ pacman install pkg
```

openSUSE

- قوي للإدارة والضبط
- مميزة YaST أداة
- مناسب للمطورين والإداريين

```
$ zypper install pkg
```

Kali Linux

- مخصص للاختبار الأمني
- أدوات أمن جاهزة
- ليس توزيعة يومية

```
$ apt install pkg
```

كيف أختار التوزيعة ؟

للتعلم 🎓	للشركات 🏢	للتخصيص ⚙️	للأمن السيبراني 🔐
Ubuntu	**RHEL / Rocky**	**Arch**	**Kali**

الحوسبة السحابية

هدف الفقرة: فهم أساسيات الحوسبة السحابية وكيفية استخدامها لنشر التطبيقات وإدارة البنية التحتية

1 ما هي الحوسبة السحابية؟

- توفير موارد الحوسبة (خوادم، تخزين، شبكات، قواعد بيانات..) عبر الإنترنت عند الطلب
- تدفع مقابل ما تستخدمه فقط
- مرونة عالية وقابلية للتوسع
- لا تحتاج لإدارة العتاد بنفسك

2 نماذج الخدمة السحابية

SaaS (Software as a Service)	PaaS (Platform as a Service)	IaaS Infrastructure as a Service
		
تطبيقات جاهزة تستخدمها عبر المتصفح دون أي إدارة. مثال Gmail, Office 365	توفر منصة جاهزة لتطوير ونشر التطبيقات دون إدارة البنية التحتية. مثال: App Service.	تستأجر البنية التحتية (خوادم، تخزين، شبكة) وتدير ما فوقها. مثال: EC2, VM

الحوسبة السحابية

عناصر أي بنية سحابية

2 أشهر مزودي الخدمة السحابية

Google Cloud	Amazon Web Services	Microsoft Azure
- قوي في البيانات والذكاء الاصطناعي. - تكامل ممتاز مع خدمات Google.	- الأكبر عالمياً من حيث الخدمات والبنية التحتية. - خيارات وخدمات ضخمة.	- تكامل قوي مع منتجات Microsoft. - قوي في الشركات والمؤسسات.

الحوسبة السحابية

كيف تعمل السحابة؟

1- أنت(تطبيقك)ترسل طلبك عبر الإنترنت

2- مقدم الخدمة السحابية يستقبل الطلب

3- يوفر الموارد المطلوبة ويدير الاتصال

4- تشغيل التطبيق، تخزين البيانات، تأمين الاتصال

✓ لماذا نستخدم السحابة؟

- ✓ توفير التكاليف – لا حاجة لشراء عتاد مكلف
- ✓ قابلية التوسع – زيادة أو تقليل الموارد بسهولة
- ✓ الاعتمادية والتوافر – خدمات موثوقة ومتاحة عالمياً
- ✓ السرعة في النشر – تشغيل تطبيق خلال دقائق
- ✓ التركيز على التطبيق – تترك البنية التحتية للمزود

السحابة تساعدك تبني بسرعة، بأمان، وتتوسع بلا حدود, جميع الخدمات متكاملة وتعمل معاً عبر الإنترنت.

السيرفرات والاستضافة وبيئة النشر (Deployment Environment)

ما هو السيرفر

هو جهاز قوي يعمل 24/7 لتشغيل التطبيقات وتخزين البيانات وتقديمها للمستخدمين عبر الإنترنت.

مواصفات مهمة: (معالج – ذاكرة – تخزين – سرعة اتصال – توافر عالي)

أنواع الاستضافة

Dedicated Server (سيرفر مخصص)	VPS Hosting(سيرفر افتراضي)	Shared Hosting (مشاركة)
•سيرفر كامل لك وحدك •أداء وأمان عالي •الأعلى سعراً	•موارد مخصصة لك •مرونة وتحكم أعلى •سعر متوسط	•مناسب للمواقع الصغيرة •موارد مشتركة مع مواقع أخرى •أرخص خيار

إعدادات مهمة لأي سيرفر

Security -الأمان (تحديثات تجديد .Firewall, SSH)

Domain الدومين
ربط اسم النطاق بالتطبيق. myapp.com

SSL Certificate (شهادة الأمان)
تفعيل HTTPSلحماية البيانات.

Backup (النسخ الاحتياطي)
نسخ بياناتك بانتظام تحسبًا لأي عطل.

أساسيات الأمان في السحابة Cloud Security Basics

الأمن مسؤولية مشتركة: المزود يوفر البنية التحتية الآمنة، وأنت تؤمن تطبيقاتك وبياناتك وصلاحياتك.

1 جدار الحماية Firewall

- يتحكم في حركة المرور الواردة والصادرة
- فتح المنافذ (Ports) اللازمة فقط
- استخدم Security Groups أو Network ACLs
- مثال: السماح بـ 80 و 443 فقط وحظر الباقي

2 الوصول الآمن SSH

- استخدم SSH بدلاً من كلمات المرور
- استخدم مفاتيح SSH Key
- غير المنفذ الافتراضي (22) إن أمكن
- مثال: ssh -i key.pem user@your-server

3 Secrets & Passwords

- لا تضع كلمات السر في الكود
- استخدم إدارة الأسرار (Secrets Manager)
- قم بتدوير (تغيير) الأسرار دورياً
- مثال: AWS Secrets Manager, Azure Key Vault

4 التحكم في الوصول Access Control

- أعطِ كل مستخدم أقل صلاحية ممكنة (Least Privilege)
- استخدم IAM Users و Groups
- لا تستخدم الحساب الجذري للمزود في العمل اليومي
- مثال: مطور = وصول إلى S3 فقط

5 التحديثات Update & Patching

- حدّث نظام التشغيل والتطبيقات بانتظام
- استخدم تصحيحات الأمان (Patches)
- فعّل التحديثات التلقائية إن أمكن
- مثال: تحديث Ubuntu أو Windows وقواعد البيانات

6 إدارة الصلاحيات بدقة

- راجع الصلاحيات بشكل دوري
- احذف المستخدمين غير النشطين
- فعّل المصادقة متعددة العوامل (MFA)
- مثال MFA: بمراجعة الصلاحيات كل شهر (Access Review)

أدوات الأمان في السحابة

AWS IAM	AWS Security Groups
AWS Secrets Manager	AWS GuardDuty
Azure AD	Azure Key Vault

المصادقة متعددة العوامل (MFA)

كلمة المرور + هاتفك + رمز = تحقق دخول آمن

طبقة أمان إضافية تمنع الوصول حتى لو تسرّبت كلمة المرور

نموذج المسؤولية المشتركة

مسؤوليتك : البيانات والتطبيقات، إدارة الهوية والصلاحيات، تأمين نظام التشغيل (OS)، تهيئة الشبكة (Security Groups)، نسخ احتياطية للتطبيق والبيانات

مسؤولية المزود: (Cloud Provider) مراكز البيانات والبنية التحتية، الأجهزة والشبكات الأساسية، الطاقة والتبريد والاتصال الفيزيائي، أمن الخدمة والخدمات الأساسية

بعد النشر... ماذا يحدث؟ Backup, Monitoring, Logs

النشر ليس نهاية العمل، بل بداية التشغيل والاهتمام المستمر لضمان استقرار التطبيق وسرعة حل المشاكل.

النسخ الاحتياطي (Backup)

النسخ الاحتياطي يحمي بياناتك من الحذف، الأعطال، أو الهجمات

استراتيجيات النسخ الاحتياطي

- ❑ كامل (Full Backup)نسخ كل البيانات مرة واحدة
- ❑ تزايدي (Incremental)نسخ التغيرات فقط بعد آخر نسخة
- ❑ تفاضلي (Differential)نسخ التغيرات منذ آخر نسخة كاملة.

➢ ما يتم نسخه عادة قواعد البيانات الملفات والمستندات إعدادات التطبيق صور وملفات مرفوعة

المراقبة (Monitoring)

تنظيم البيانات في صفوف وأعمدة كل جدول يمثل كياناً واحداً

ماذا نراقب؟

- ❑ توفر الخدمة ((Uptime
- ❑ حركة الشبكة
- ❑ استخدام التخزين
- ❑ استهلاك الذاكرة
- ❑ استهلاك المعالج

أدوات مراقبة شائعة في السحابة

- ❑ AWS CloudWatch
- ❑ Azure Monitor
- ❑ Google Cloud Monitoring
- ❑ Prometheus + Grafana

سجلات النظام (Logs)

PK: يحدد كل سجل بشكل فريد
FK: يربط بين الجداول

أدوات إدارة السجلات

- ❑ AWS CloudWatch Logs
- ❑ Azure Log Analytics
- ❑ Google Cloud Logging
- ❑ ELK Stack (Elasticsearch, Logstash, Kibana)

أدوات المطور للتعامل مع السحابة والنشر

VS Code

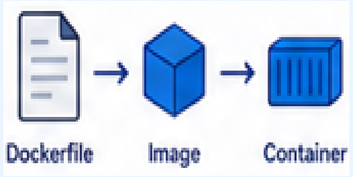
محرر الكود الذكي

مثال أمر:

```
Extensions + Terminal
```

Docker

حاويات التطبيقات



مثال أمر:

```
docker build -t myapp .
```

Git

نظام التحكم بالإصدارات

مثال أمر:

```
git push origin main
```

Azure CLI

التحكم بـ Microsoft Azure

مثال أمر:

```
az group list
```

AWS CLI

إدارة موارد AWS من سطر الأوامر

مثال أمر:

```
aws s3 ls
```

رحلة الكود من جهازك إلى السحابة:

VS Code
كتابة الكود

→

Git
حفظ ورفع

→

Docker
بناء Image

→

AWS/Azure
نشر على السحابة

كيف يصبح النشر آليًا داخل الشركات؟ CI/CD Pipelines

رحلة التعديل من الكود إلى المستخدم

💡 الفكرة الأساسية:

كلما حدث تغيير في الكود، يقوم النظام بكل شيء تلقائياً حتى يصل التطبيق للمستخدم.

1 **تعديل الكود**

VS Code

2 **رفع الكود**

GitHub/GitLab

3 **Pipeline (CI)**

جلب + فحص + اختبار

4 **بناء Image**

Docker Image

5 **النشر (CD)**

دفع للسحابة

6 **وصول المستخدم**

أحدث نسخة

CI (Continuous Integration) — التكامل المستمر

جلب الكود ← تثبيت المتطلبات ← فحص الكود (Lint) ← تشغيل الاختبارات ← ✅ نجاح

هدف CI: التأكد من أن الكود يعمل بشكل صحيح مع باقي الكود في المشروع.

