To assist a judge in managing affairs, schedules, and other daily tasks, particularly in an Arabic language system, you'll need a domain model that handles relationships between affairs, procedures, and scheduling.

Here’s a draft domain model to help with scheduling **Affairs** based on their **Procedures**:

**Entities:**

1. **Judge**
   * **Attributes:**
     + id (Unique identifier)
     + name (Judge's name)
     + email (Contact details)
     + assignedAffairs[] (List of affairs assigned to the judge)
2. **Affair**
   * **Attributes:**
     + id (Unique identifier)
     + title (Title of the affair)
     + description (Details of the affair)
     + startDate (Date when the affair was initiated)
     + status (Open, Closed, etc.)
     + assignedJudgeId (Link to Judge handling the affair)
     + procedures[] (List of procedures associated with the affair)
3. **Procedure**
   * **Attributes:**
     + id (Unique identifier)
     + affairId (Link to associated affair)
     + type (Type of procedure, e.g., Hearing, Documentation, Appeal)
     + status (Pending, Completed, Scheduled, etc.)
     + date (Date of procedure completion or schedule)
     + dueDate (Date by which the procedure must be completed)
     + scheduledBy (Link to user who scheduled the procedure)
4. **Schedule**
   * **Attributes:**
     + id (Unique identifier)
     + affairId (Link to associated affair)
     + procedureId (Link to associated procedure)
     + scheduledDate (Date when the affair is scheduled)
     + isCompleted (Boolean to mark if the affair has been handled or not)
5. **User** (This can represent either a **Judge** or **Clerk** scheduling affairs)
   * **Attributes:**
     + id (Unique identifier)
     + name (Name of the user)
     + role (Role in the system, e.g., Judge, Clerk, Admin)

**Relationships:**

* A **Judge** can handle multiple **Affairs**.
* An **Affair** can have multiple **Procedures**.
* A **Procedure** is always associated with an **Affair**.
* A **Schedule** is associated with an **Affair** and a **Procedure**.
* **Procedures** dictate the **Schedule** of the affair based on either completed or scheduled dates.

**Scheduling Logic:**

To schedule an **Affair** according to its latest **Procedure**:

* The system should always fetch the **latest procedure** based on either the date (if completed) or dueDate (if pending or scheduled).
* The system can then generate a **Schedule** based on this information, ensuring the next action is based on the latest procedure's date or status.

**Sample Domain Model in UML-like Format:**

plaintext

Copier le code

Judge

------------------------

- id: Integer

- name: String

- email: String

- assignedAffairs[]: Affair[]

Affair

------------------------

- id: Integer

- title: String

- description: String

- startDate: Date

- status: String

- assignedJudgeId: Integer (FK to Judge)

- procedures[]: Procedure[]

Procedure

------------------------

- id: Integer

- affairId: Integer (FK to Affair)

- type: String

- status: String

- date: Date

- dueDate: Date

- scheduledBy: Integer (FK to User)

Schedule

------------------------

- id: Integer

- affairId: Integer (FK to Affair)

- procedureId: Integer (FK to Procedure)

- scheduledDate: Date

- isCompleted: Boolean

User

------------------------

- id: Integer

- name: String

- role: String (Judge, Clerk, Admin)

**Example Use Case:**

1. **Affair Creation**: The clerk creates a new affair and assigns it to a judge.
2. **Procedure Tracking**: As the affair progresses, different procedures (such as hearings, documentation, appeals) are created. Each procedure has a date and dueDate.
3. **Scheduling**: The system schedules the next hearing or action based on the last procedure’s dueDate or status. If the last procedure is marked as completed, it schedules the next procedure accordingly.
4. **Judge Views**: The judge sees all assigned affairs and their upcoming procedures based on the scheduling system.

This domain model should help manage the flow of judicial work and procedures, while also allowing for smooth scheduling based on the last or scheduled procedure.

Let me know if you'd like to expand on any particular aspect or need additional attributes for the entities.

4o

**Vous avez dit :**

please translate to arbic :)

**ChatGPT a dit :**

ChatGPT

إليك النموذج المقترح لإدارة القضايا والإجراءات اليومية للقضاة باللغة العربية:

