1. **أنظمة ال Authentication :**

* **JWT Authentication System ( JSON Web Token ) :**
* السيرفر يُنشئ **JWT** يتكون من Header + Payload + Signature
* يُرسل للعميل بدون تخزينه في السيرفر.
* كل طلب لاحق يحتوي على JWT، والسيرفر يتحقق من التوقيع فقط (لا يحتاج قاعدة بيانات).

**مميزاته :**

* لا يحتاج تخزين في السيرفر (stateless)
* مناسب جدًا لـ Microservices وDistributed Systems
* يحتوي معلومات المستخدم (claims) داخل التوكن نفسه.
* أداء عالي في التحقق (فقط تحقق من التوقيع).

**عيوبه :**

* حجم التوكن كبير (يزيد من حجم الطلبات).
* لا يمكن إلغاء JWT بسهولة إلا بآليات إضافية (blacklist / short expiry + refresh token)
* إذا انكشف سر التوقيع (secret key) كل التوكنات معرضة للخطر.
* **Session Authentication System :**
* يعتمد على **تخزين الجلسة (session)** في السيرفر (عادةً داخل قاعدة بيانات أو في الذاكرة).
* يرسل السيرفر **Session ID** للعميل ويخزنها في Cookie
* كل طلب لاحق يرسل نفس الـ Session ID ويتم التحقق منه بالسيرفر.

**مميزاته :**

* آمن جدًا إذا تم تفعيل HTTPS + CSRF Tokens
* سهل التنفيذ وموجود في معظم الـ Framework
* يمكن التحكم في الجلسة من السيرفر.

**عيوبه :**

* يحتاج السيرفر لتخزين بيانات لكل جلسة (غير عملي عند وجود ملايين المستخدمين).
* غير مناسب لـ APIs وMobile Apps ( لأنه يعتمد على Cookies **)**
* **Token Authentication System ( Basic Token ) :**
* السيرفر يُنشئ **Token ( رمز عشوائي )**  عند تسجيل الدخول ويخزنه في قاعدة بيانات.
* العميل يخزن الـ Token ويرسله مع كل طلب ) عادةً في الـ ( Authorization Header
* السيرفر يطابقه مع ما هو مخزن في قاعدة البيانات.

**مميزاته :**

* أبسط من JWT
* يسهل إلغاء أي Token مباشرة من قاعدة البيانات.
* مناسب لو APIs صغيرة أو داخل أنظمة داخلية.

**عيوبه :**

* السيرفر يحتاج lookup لكل Token
* أقل أداء من JWT
* لا يحتوي بيانات إضافية ( مثل User Role أو ( Expiryداخل التوكن نفسه
* **مقارنة بين الأنواع الثلاثة :**

|  |  |  |  |
| --- | --- | --- | --- |
| **النظام**  **وجه**  **المقارنة** | SessionAuthentication | TokenAuthentication | JWTAuthentication |
| طريقة التخزين | Session ID محفوظ في السيرفر Cookie +في العميل | Token محفوظ في قاعدة البيانات والسيرفر + عند العميل | JWT محفوظ عند العميل فقط (عادةً في LocalStorage أو Cookie ) |
| اعتماد السيرفر | السيرفر لازم يتذكر كل Session (stateful) | السيرفر لازم يخزن كل Token (stateful) | السيرفر لا يخزن شيء عن الجلسات (stateless) |
| الحجم المرسل | صغير جدًا  Session ID ) قصير) | متوسط  Token ) قصير عشوائي ) | أكبر  JWT )يحتوي Payload  + توقيع ) |
| الأداء | ممتاز للمشاريع الصغيرة  ) لكن يحتاج ذاكرة لتخزين Sessions ) | أبطأ قليلًا لو قاعدة البيانات كبيرة  lookup )لكل طلب ) | أسرع على السيرفر  ) لا ( lookupلكن أبطأ قليلاً بسبب حجم الـ JWT |
| الأمان | يعتمد على Cookie + CSRF Protection | يعتمد على سرية الـ Token HTTPS + | يعتمد على توقيع HMAC أو RSA + HTTPS |
| قابلية التوسع | ضعيف عند الـ load العالي  ( لأن السيرفر لازم يتذكر كل Session ) | متوسط  ( لكن مازال يحتاج قاعدة بيانات Tokens ) | عالي جدًا  Stateless ) السيرفر لا يتذكر شيء) |
| إدارة الجلسة | يمكن إنهاء Session من السيرفر مباشرة | يمكن إلغاء Token بحذفه من DB | لا يمكن إلغاء JWT بسهولة  )إلا بـ blacklist أو تقليل زمن الانتهاء ) |
| سهولة الاستخدام | مدمج افتراضيًا في أغلب Frameworks  (Django, Laravel) | بسيط وسهل التنفيذ | يتطلب مكتبات للتوقيع والتحقق |
| حماية من الهجمات | عرضة لـ CSRF ( إذا Cookie بدون حماية ) | عرضة للسرقة  (Token Hijacking ) | عرضة للسرقة + مشكلة صعوبة إلغاء التوكن قبل انتهاء صلاحيته |
| أفضل استخدام | مواقع تقليدية  Web Apps )قديمة تعتمد على Cookies ) | APIs بسيطة أو داخل نفس الشبكة | Microservices، REST APIs، Mobile Apps، أنظمة موزعة |
| الهيكلية | **Stateful** السيرفر يخزن كل Session في الذاكرة أو DB | **Stateful** السيرفر يخزن كل Token في قاعدة بيانات | السيرفر لا يخزن أي شيء عن الجلسة |
| كيف يتم التحقق؟ | - عند تسجيل الدخول السيرفر ينشئ Session ويخزنها في DB.  - يرسل Session ID للعميل داخل Cookie.  - مع كل طلب العميل يرسل الـ Cookie.  - السيرفر يبحث عن Session ID ويطابقها مع المستخدم. | - عند تسجيل الدخول السيرفر ينشئ Token (رمز عشوائي) ويخزنه في DB.  - يرسل الـ Token للعميل  ( عادةً في Header )  - مع كل طلب العميل يرسل الـ Token في  **Authorization Header**  - السيرفر يبحث عن الـ Token في DB ويتأكد من صلاحيته. | - عند تسجيل الدخول السيرفر ينشئ JWT  (Header + Payload + Signature) ويرسله للعميل.  - العميل يخزن الـ JWT ( في LocalStorage أو Cookie )  - مع كل طلب العميل يرسل الـ JWT في Authorization Header.  - السيرفر يتحقق من **التوقيع الرقمي** فقط (بدون الحاجة لقاعدة بيانات). |