CD (Continuous Deployment) — النشر المستمر

بناء Docker Image ← رفع Image إلى Registry ← نشر التطبيق على السحابة ← تحديث تلقائي ← ✅ وصل للمستخدم

هدف CD: نشر التطبيق تلقائياً بعد نجاح جميع مراحل الـ CI.

البرمجة الحديثة دورة حياة كاملة وليست مجرد كتابة كود.

ما بعد كتابة الكود
لماذا لا ينتهي العمل عند البرمجة؟

Installation & Infrastructure
إعداد السيرفر، قاعدة البيانات، الدومين، و **SSL**

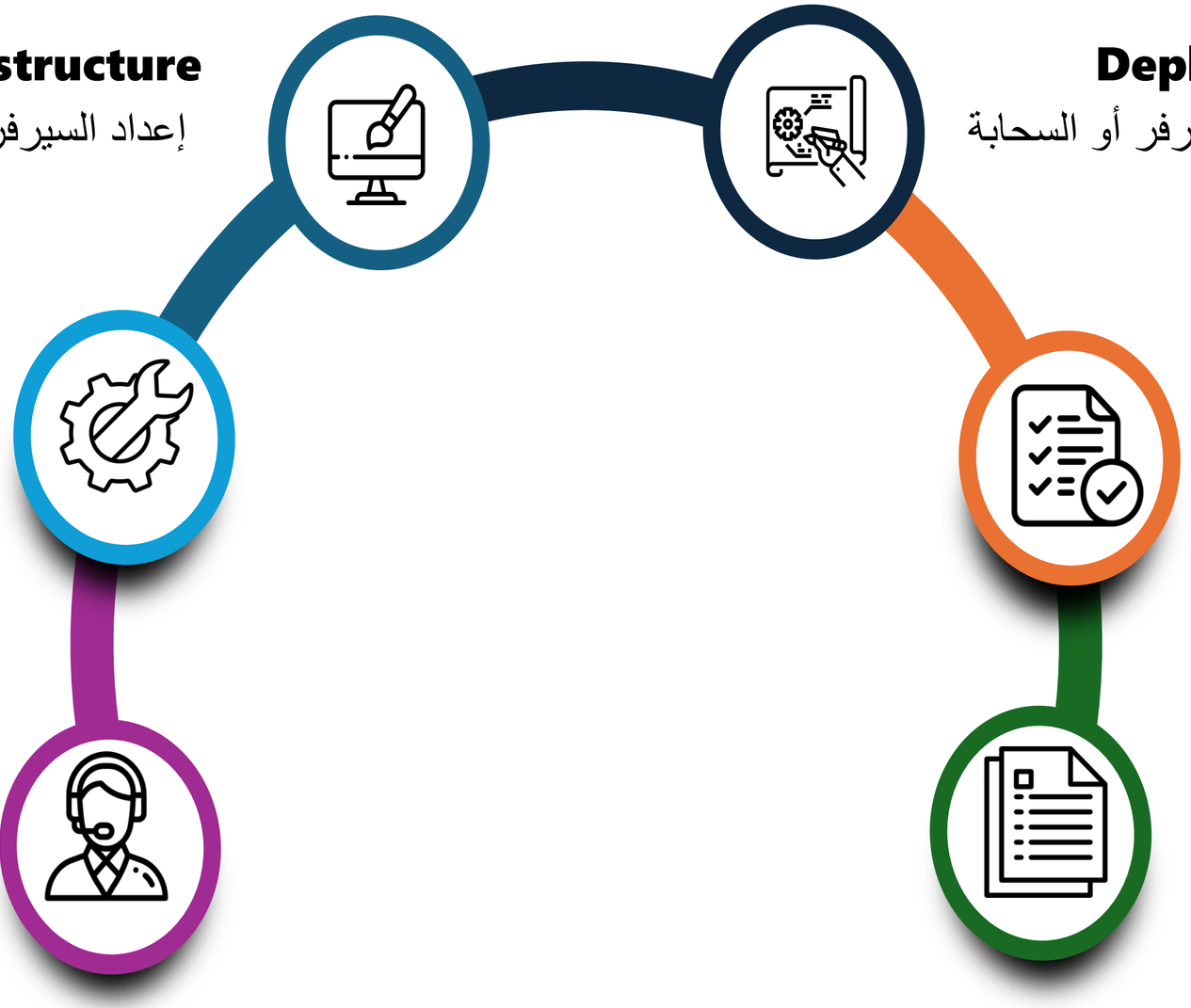
Deployment
نشر التطبيق على السيرفر أو السحابة

Maintenance
إصلاح الأخطاء، التحديثات، وتحسين الأداء

Testing
التحقق من صحة الكود قبل وصوله للمستخدم

Support
مساعدة المستخدمين والعملاء بعد إطلاق النظام

Documentation
توثيق طريقة تشغيل النظام، الإعدادات، وشرح المكونات



Documentation & Testing

💡 Documentationيسهل الفهم، و Testingيقلل الأخطاء.

Testing

- **Unit Testing**
  - اختبار جزء صغير من الكود.
- **Integration Testing**
  - اختبار تكامل الأجزاء معًا.
- **Manual Testing**
  - اختبار يدوي من قبل المطور أو المختبر.
- **Automated Testing**
  - اختبارات آلية تتكرر مع كل تحديث.

Documentation

- **README**
  شرح سريع للمشروع وطريقة التشغيل.
- **API Documentation**
  - توضيح نقاط الـ APIوطريقة استخدامها.
- **Technical Notes**
  - ملاحظات للمطورين حول البنية والإعدادات.
- **User Guide**
  - دليل مختصر للمستخدم أو العميل.

النشر الناجح يحتاج إعداداً صحيحاً للبنية التحتية.

Deployment, Installation & Infrastructure

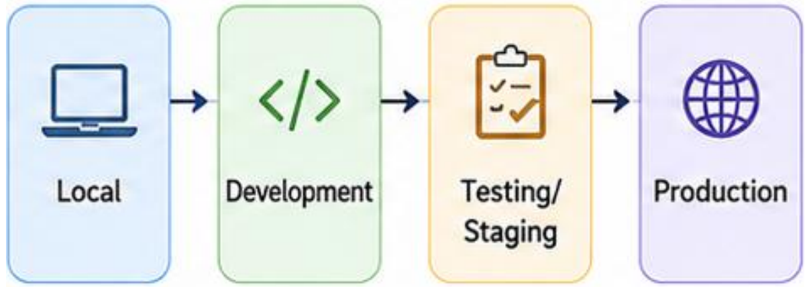
Deployment

- نقل التطبيق من جهاز المطور إلى بيئة التشغيل.
- قد يكون على Serverأو.Cloud
-يحتاج إعدادات ونسخة مستقرة.

Installation & Infrastructure

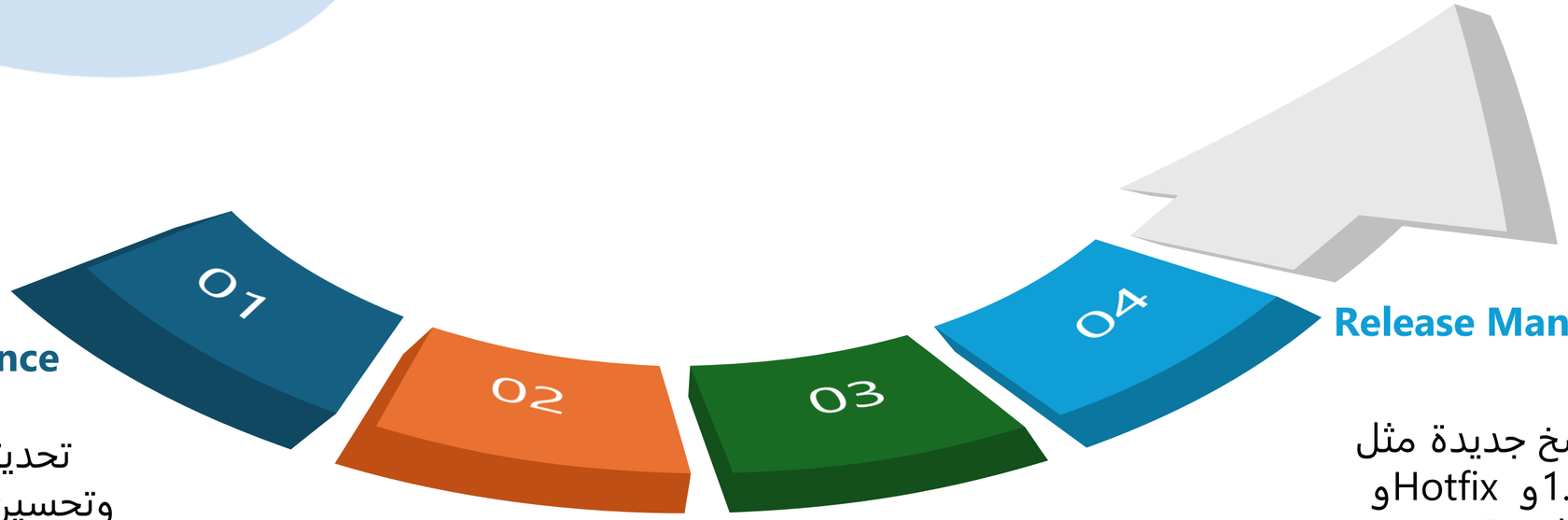
- إعداد قاعدة البيانات.
- إعداد Domainو .SSL
-إعداد صلاحيات وملفات التهيئة.
- التأكد من عمل الخدمات المرتبطة.

بيئات العمل



لا نجرب مباشرة على بيئة المستخدمين.

Maintenance, Bug Fixing, Support & Releases



Maintenance

تحديثات مستمرة وتحسين الأداء ومتابعة الاستقرار.

Bug Fixing

اكتشاف الخطأ، تحليله، إصلاحه، ثم إعادة اختباره.

Support

التعامل مع مشاكل المستخدمين وشرح استخدام النظام.

Release Management

إصدار نسخ جديدة مثل 1.0 و 1.1 و Hotfix و Patch.

بعد الإطلاق تبدأ مرحلة طويلة من التحسين والمتابعة.

لماذا نحتاج إدارة مشاريع برمجية

البرمجة ليست فقط كتابة كود، بل إدارة عمل كامل من البداية للنهاية.

لماذا هي مهمة؟

- المشاريع البرمجية كبيرة ومعقدة
- الفريق يضم أدوارًا مختلفة
- المتطلبات قد تتغير باستمرار
- الوقت والميزانية محدودة
- نحتاج جودة عالية وتسليم الوقت المناسب

أهداف إدارة المشاريع

- 🗓️ الالتزام بالوقت والميزانية
- 🎯 تحقيق أهداف المشروع
- 👥 تنظيم العمل والفريق
- ✅ ضمان الجودة

💡 إدارة جيدة = مشروع ناجح +فريق سعيد +زبون راضٍ

نموذج Waterfall (النموذج التقليدي)

1 | تحليل المتطلبات Requirements
نفهم ماذا يريد الزبون ونجمع كل المتطلبات.