**الكيانات:**

1. **القاضي**
   * **الخصائص:**
     + معرف (معرف فريد)
     + الاسم (اسم القاضي)
     + البريد الإلكتروني (معلومات الاتصال)
     + القضايا\_المسندة[] (قائمة بالقضايا المسندة للقاضي)
2. **القضية**
   * **الخصائص:**
     + معرف (معرف فريد)
     + العنوان (عنوان القضية)
     + الوصف (تفاصيل القضية)
     + تاريخ\_البداية (تاريخ بدء القضية)
     + الحالة (مفتوحة، مغلقة، إلخ.)
     + معرف\_القاضي (رابط إلى القاضي المسؤول عن القضية)
     + الإجراءات[] (قائمة بالإجراءات المرتبطة بالقضية)
3. **الإجراء**
   * **الخصائص:**
     + معرف (معرف فريد)
     + معرف\_القضية (رابط إلى القضية المرتبطة)
     + النوع (نوع الإجراء، مثل جلسة، وثائق، استئناف)
     + الحالة (قيد الانتظار، مكتمل، مجدول، إلخ.)
     + التاريخ (تاريخ اكتمال أو جدولة الإجراء)
     + تاريخ\_الاستحقاق (التاريخ الذي يجب فيه الانتهاء من الإجراء)
     + مجدول\_بواسطة (رابط إلى المستخدم الذي قام بجدولة الإجراء)
4. **الجدول**
   * **الخصائص:**
     + معرف (معرف فريد)
     + معرف\_القضية (رابط إلى القضية المرتبطة)
     + معرف\_الإجراء (رابط إلى الإجراء المرتبط)
     + تاريخ\_الجدولة (تاريخ الجدولة)
     + مكتمل (علامة تحدد إذا كانت القضية قد تمت معالجتها أم لا)
5. **المستخدم** (يمكن أن يكون هذا الكيان **قاضي** أو **كاتب** يقوم بجدولة القضايا)
   * **الخصائص:**
     + معرف (معرف فريد)
     + الاسم (اسم المستخدم)
     + الدور (دور المستخدم في النظام، مثل قاضي، كاتب، مدير)

**العلاقات:**

* يمكن أن يتولى **القاضي** مسؤولية عدة **قضايا**.
* يمكن أن تحتوي **القضية** على عدة **إجراءات**.
* كل **إجراء** مرتبط دائمًا بقضية.
* يتم ربط **الجدول** بكل من **القضية** و**الإجراء**.
* **الإجراءات** تحدد **جدولة** القضية بناءً على تواريخ أو حالة الإجراءات المكتملة أو المجدولة.

**منطق الجدولة:**

لجدولة **القضية** بناءً على أحدث **إجراء**:

* يجب على النظام دائمًا استرداد **أحدث إجراء** بناءً على التاريخ (إذا كان مكتملًا) أو تاريخ\_الاستحقاق (إذا كان قيد الانتظار أو مجدولًا).
* يمكن للنظام بعد ذلك إنشاء **جدول** استنادًا إلى هذه المعلومات، لضمان أن يكون الإجراء التالي مستندًا إلى تاريخ أو حالة الإجراء الأخير.

**نموذج المجال في صيغة تشبه UML:**

plaintext

Copier le code

القاضي

------------------------

- معرف: عدد صحيح

- الاسم: نص

- البريد الإلكتروني: نص

- القضايا\_المسندة[]: قضية[]

القضية

------------------------

- معرف: عدد صحيح

- العنوان: نص

- الوصف: نص

- تاريخ\_البداية: تاريخ

- الحالة: نص

- معرف\_القاضي: عدد صحيح (مفتاح خارجي إلى القاضي)

- الإجراءات[]: إجراء[]

الإجراء

------------------------

- معرف: عدد صحيح

- معرف\_القضية: عدد صحيح (مفتاح خارجي إلى القضية)

- النوع: نص

- الحالة: نص

- التاريخ: تاريخ

- تاريخ\_الاستحقاق: تاريخ

- مجدول\_بواسطة: عدد صحيح (مفتاح خارجي إلى المستخدم)

الجدول

------------------------

- معرف: عدد صحيح

- معرف\_القضية: عدد صحيح (مفتاح خارجي إلى القضية)

- معرف\_الإجراء: عدد صحيح (مفتاح خارجي إلى الإجراء)

- تاريخ\_الجدولة: تاريخ

- مكتمل: بوليان

المستخدم

------------------------

- معرف: عدد صحيح

- الاسم: نص

- الدور: نص (قاضي، كاتب، مدير)

**مثال لحالة استخدام:**

1. **إنشاء قضية**: يقوم الكاتب بإنشاء قضية جديدة ويسندها إلى القاضي.
2. **متابعة الإجراءات**: مع تقدم القضية، يتم إنشاء إجراءات مختلفة (مثل الجلسات، الوثائق، الاستئناف). لكل إجراء تاريخ وتاريخ\_استحقاق.
3. **الجدولة**: يقوم النظام بجدولة الجلسة أو الإجراء التالي بناءً على تاريخ\_الاستحقاق أو حالة الإجراء الأخير.
4. **عرض القاضي**: يستطيع القاضي مشاهدة جميع القضايا المسندة والإجراءات القادمة بناءً على نظام الجدولة.

هذا النموذج يجب أن يساعد في إدارة سير العمل القضائي والإجراءات مع توفير جدول زمني يعتمد على أحدث أو مجدول إجراء.

أخبرني إذا كنت ترغب في توسيع أي جزء معين أو إضافة خصائص أخرى للكيانات