* **مكتبة django-widget-tweaks :**

مرونة أكبر لتخصيص  **Model Forms** و **Form Fields** داخل القوالب (templates) بدون الحاجة لتعريف كل شيء في الكود.

**تثبيت المكتبة :**

**الاعداد في sttings.py :**

**طريقة الاستخدام :**

نعمل اول شيء load في أي templates :

إضافة CSS Classes أو Attributes :

استخدام أكثر من تعديل :

**مثال كامل على ModelForm مع Bootstrap:**

ملف models.py

ملف forms.py

ملفtemplate.html

**مقارنة بين django-widget-tweaks و django-crispy-forms :**

|  |  |  |
| --- | --- | --- |
| **وجه المكتبة**  **المقارنة** | django-widget-tweaks | django-crispy-forms |
| الفكرة الأساسية | تعديلات مباشرة على عناصر الـ Form داخل الـ template  (CSS classes, attributes). | مكتبة كاملة لبناء وتنسيق النماذج بطريقة ديناميكية ومرتبة. |
| طريقة الاستخدام | تعتمد على  **template filters** مثل add\_class, attr. | تعتمد على **FormHelper** و **Layout objects** (Field, Row, Column). |
| سهولة التعلم | سهلة جدًا وبسيطة ) تضيف كلاس أو Attribute في السطر نفسه ) | أعمق وأكثر تعقيدًا لأنها تعطيك تحكم كامل في الـ layout. |
| المناسب لها | تحسينات سريعة على تصميم الـ forms  ( مثلاً مع Bootstrap ) | أنظمة تحتاج **نموذج متكامل** مع ترتيب الأعمدة والأسطر والحقول بشكل منظم. |
| مثال عملي | `{{ form.username | add\_class:"form-control" |
| التوافق | يعمل مع أي CSS framework (Bootstrap, Tailwind, ...). | يدعم ) Bootstrap 4/5, Tailwind بإضافات ) Foundation… |
| الاعتماد على السيرفر | لا يغير شيء في الـ Python form نفسه  ( التغيير فقط في الـ template ) | يغير على مستوى الـ form definition باستخدام **FormHelper**. |
| مستوى المرونة | عالي جدًا لتعديلات صغيرة ومباشرة. | عالي جدًا لكن يحتاج شغل أكثر (مناسب لمشاريع ضخمة). |

* **مكتبة django-extensions :**

مكتبة قوية تضيف مجموعة أوامر إضافية لإدارة مشروع Django ، ومن أهمهاالأمر graph\_models اللي يستخرج ال  **ERD (Entity Relationship Diagram)** أو **Schema للموديلات.**

**تثبيت المكتبة :**

**إضافة المكتبة للمشروع :**

**استخرج كود الـ Schema :**

= -a كل التطبيقات (apps)

-o schema.dot = يخرج ملف نصي (DOT format)

**نفتح ملف المشروع :**

سنجد ملف اسمه schema.dot بجانب المشروع

نفتحه بأي محرر نصوص (Notepad, VS Code) وننسخ المحتوى.

**استخدام موقع رسم الـ Schema :**

نفتح موقع **Graphviz Online** ثم نلصق المحتوى المنسوخ في مربع الإدخال، ونضغط **Generate** يطلع لنا الـ **Schema Diagram** لمشروعك

**ال Schema للتنفيذ حق المحاضرات بااستخدام هذه المكتبة :**