2 | التصميم Design
نصمم النظام (المعمارية، قاعدة البيانات، الواجهات..)

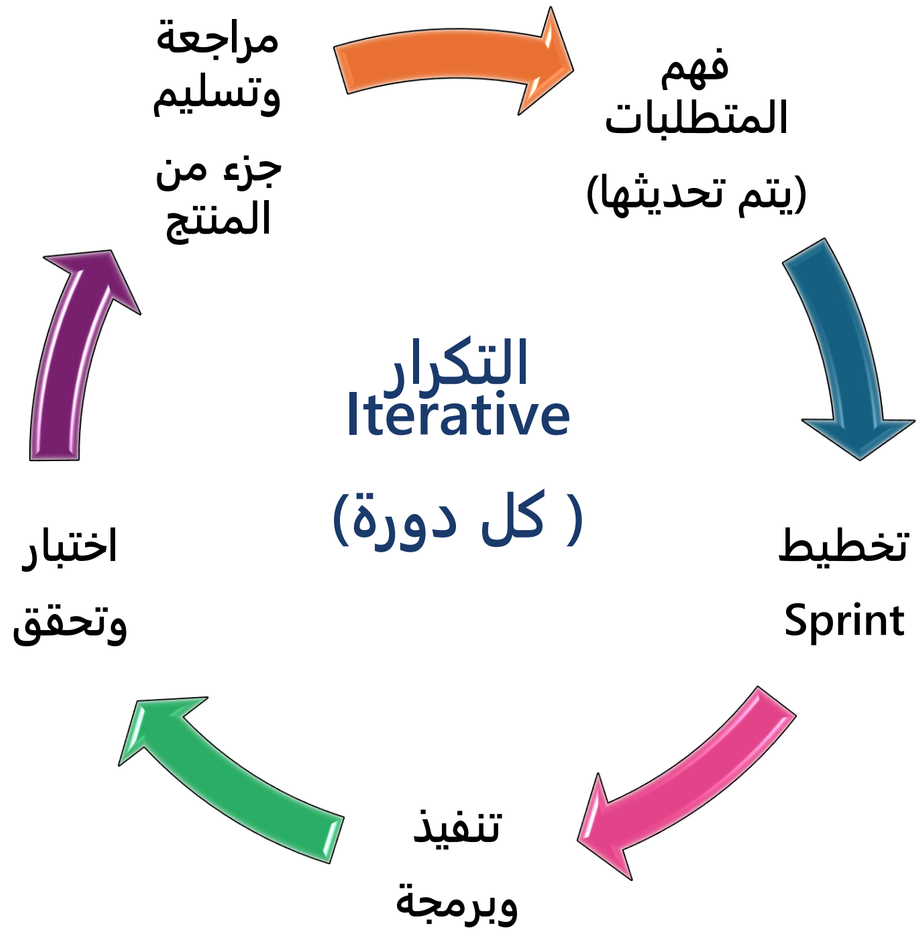
3 | البرمجة Implementation
نكتب الكود بناءً على التصميم.

4 | الاختبار Testing
نختبر النظام للتأكد من أنه يعمل بشكل صحيح.

5 | النشر Deployment
نسلم النظام للمستخدمين وننشره في بيئة الإنتاج.

مناسب عندما تكون المتطلبات واضحة وثابتة ولا تتغير كثيرًا.

نموذجAgile (العمل المرن والتكراري)



✅ مميزاته

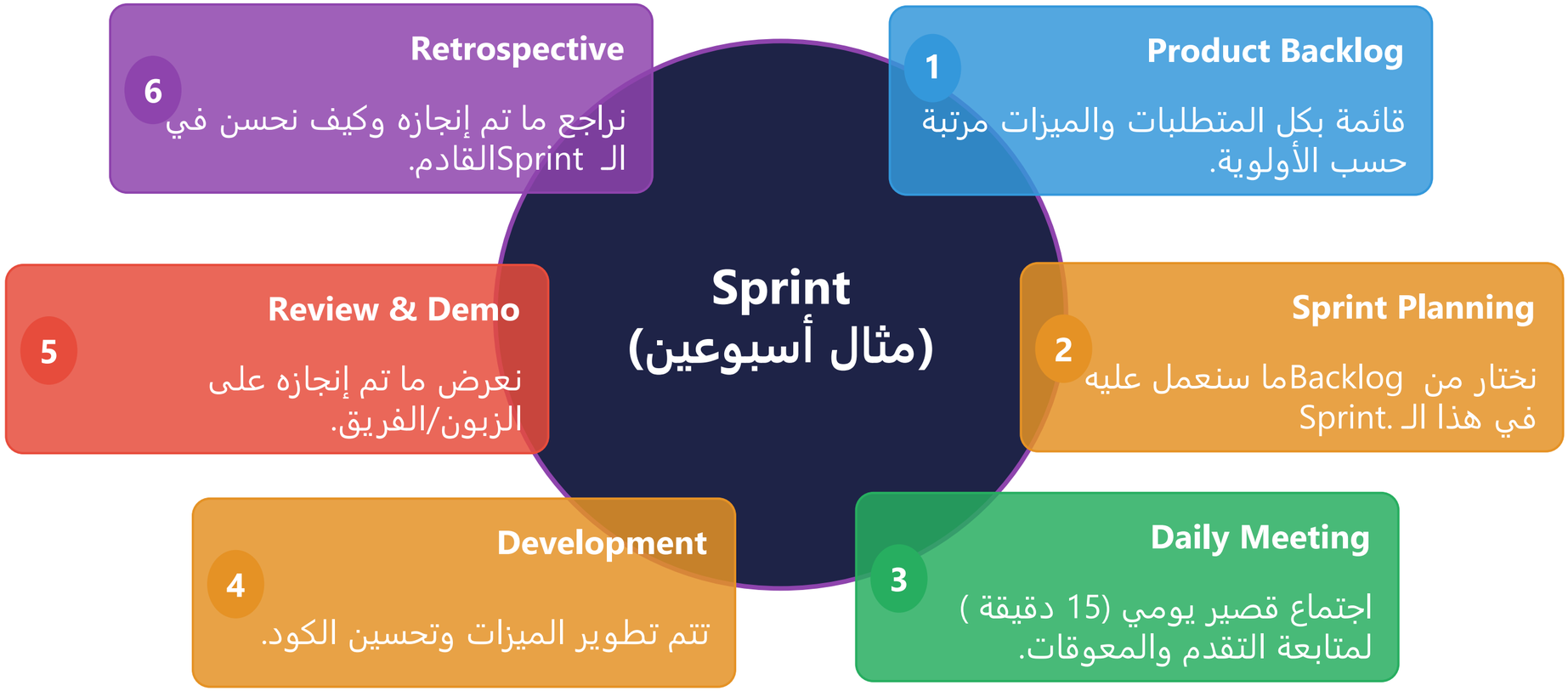
- مرن ويتعامل مع التغييرات بسهولة
- نسلم أجزاء قابلة للاستخدام بشكل متكرر
- تواصل مستمر مع الزبون
- اكتشاف المشاكل مبكرًا

عيوبه X

- يتطلب تواصلاً دائمًا وانضباطًا من الفريق
- قد يكون صعبًا في المشاريع الكبيرة جدًا بدون تنظيم جيد

Scrumعملي Sprint / Backlog / Daily / Review

Scrumهو إطار عمل **Agile**لتطبيق **Agile**بطريقة منظمة وفعالة.



من فكرة الزبون إلى Task ثم مرحلة ثم Release

1

فكرة الزبون

الزبون يطلب ميزة أو يبلغ عن مشكلة

2

User Story

تحويل الفكرة إلى Storyلاستخدام مخصص

3

Tasks

تقسيم الـ Story إلى مهام صغيرة

4

Development

المطور يكتب الكود ويقوم بـ Code Review

5

Testing & Review

المختبر يختبرها ويتم المراجعة

6

Release

إدراج ونشر في بيئة الإنتاج

الأدوات المستخدمة في كل مرحلة

Monitoring

مراقبة التطبيق بعد النشر

Testing Tools

اختبار تلقائي ويدوي

CI/CD Pipeline

بناء وإنتاج تلقائي وأبودي

Git / GitHub / GitLab

إدارة الكود وتاريخ الفروع

Jira / Azure DevOps

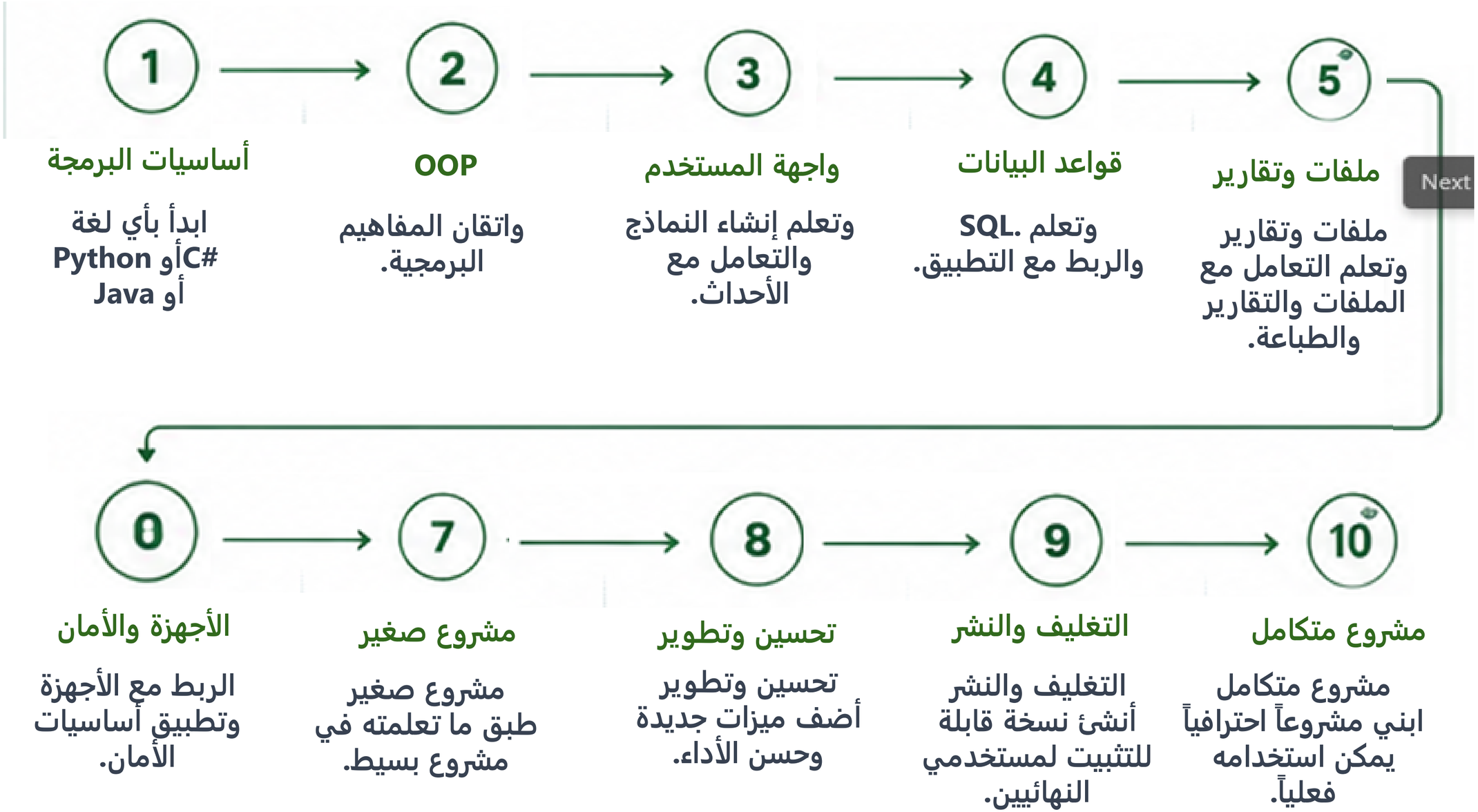
إدارة التذاكر أو أبودي

Agile vs Waterfall

العنصر	Agile	Waterfall
النهج	تكراري ومرن	خطي ومتسلسل
التعامل مع المتطلبات	تتغير باستمرار وبسهولة	ثابتة من البداية، تغييرها صعب
تسليم المنتج	تسليم متكرر في كل دورة (Sprint)	مرة واحدة في نهاية المشروع
رؤية المنتج	ترى أجزاء من المنتج بشكل مستمر	ترى المنتج كاملًا في النهاية فقط
المخاطر	اكتشاف الأخطاء مبكرًا	اكتشاف الأخطاء متأخرًا
التواصل مع الزبون	مستمر طوال المشروع	قليل بعد جمع المتطلبات

لا يوجد نموذج أفضل دائمًا! الاختيار يعتمد على: طبيعة المشروع، استقرار المتطلبات، حجم الفريق، والوقت والميزانية.

خريطة التعلم المقترحة (ترتيب منطقي)



كيف تبدأ عملياً؟

اختر مجالاً قريباً من ميولك

اختر مجالاً تستمتع به وتراه مناسباً
لقدراتك واهتماماتك.

تعلّم أساسيات البرمجة.

ابدأ بلغة واحدة مناسبة للمجال
الذي اخترته وأتقن أساسياتها .

تعلّم SQL و Git

SQLللتعامل مع البيانات، و **Git**لإدارة
الإصدارات والتعاون .

أنشئ مشروعاً صغيراً.

طبق ما تعلمته في مشروع بسيط
يعكس مهاراتك.

ارفعه على و اشرحه بكلام واضح. GitHub

أنشئ حساباً وارفغ مشروعك ليكون قابلاً للعرض و
اكتب ملف **README**يوضح الفكرة، التقنيات، وكيف
يعمل المشروع .

اكتب CV تقني بسيط

ركّز على المهارات، المشاريع،
التقنيات، وروابط أعمالك

ابحث عن تدريب أو فرصة Junior

قدّم على فرص تدريب أو وظائف
مبتدئة لاكتساب الخبرة.

استمر في التعلم والتحسين.

التعلم المستمر هو سر التطور
والنجاح في هذا المجال.

لماذا الرياضيات مهمة في البرمجة؟

الرياضيات تعطي المبرمج طريقة تفكير قوية، وليست مجرد مادة نظرية
البرمجة ليست مجرد كتابة أوامر، بل تحليل للمشكلات، بناء نماذج، واتخاذ قرارات منطقية

01
التفكير المنطقي
يساعد على فهم الشروط والقرارات داخل البرامج

02
تحليل المشكلات
يحول المشكلة الكبيرة إلى خطوات صغيرة قابلة للحل

03
التجريد والنمذجة
يمكننا من تمثيل الواقع داخل نظام برمجي

04
الخوارزميات
يبني خطوات مرتبة وواضحة للوصول إلى الحل

05
البيانات والإحصاء
يفيد في التحليل، التوقع، وفهم النتائج

طالب الرياضيات لا يبدأ من الصفر في البرمجة، بل يملك أساساً قوياً للتعلم السريع والفهم العميق

المنطق و الشروط واتخاذ القرار

المنطق الرياضي يظهر يومياً في if / else وعمليات AND وOR وNOT
البرنامج يتخذ قرارات بناءً على شروط منطقية، تماماً كما في المنطق الرياضي

Boolean

القيمتان الأساسيتان True وFalse

AND / OR / NOT

ربط الشروط أو نفيها لاتخاذ القرار الصحيح

if / else

بنية برمجية لاختيار مسار التنفيذ

التحقق

مثل فحص العمر، كلمة المرور، أو الصلاحيات

مثال


```
if  age >= 18
&& isActive
```

كل نظام برمجي يحتوي آلاف القرارات المنطقية الصغيرة

الدوال الرياضية والدوال البرمجية

فكرة الدالة في الرياضيات تمتد طبيعياً إلى البرمجة
كل دالة تستقبل مدخلات، تعالجها، ثم تعطي مخرجات

01

في الرياضيات

$fx = x^2 + 2x + 1$

مثال على دالة رياضية

02

في البرمجة

تستقبل **Function**

وترجع **Parameters**

Result

03

Input

مثل السعر، الكمية،

أو اسم المستخدم

04

Output

مثل المجموع،

النتيجة،

أو رسالة التحقق

05

مثال

CalculateTotal

price, quantity

الدالة البرمجية هي امتداد عملي لفكرة
input → process → output

المصفوفات والجبر الخطي والبيانات

أين تظهر؟

- معالجة الصور
- الذكاء الاصطناعي
- Machine Learning
- الرسوميات
- Data Science

ما هي المصفوفة؟

ترتيب منظم للقيم في صفوف وأعمدة

1	2	3
4	5	6
7	8	9

مثال تطبيقي

يمكن تمثيل صورة رمادية بدرجات البكسل كمصفوفة من القيم

255	180	90	30
190	120	60	20
150	90	40	10
80	40	15	5

في الذكاء الاصطناعي نتعامل مع

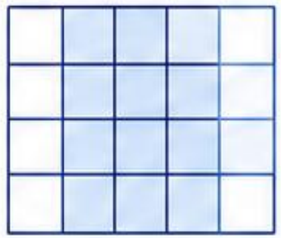
متجهات Vectors

قائمة مرتبة من الأعداد تمثل بيانات مثل الميزات أو التنبوءات

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

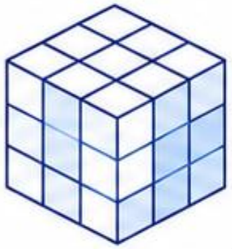
مصفوفات Matrices

جدول من القيم في صفوف وأعمدة تستخدم في التحويلات والنمذجة



توترات Tensors

مصفوفات متعددة الأبعاد أكثر من بعدين تستخدم لتمثيل بيانات معقدة مثل الصور والفيديو



الجبر الخطي لغة أساسية في الذكاء الاصطناعي وتحليل الصور والبيانات

الخوارزميات وحل المشكلات

الخوارزمية هي مجموعة خطوات مرتبة لحل مشكلة محددة

05
الهدف
الوصول إلى حل صحيح وواضح وفعال

04
الأمثلة
البحث، الترتيب، والتحقق من البيانات

03
الخطوات
ترتيب الحل بشكل منطقي ومتسلسل

02
التقسيم
تقسيم المشكلة الكبيرة إلى أجزاء أصغر

01
التحليل
فهم المشكلة وتحديد المطلوب بدقة

مثال: البحث عن طالب في قائمة

1. ابدا من اول طالب
2. قارن الاسم
3. اذا وجدته، ارجع النتيجة
4. اذا لم تجده، انتقل للطالب التالي
5. اذا انتهت القائمة، ارجع غير موجود

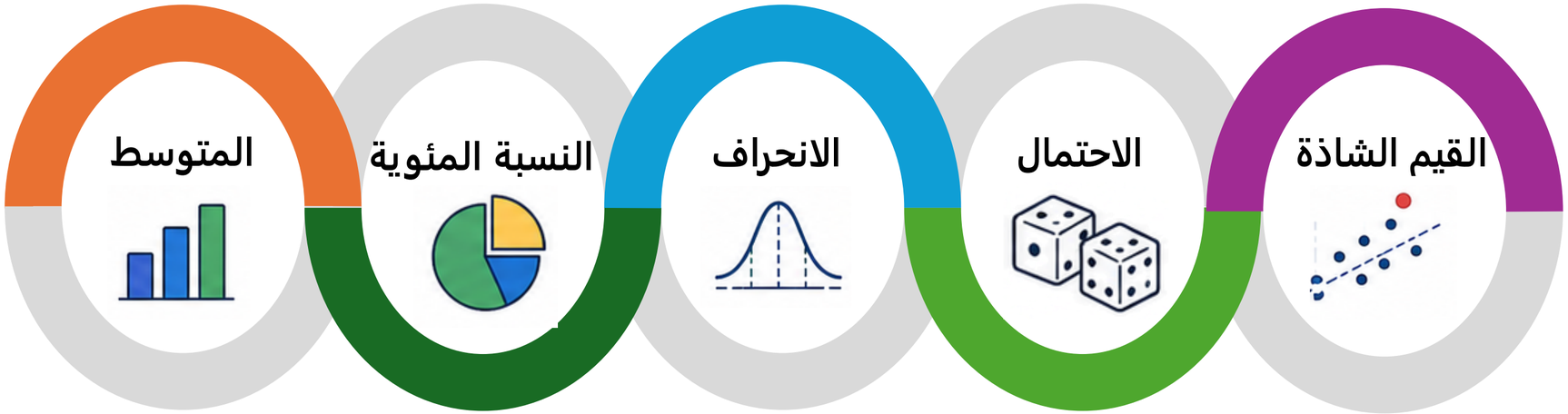
المبرمج القوي لا يحفظ الكود فقط،
بل يحلل المشكلة ويحوّلها إلى خطوات قابلة للتنفيذ

الإحصاء والاحتمالات وتحليل البيانات

أين تظهر؟

- تحليل البيانات
- تقارير الأعمال
- التنبؤ والتوقع
- تحليل المخاطر
- تعلم الآلة

مفاهيم مهمة



أمثلة عملية

- نسبة النجاح
- أكثر منتج مبيعًا
- توقع عدد المستخدمين
- اكتشاف سلوك غير طبيعي

مثال بسيط

$$\text{متوسط الدرجات} = \frac{\text{مجموع الدرجات}}{\text{عدد الطلاب}}$$

```
var average = grades.Average();
```

كل قرار مبني على البيانات يحتاج فهمًا إحصائيًا

العلاقات و قواعد البيانات

قواعد البيانات العلاقية تطبيق عملي مباشر لفكرة المجموعات والعلاقات.
في الأنظمة البرمجية نمثل الكيانات كجداول، ونربطها بعلاقات منطقية واضحة

أين تظهر في البرمجة؟

- قواعد البيانات العلاقية
- الجداول
- العلاقات بين الكيانات
- Primary Key / Foreign Key
- One-to-Many , Many-to-Many

مثال رياضي

Students = {Ali, Sara, Omar}

Classes = {Math, Physics, Programming}

:العلاقة

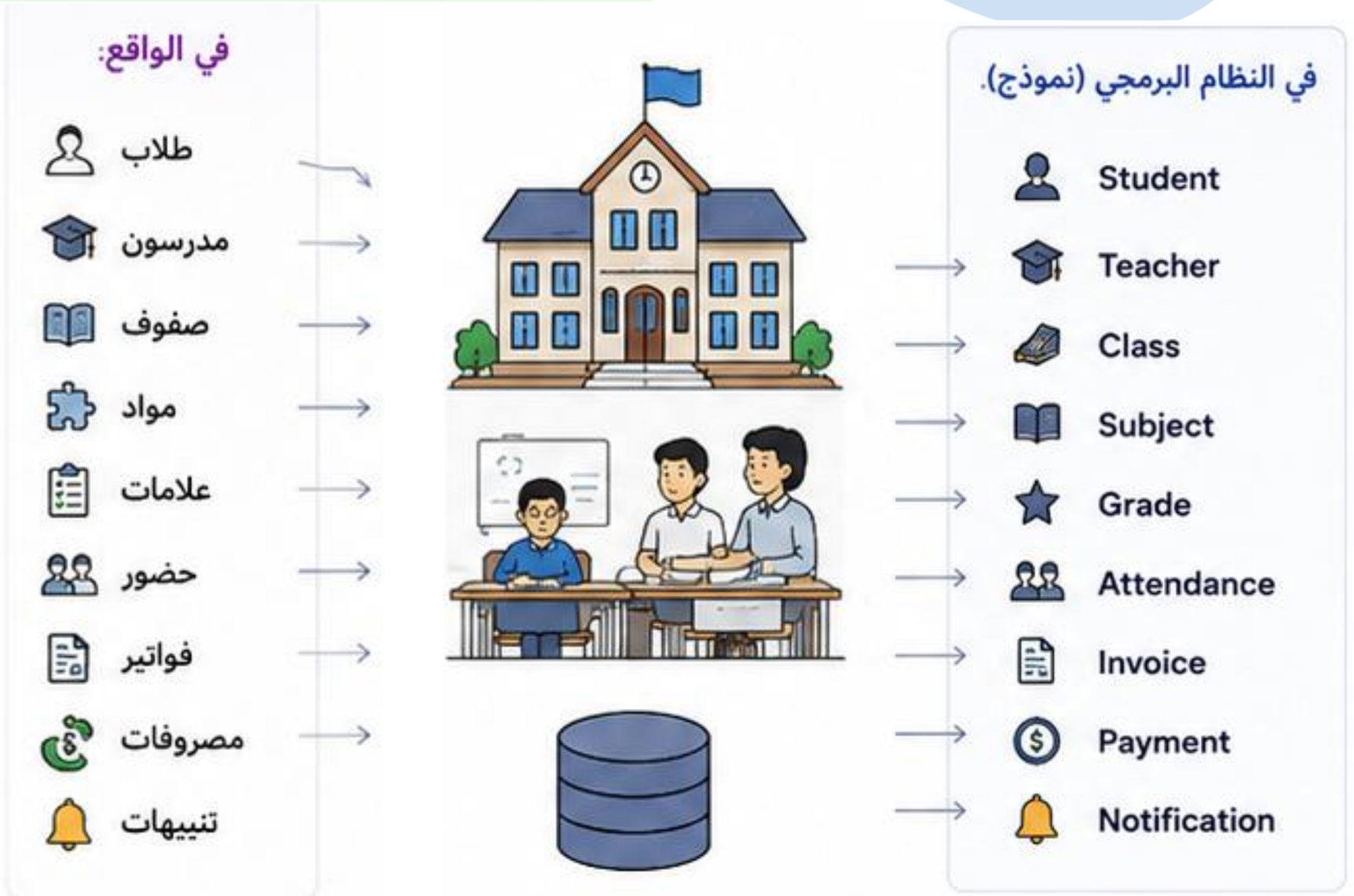
Student enrolled in Class

تمثيلها في قاعدة البيانات

التفكير المجرد وتصميم الأنظمة

ما هو التجريد؟
هو أخذ الأشياء المعقدة في الواقع، وتجميعها إلى نماذج بسيطة أو مجردة، قابلة للنظام للتعامل معها

تصميم النظام البرمجي هو عملية نمذجة للواقع.
كلما كان النموذج جيدًا، كان النظام أكثر دقة وسهولة في التطوير والصيانة



التفكير المجرد وتصميم الأنظمة

الفكرة الأساسية
ليس المهم أن نجد حلاً فقط، بل أن نجد أفضل حل ممكن

أين تظهر؟

- قواعد البيانات العلاقية
- الجداول
- العلاقات بين الكيانات
- Primary Key / Foreign Key
- One-to-Many , Many-to-Many

شركة توصيل
إيجاد أقصر الطرق لتوصيل أكبر عدد من الطلبات

توزيع الطلاب على القاعات
توزيع الطلاب على القاعات لتقليل التعارض وضمان الراحة

تقليل استهلاك السيرفر
تحسين استخدام الموارد لتقليل الطاقة والتكلفة

اختيار أقل تكلفة شحن
اختيار شركة الشحن التي تقدم أفضل تكلفة ضمن الوقت المطلوب

التشفير والأمن السيبراني والرياضيات

أين تظهر؟

- ✓ تشفير البيانات
- ✓ HTTPS
- ✓ كلمات المرور
- ✓ Digital Signatures
- ✓ JWT Tokens
- ✓ Blockchain
- ✓ Cybersecurity

أمثلة رياضية مستخدمة

- ✓ الأعداد الأولية
- ✓ الحسابات المعبارية Modular Arithmetic
- ✓ الدول أحادية الاتجاه
- ✓ الاحتمالات
- ✓ الجبر

عندما تدخل إلى موقع يبدأ بـ HTTPS، فهنالك تشفير يحمي الاتصال بين المتصفح والسيرفر

الأمن السيبراني من أكثر المجالات التي يظهر فيها العمق الرياضي بوضوح

أمثلة مباشرة من سوق العمل

المجال	المفاهيم الرياضية المستخدمة
Web Development	منطق، دوال، مصفوفات، قواعد بيانات
Backend Development	خوارزميات، علاقات، نمذجة، هياكل بيانات
Database Systems	مجموعات، علاقات، منطق، تطبيع الجداول
AI / Machine Learning	جبر خطي، إحصاء، احتمالات، أمثلة
Cybersecurity	نظرية الأعداد، تشفير، دوال دلتا
DevOps / Automation	منطق، خوارزميات، تحليل الأخطاء
Data Analysis	إحصاء، احتمالات، رسوم بيانية، توزيعات
Game Development	هندسة، منتجات، فورمات، مصفوفات
Computer Graphics	مصفوفات، تحولات هندسية، منتجات
ERP / Business Systems	نمذجة، قواعد بيانات، عمليات تجارية

طالب الرياضيات وسوق العمل التقني

طالب الرياضيات لا يبدأ من الصفر في سوق التقنية، لكنه يحتاج إلى تحويل التفكير الرياضي إلى مهارات عملية مطلوبة في الشركات.

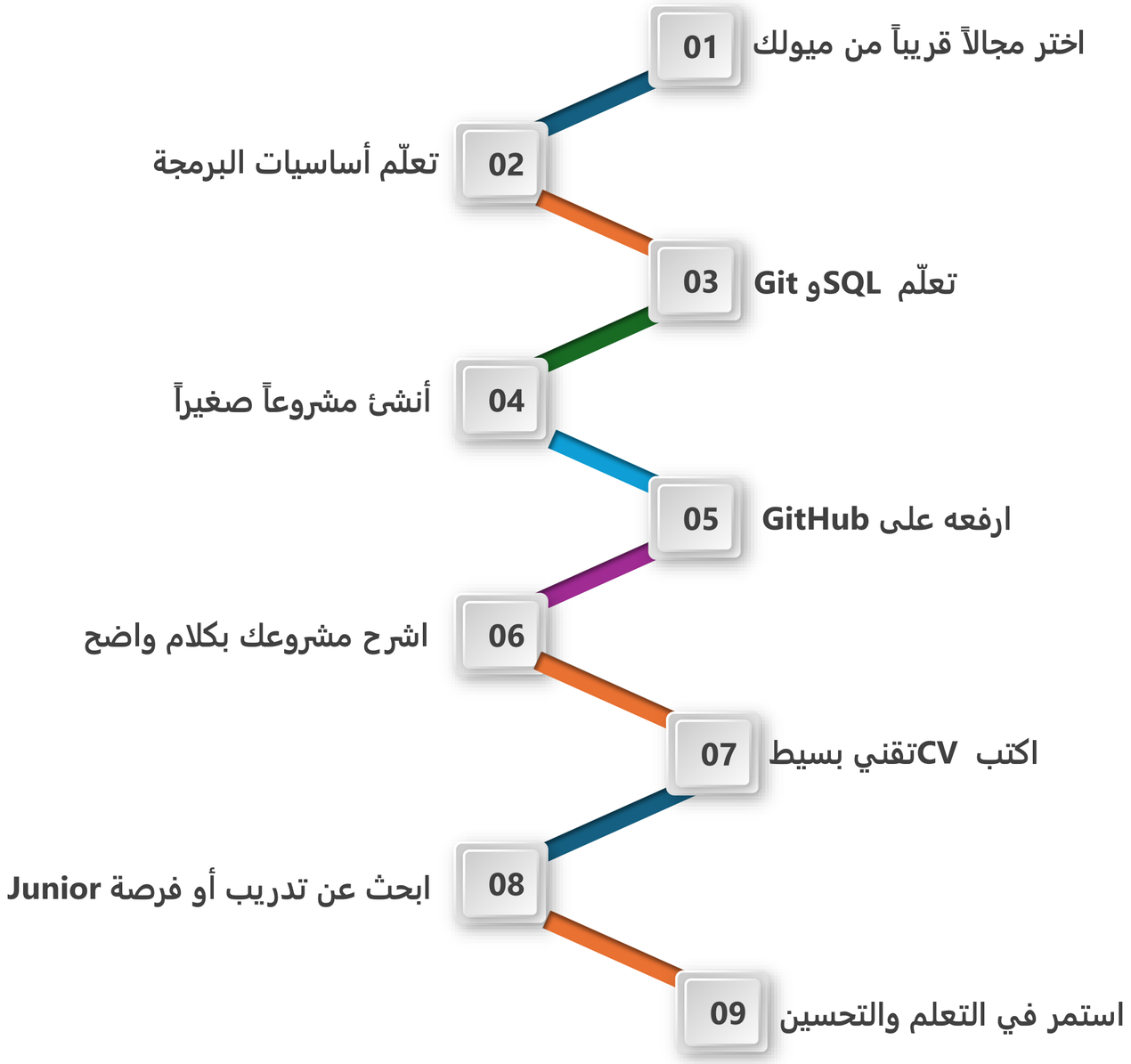
لماذا طالب الرياضيات مناسب لسوق التقنية؟

- قدرة قوية على التحليل
- يفهم التجريد والنمذجة
- يستطيع التعامل مع المشكلات المعقدة
- قابلية جيدة لفهم الخوارزميات والبيانات
- الانتقال إلى مجالات متعددة: البرمجة، الذكاء الاصطناعي..

ماذا يجب أن يضيف الطالب إلى جانب دراسته للرياضيات؟

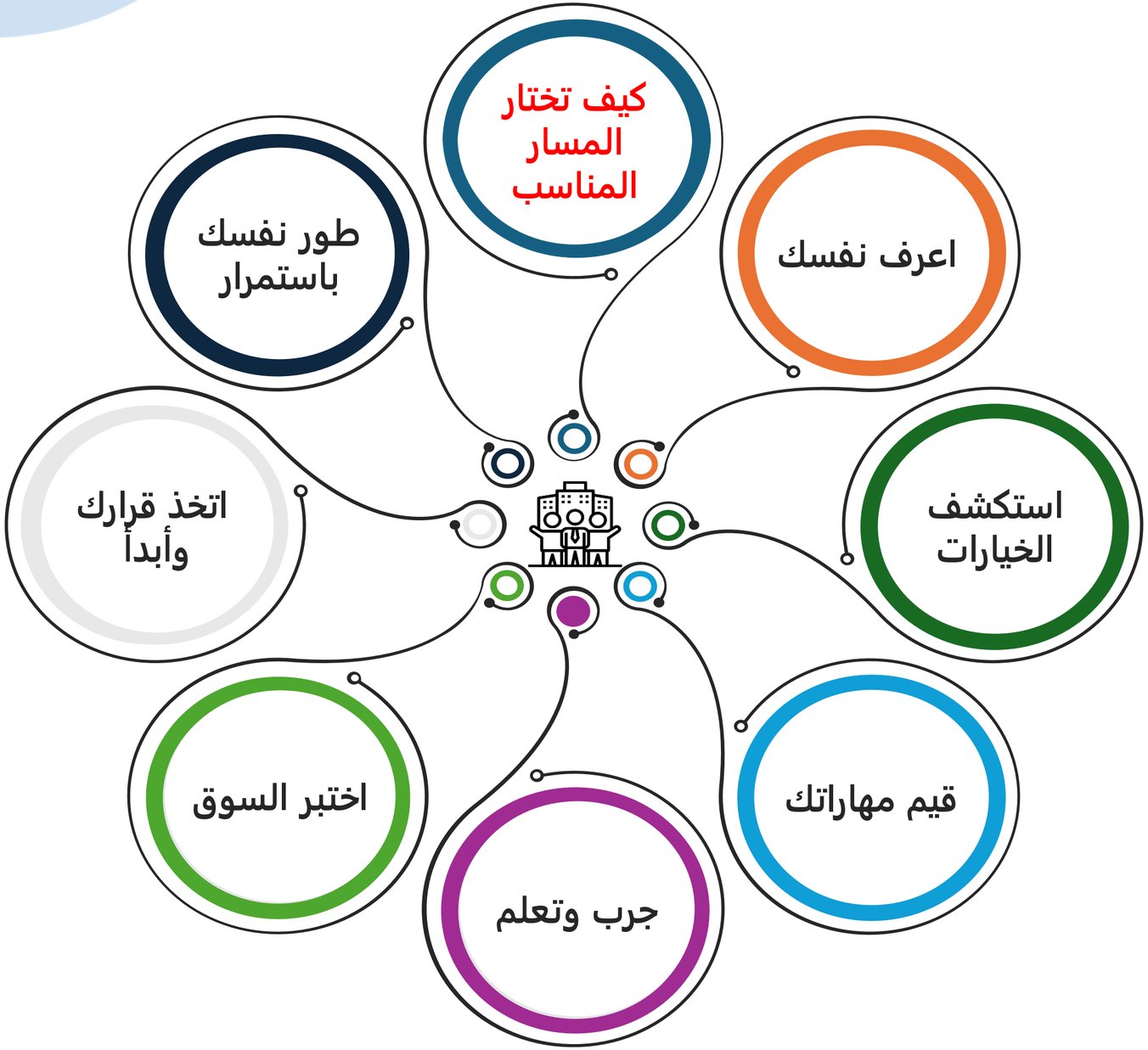
الرياضيات ليست بعيدة عن سوق العمل التقني، بل هي الأساس الفكري الذي يمنحك فهماً أعمق وحلولاً أفضل.

كيف تبدأ عملياً
خارطة دخول بسيطة إلى سوق العمل



لا تبدأ بمعرفة كل شيء، بل ابدأ بخطوة واضحة ومشروع صغير قابل للعرض.

خطوات عملية لاختيار مسارك الوظيفي بثقة ووضوح



التسويق الرقمي / تحسين محركات البحث

لماذا قد يناسب طالب الرياضيات؟

- ✓ التفكير التحليلي وفهم البيانات.
- ✓ القدرة على قراءة الأرقام والمؤشرات.
- ✓ تنظيم الأفكار وبناء استنتاجات واضحة.
- ✓ ربط النتائج بالقرارات العملية.

ماذا يعمل؟

- ✓ التسويق لمنتجات وخدمات رقمية .
- ✓ تحسين ظهور المواقع في Google
- ✓ تحليل سلوك المستخدمين والزوار
- ✓ شرح الحلول التقنية للعملاء ودعم عمليات البيع .
- ✓ ربط الجانب التقني باحتياجات السوق والعملاء.

ماذا يجب أن يدرس؟

- ✓ Digital Marketing.
- ✓ SEO Basics.
- ✓ Content Strategy.
- ✓ Google Analytics / Search Console.
- ✓ أساسيات Web و Hosting و SaaS.
- ✓ مهارات العرض والتواصل.
- ✓ فهم المنتج التقني وقيمة الحمة للعميل.

أدوات مهمة

- ✓ Google Analytics
- ✓ Google Search Console
- ✓ SEMrush / Ahrefs
- ✓ CRM
- ✓ Canva / PowerPoint
- ✓ Email Marketing Tools

مشاريع تدريبية مناسبة

- ✓ تحليل موقع بسيط واقتراح تحسينات SEO.
- ✓ إعداد خطة تسويق لتطبيق تعليمي.
- ✓ تصميم صفحة تعريفية لمنتج SaaS.
- ✓ إعداد عرض تقديمي لبيع نظام مدرسي.

الدعم الفني / نجاح العملاء

01

لماذا يناسب طالب الرياضيات؟

✓ لأنه يعتمد على التحليل المنطقي وحل المشكلات.

✓ لأنه يتطلب فهماً للأنظمة والعمليات خطوة بخطوة.

✓ لأنه يحتاج لعمل دقيق وصبور ومنظم.

✓ لأنه يجمع بين المهارات التقنية والتواصل مع الناس.

02

ماذا يعمل؟

✓ الرد على استفسارات العملاء وحل المشكلات.

✓ تشخيص الأعطال وتقديم الحلول المناسبة.

✓ مساعدة العملاء على استخدام الأنظمة والمنتجات.

✓ متابعة حالة التذاكر والتأكد من الحل النهائي.

✓ بناء رضا العملاء وتحسين تجربة الاستخدام.

03

ماذا يجب أن يدرس؟

✓ أساسيات الشبكات ونظم التشغيل

✓ مبادئ قواعد البيانات SQL

✓مهارات الاتصال والتواصل

✓ إدارة علاقات العملاء CRM

✓إدارة المشكلات وحلها

✓ توثيق المعرفة وكتابة التقارير

✓ أساسيات الأمن السيبراني

✓ اللغة الإنجليزية محادثة وكتابة

04

أدوات وتقنيات مهمة؟

✓ Zendesk

✓ Freshdesk

✓ Jira Service Desk

✓ HubSpot Service Hub

✓ TeamViewer

✓ AnyDesk

✓ Slack

✓ Trello

05

مهام يومية نموذجية؟

✓ استقبال طلب العميل وفهم المشكلة.

✓ تشخيص السبب وتحديد الحل المناسب.

✓ تطبيق الحل أو توجيه العميل للحل.

✓ متابعة الحالة والتأكد من الحل.

✓ توثيق التذكرة وتحديث قاعدة المعرفة.

✓ قياس رضا العميل وجمع ملاحظاته.

مستشار تقنية المعلومات / مستشار تخطيط موارد المؤسسات

لماذا يناسب طالب الرياضيات؟

- ✓ لأنه يعتمد على التحليل المنطقي وحل المشكلات.
- ✓ لأنه يتطلب نمذجة البيانات والعمليات وتحليلها
- ✓ لأنه يحتاج إلى فهم النظم والتكامل واتخاذ القرار
- ✓ لأنه يجمع بين التفكير التقني والفهم التجاري
- ✓ لأنه مجال استشاري واسع الطلب ومتنوّع القطاعات.

ماذا يعمل؟

- ✓ تحليل احتياجات العمل ودراسة الوضع الحالي.
- ✓ تصميم الحلول التقنية والوظيفية المناسبة.
- ✓ تقييم واختيار الأنظمة والتقنيات الملائمة.
- ✓ إدارة مشاريع التنفيذ والتكامل والتغيير.
- ✓ تدريب المستخدمين وتوثيق الإجراءات.

متابعة الأداء والتحسين المستمر.

تقديم الاستشارات والدعم الفني والإداري.

ماذا يجب أن يدرس؟

أساسيات نظم المعلومات وإدارة الأعمال.

- ✓ - تحليل وتصميم النظم System Analysis.
- ✓ -قواعد البيانات.
- ✓ -إدارة المشاريع.
- ✓ -مفاهيم ERP Fundamentals
- ✓ -إدارة التغيير وإدارة المخاطر.
- ✓ - تحليل العمليات والمنذجة BPMN.
- ✓ -ذكاء الأعمال وتحليل البيانات.
- ✓ - مهارات التواصل والتقديم والعرض.
- ✓ -glish for IT Consultants

دوات وتقنيات مهمة

- ✓ - Visio
- ✓ - Bizagi
- ✓ - Power BI
- ✓ - Tableau
- ✓ - SQL
- ✓ - Excel
- ✓ - SAP
- ✓ - ORACLE
- ✓ - Microsoft Dynamics 365
- ✓ - Jira

قطاعات العمل

- ✓ - التعليم
- ✓ - الحكومة
- ✓ - الرعاية الصحية
- ✓ - التجارة والتجارة
- ✓ - التصنيع
- ✓ - القطاع المالي والمصرفي

مدير مشاريع / مدير منتجات / سكرام ماستر

لماذا قد يناسب طالب الرياضيات؟

- ✓ لأنه يعتمد على التفكير التحليلي وحل المشكلات.
- ✓ لأنه يحتاج لتنظيم البيانات والمواعيد والموارد.
- ✓ لأنه يتطلب فهم العلاقات بين المهام والأولويات.
- ✓ لأنه يساعد في اتخاذ قرارات مبنية على المنطق.
- ✓ لأنه يجمع بين التخطيط الاستراتيجي والتنفيذ العملي.
- ✓ لأنه مجال مطلوب في جميع الصناعات والتقنيات.

ماذا يعمل؟

- ✓ تحديد الأهداف والنطاق والنتائج المتوقعة.
- ✓ تخطيط المشروع أو المنتج وتقدير الجهد والوقت.
- ✓ إدارة الفرق والموارد والتواصل مع أصحاب المصلحة.
- ✓ متابعة التقدم وإدارة المخاطر والمشكلات.
- ✓ ضمان جودة التسليم وتحقيق القيمة.
- ✓ تحسين العمليات والتعلم المستمر.

ماذا يجب أن يدرس؟

- ✓ إدارة المشاريع
- ✓ إدارة
- ✓ منهجيات أجايل وسكرام
- ✓ تحليل المتطلبات وتحديد النطاق
- ✓ إدارة المخاطر والوقت والميزانية
- ✓ التواصل الفعال وقيادة الفرق
- ✓ تحليل البيانات ومؤشرات الأداء
- ✓ إدارة الجودة وضمان القيمة
- ✓ أدوات التعاون والتخطيط والتتبع

أدوات مهمة

- ✓ Jira
- ✓ Trello
- ✓ Asana
- ✓ Monday.com
- ✓ Notion
- ✓ Confluence
- ✓ Miro

نصيحة ذهبية

✓ النجاح في إدارة المشاريع أو المنتجات لا يعتمد فقط على الأدوات، بل على فهم الناس، ووضوح الهدف، والتركيز على القيمة.

خطط بذكاء - نفذ بتميز - سلّم قيمة - طوّر باستمرار

محلل نظم / محلل أعمال

01

لماذا يناسب طالب الرياضيات؟

- ✓ لأنه يعتمد على التحليل المنطقي وحل المشكلات.
- ✓ لأنه يحتاج لفهم البيانات واستخلاص النتائج.
- ✓ لأنه يتطلب بناء نماذج وعمليات وتحسينها.
- ✓ لأنه يجمع بين التفكير التحليلي والتواصل الفعّال.
- ✓ لأنه مجال مطلوب في جميع القطاعات.

02

ماذا يعمل؟

- ✓ جمع وتحليل متطلبات العمل من أصحاب المصلحة.
- ✓ دراسة العمليات الحالية وتحديد نقاط التحسين.
- ✓ توثيق المتطلبات في وثائق واضحة ومفصلة.
- ✓ تحليل البيانات واستخلاص الرؤى والتوصيات.
- ✓ التعاون مع فرق التقنية لإيجاد الحلول المناسبة.
- ✓ اختبار الحلول والتأكد من تحقيق الأهداف.
- ✓ دعم اتخاذ القرار من خلال التقارير والتحليلات.

03

ماذا يجب أن يدرس؟

- ✓ أساسيات نظم المعلومات وتحليل النظم
- ✓ إدارة قواعد البيانات SQL
- ✓ تحليل المتطلبات وتوثيقها
- ✓ نمذجة العمليات UML, BPMN
- ✓ تحليل البيانات والإحصاء
- ✓ إدارة المشاريع دورات PMPالأساسية
- ✓ تواصل الأعمال وكتابة التقارير
- ✓ إدارة التغيير والعمليات
- ✓ التفكير التصميمي وحل المشكلات
- ✓ أساسيات البرمجة لفهم الحلول التقنية

04

أدوات وتقنيات مهمة؟

- ✓ Jira
- ✓ Confluence
- ✓ Microsoft Visio
- ✓ Lucidchart
- ✓ draw.io
- ✓ Excel | Power BI
- ✓ SQL
- ✓ Tableau
- ✓ Microsoft Project

05

نصيحة ذهبية

افهم العمل بعمق، حلل بذكاء، وتواصل بوضوح.. لتقدم حلولًا تُحدث فرقًا وتدعم نجاح المؤسسة.

محلل النظم الناجح يبني الجسر بين الأهداف والحلول. | تحليل أفضل = قرارات أفضل = نتائج أفضل..

مصمم واجهات وتجربة المستخدم

لماذا يناسب طالب الرياضيات؟

- ✓ لأنه يعتمد على المنطق وحل المشكلات.
- ✓ لأنه يحتاج فهم البنية والتنظيم والعلاقات.
- ✓ لأنه يعتمد على التحليل واتخاذ القرارات.
- ✓ لأنه يحتاج دقة في القياس والمحاذاة والتخطيط.
- ✓ لأنه يجمع بين الإبداع والتفكير المنطقي..

ماذا يعمل؟

- ✓ تحليل احتياجات العمل ودراسة الوضع الحالي.
- ✓ تصميم الحلول التقنية والوظيفية المناسبة.
- ✓ تقييم واختيار الأنظمة والتقنيات الملائمة.
- ✓ إدارة مشاريع التنفيذ والتكامل والتغيير.
- ✓ تدريب المستخدمين وتوثيق الإجراءات.

متابعة الأداء والتحسين المس

تقديم الاستشارات والدعم
والإداري.

ماذا يجب أن يدرس؟

أساسيات التصميم البصري

مبادئ تجربة المستخدم

- ✓ أبحاث المستخدم
- ✓ تصميم واجهات
- ✓ Wireframing
- ✓ Prototyping
- ✓ التفكير التصميمي
- ✓ سيكولوجية المستخدم
- ✓ قابلية الاستخدام
- ✓ إمكانية الوصول
- ✓ أساسيات HTML & CSS

أدوات وتقنيات مهمة

- ✓ Figma
- ✓ Adobe XD
- ✓ Sketch
- ✓ Photoshop
- ✓ Illustrator
- ✓ Canva
- ✓ InVision
- ✓ Miro
- ✓ Notion
- ✓ Maze

نصيحة ذهبية

المصمم الجيد لا يجعل الأشياء جميلة فقط، بل يجعل الأمور المعقدة سهلة وممتعة للمستخدم.

تصميم جيد + تجربة ممتازة = مستخدم سعيد ومنتج ناجح

ضمان الجودة / اختبار البرمجيات

لماذا يناسب طالب الرياضيات؟

- لأن التفكير المنطقي والتحليل الدقيق مهم جداً.
- لأن فهم الأنماط، الاحتمالات، والإحصاء مفيد.
- لأن حل المشكلات جزء أساسي من العمل.
- لأن الرياضيات تساعد في تحليل البيانات والنتائج.

ماذا يعمل؟

- اختبار البرامج والتطبيقات قبل إطلاقها.
- العثور على الأخطاء Bugs وتوثيقها.
- التأكد من جودة وأداء وأمان البرنامج.
- التعاون مع المطورين لتحسين المنتج.
- كتابة تقارير واضحة عن المشاكل.

ماذا يجب أن يدرس؟

- أساسيات اختبار البرمجيات
- أبحاث المستخدم-
- تقنيات الاختبار اليدوي والآلي.-
- كتابة حالات الاختبار .Test Cases -
- إدارة العيوب .Bug Tracking-
- التعامل مع قواعد SQL- البيانات.

أدوات وتقنيات مهمة

- Postman
- Selenium WebDriver
- JIRA
- TestRail
- Bugzilla
- Git
- Jenkins
- Figma
- Swagger
- SQL

مشاريع تدريبية مناسبة

- كتابة تقرير اختبار كامل.
- العثور على الأخطاء وتوثيقها.
- اختبار تطبيق بسيط على الهاتف.
- اختبار صفحة متجر إلكتروني.
- اختبار صفحة تسجيل دخول لموقع.

أخصائي الأمن السيبراني

لماذا يناسب طالب الرياضيات؟

✓ لأنه يعتمد على التفكير المنطقي والتحليل العميق.

✓ لأنه يحتاج لفهم الرياضيات في التشفير Cryptography.

✓لأنه يتطلب تحليل الأنماط والكشف عن الشذوذ.

✓ لأنه يعتمد على الإحصاء لتقييم المخاطر.

✓ لأنه مجال دائم الطلب وذو رواتب ممتاز

ماذا يعمل؟

• حماية الشبكات والأنظمة من الهجمات.

• اختبار الثغرات واكتشاف نقاط الضعف.

• مراقبة الأنظمة واكتشاف الأنشطة المشبوهة.

• الاستجابة للحوادث الأمنية وتحليلها.

• تطبيق سياسات وإجراءات الأمن.

• التوعية الأمنية للمستخدمين.

ماذا يجب أن يدرس؟

• أساسيات الشبكات

•أنظمة التشغيل

الأمن السيبراني أساسيات ومفاهيم

• الشبكات وأمنها

•تشفير البيانات

•إدارة الهوية والصلاحيات

•اختبار الاختراق

•أمن التطبيقات

•تحليل البرمجيات الخبيثة

• •تحليل السجلات الاستجابة للحوادث

•الامتثال والسياسات

أدوات وتقنيات مهمة

Kali Linux

Wireshark

Nmap

Metasploit

Burp Suite

OSSEC

Splunk

Snort

Suricata

John the Ripper

Hashcat

pfSense

مشاريع تدريبية مناسبة

• إنشاء مختبر منزلي لاختبار الأدوات Home Lab.

•تحليل حركة الشبكة واكتشاف نشاط مشبوه.

• إجراء فحص ثغرات على نظام أو موقع تجريبي.

• تحليل برمجيات خبيثة في بيئة معزولة.

• تصميم خطة استجابة لحادث أمني افتراضي.

• إنشاء تقرير أمني يتضمن التهديدات والحلول.

مهندس DevOps / مهندس الحوسبة السحابية

لماذا قد يناسب طالب الرياضيات؟

- ✓ - لأنه يعتمد على التفكير المنطقي وحل المشكلات.
- ✓ - لأنه يتطلب فهم الخوارزميات والأنظمة.
- ✓ - لأنه يتضمن تصميم هياكل فعالة وقابلة للتوسع.
- ✓ - لأنه يحتاج إلى الإحصاء لتحليل الأداء والمراقبة.
- ✓ - لأنه يجمع بين الرياضيات، البرمجة، والأنظمة.

ماذا يعمل؟

- ✓ - أتمتة نشر التطبيقات وتحديثها.
- ✓ - إدارة البنية التحتية والخوادم والسحابة.
- ✓ - مراقبة الأنظمة وضمان توافرها واستقرارها.
- ✓ - بناء خطوط التكامل والنشر المستمر .CI/CD
- ✓ -ضمان الأمان والأداء وتقليل الأعطال.

ماذا يجب أن يدرس؟

- ✓ - أساسيات شبكات الحاسوب
- ✓ - إدارة الأنظمة
- ✓ - CI/CD Concepts
- ✓ - Docker
- ✓ - Kubernetes
- ✓ - Cloud Basics AWS / Azure / GCP
- ✓ - IaC Infrastructure as Code
- ✓ - Monitoring & Logging
- ✓ - Security Basics
- ✓ - Agile & DevOps Culture

أدوات مهمة

- ✓ Git
- ✓ GitHub
- ✓ Jenkins
- ✓ GitLab CI
- ✓ CircleCI
- ✓ Docker
- ✓ Kubernetes
- ✓ Terraform
- ✓ Ansible
- ✓ AWS
- ✓ Azure
- ✓ Google Cloud

مشاريع تدريبية مناسبة

- ✓ إنشاء خط CI/CD لمشروع بسيط ونشره تلقائياً.
- ✓ تعبئة تطبيق باستخدام Docker وتشغيله.
- ✓ نشر تطبيق على محلياً باستخدام
- ✓ رفع موقع أو تطبيق بسيط على السحابة بناء لوحة مراقبة لمراقبة أداء الخادم.
- ✓ إدارة البنية التحتية باستخدام Terraform أو Ansible.

عالم بيانات / مهندس ذكاء اصطناعي / مهندس تعلم آلة

01

لماذا يناسب طالب الرياضيات؟

- ✓ يعتمد على الإحصاء والاحتمالات
- ✓ يستخدم الجبر الخطي والمصفوفات
- ✓ يحتاج إلى Optimization وتحليل منطقي
- ✓ يرتبط بالنمذجة والتجريد

02

ماذا يعمل؟

- ✓ يحلل كميات كبيرة من البيانات
- ✓ يبني نماذج تتعلم من البيانات
- ✓ يتنبأ بالنتائج المستقبلية
- ✓ يصنف البيانات ويكتشف الأنماط
- ✓ يطور حلولاً للتوصية وتحليل الصور والنصوص

03

ماذا يجب أن يدرس؟

- ✓ Python NumPyPandas
- ✓ Statistics Probability Linear Algebra
- ✓ Data Cleaning Data Visualization
- ✓ Machine Learning Basics Scikit-learn
- ✓ Model Evaluation Neural Networks Basics

04

أدوات وتقنيات مهمة؟

- ✓ **Python**
- ✓ **Jupyter**
- ✓ **NumPy**
- ✓ **Pandas**
- ✓ **Scikit-learn**

TensorFlow / PyTorch

Kaggle

05

مشاريع تدريبية مناسبة

- ✓ توقع علامات الطلاب
- ✓ تصنيف الرسائل Spam / Not Spam
- ✓ تحليل بيانات مبيعات
- ✓ نظام توصية بسيط

مهندس / مطور قواعد البيانات

لماذا يناسب طالب الرياضيات؟

لأنه قريب من المجموعات والعلاقات والمنطق.

لأنه يعتمد على النمذجة والتنظيم والتحليل.

لأنه يتطلب تفكيرًا دقيقًا في البنية والربط.

لأنه يحول المفاهيم المجردة إلى نموذج بيانات منظّم.

ماذا يعمل؟

- ✔ يصمم قواعد البيانات والجداول والعلاقات بينها.
- ✔ يضمن سلامة البيانات ودقتها وعدم تكرارها.
- ✔ يحسن سرعة الاستعلامات والأداء.
- ✔ يدير المفاتيح والعلاقات والفهارس.
- ✔ يساعد الأنظمة على تخزين البيانات واسترجاعها بكفاءة.

ماذا يجب أن يدرس؟

- ✔ SQL بعمق
- ✔ Relational Databases
- ✔ Normalization
- ✔ Primary Key / Foreign Key
- ✔ Indexes
- ✔ Views
- ✔ Stored Procedures
- ✔ Transactions
- ✔ Performance Tuning
- ✔ PostgreSQL / SQL Server / MySQL

أدوات وتقنيات مهمة

التفكير التحليلي

حل المشكلات

الانتباه للتفاصيل

التوثيق

العمل الجماعي

مشاريع تدريبية مناسبة

قاعدة بيانات لمدرسة

نظام فواتير

متجر إلكتروني

نظام حجوزات

إدارة مكتبة

تطوير البرمجيات
Frontend / Backend / Full Stack

لماذا يناسب طالب الرياضيات؟

لأنه يعتمد على المنطق وحل المشكلات.

لأنه يحتاج لفهم الخوارزميات والبني.

لأنه يتعامل مع البيانات والنمذجة.

لأنه يتطلب تفكيراً تحليلياً ودقة.

ماذا يعمل؟

يحول المتطلبات إلى تطبيقات ومواقع وأنظمة.

يكتب كوداً نظيفاً وسهل ليُبسط الصيانة.

يتعامل مع قواعد البيانات والواجهات UI.

يعمل مع APIs والخدمات الخارجية.

يحل المشاكل البرمجية ويطور الأداء.

ماذا يجب أن يدرس؟

أساسيات البرمجة

-Variables

- Data Types

- Loops

- Functions

- OOP Basics

مفاهيم مهمة

- Clean Code

- Debugging

- Problem Solving

- Data Structures Basics

مشاريع تدريبية مناسبة

✓ نظام إدارة طلاب تسجيل - مواد - درجات.

✓ موقع تعليمي بسيط مع تسجيل دخول.

✓ لوحة تحكم Dashboard لعرض البيانات.

✓ API لإدارة بيانات طلاب / كتب / منتجات.

Documentation & Testing

ربط نهائي بالرياضيات

الرياضيات لا تعطيك فقط معادلات ونظريات،
بل تعطيك طريقة تفكير تظهر في:

- ✓ تصميم الأنظمة
- ✓ الذكاء الاصطناعي
- ✓ بناء قواعد البيانات
- ✓ الأمن السيبراني
- ✓ كتابة الخوارزميات
- ✓ تحسين الأداء
- ✓ اتخاذ القرارات
- ✓ تحليل البيانات

لذلك، طالب الرياضيات لا يبدأ من الصفر في البرمجة، بل يبدأ من أرضية فكرية قوية جداً، ويحتاج فقط إلى تحويلها إلى مهارات عملية

الخلاصة

✓ الرياضيات تمنحك طريقة التفكير.
✓ البرمجة تمنحك أداة التطبيق.
✓ سوق العمل يحتاج من يجمع بينهم.

ابدأ بمسار واحد.
ابني مشروعاً صغيراً.
اعرض عملك.
وتعلم باستمرار.

الجملة الختامية

لا تنظر إلى الرياضيات كطريق منفصل عن البرمجة، بل انظر إليها كأساس قوي يمكنك أن تبني عليه مستقبلاً تقنياً واسعاً. ابدأ صغيراً، تعلّم باستمرار، انشئ مشاريع حقيقية، وستكتشف أن التفكير الرياضي الذي تملكه

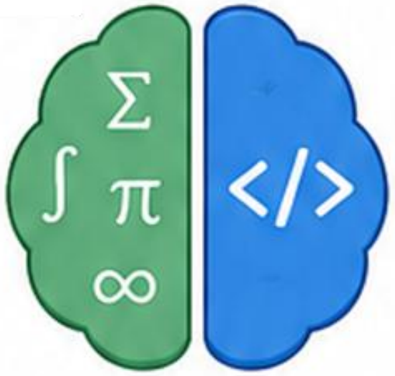
هو واحد من أقوى أدواتك في عالم البرمجيات.

ثق بقدراتك، وابدأ اليوم بخطوة صغيرة نحو مستقبل كبير!

الخاتمة النهائية للمحاضرة
من الرياضيات إلى البرمجة الحديثة
كيف تبدأ رحلتك بثقة؟

الرسالة الأساسية

البرمجة ليست بعيدة عن الرياضيات، وليست عالماً منفصلاً عنها،
بل هي امتداد عملي لطريقة التفكير الرياضي.



طالب الرياضيات يمتلك أساسًا قويًا:

- ✓ التفكير المنطقي
- ✓ التحليل وحل المشكلات
- ✓ التجريد والنمذجة
- ✓ فهم العلاقات والمفاهيم المعقدة

لكن هذا الأساس وحده لا يكفي!

يجب أن يُضاف إليه جانب عملي واضح للدخول إلى سوق العمل التقني.

الرسالة الأهم للطلاب

لا تحاول أن تتعلم كل شيء دفعة واحدة.

ليس مطلوبًا منك أن تصبحَ:

- Frontend Developer
- Backend Developer
- AI Engineer
- DevOps Engineer
- Cybersecurity Specialist
- Data Scientist
- ... وغيرها الكثير

الأفضل أن تبدأ بمسار واحد، وتبني فيه أساساً قوياً

ثم تتوسع تدريجياً

شكراً جزيلاً لحسن استماعكم.