

خیلی خوب، متوجه منظور شما شدم! شما می‌خواهید بدونید که آیا با سیستم فعلی (یا با اصلاحات پیشنهادی) می‌شه
یه ساختار سلسله‌مراتبی برای سازمان تعریف کرد که یه دفتر مرکزی هلدینگ داشته باشه، زیرش معاونت‌ها باشن، و هر
معاونت شعبه‌ها و زیرشعبه‌هایی داشته باشه. نکته‌ی کلیدی اینه که می‌خواهید پست‌ها رو طوری تعریف کنید که حتی اگر
سطح (level) یکسانی داشته باشن (مثلاً عدد یکسان)، بتونن به معاونت، سازمان، شعبه یا زیرشعبه‌ی خاصی وابسته باشن
و سلسله‌مراتب رعایت بشه. کد نمی‌نویسم و فقط سناریو رو با جزئیات پیش می‌برم تا ببینیم چطور می‌تونیم این ساختار رو
پیاده کنیم و مشکلات رو حل کنیم.

تحلیل سناریو و نیازها

1. ساختار سازمانی مدنظر

فرض کنیم ساختار سازمانی شما به این شکله:

دفتر مرکزی هلدینگ: یه سازمان اصلی که کل هلدینگ رو مدیریت می‌کنه.

معاونت‌ها: زیرمجموعه‌ی هلدینگ (مثلاً معاونت مالی، معاونت عملیات، معاونت منابع انسانی).

شعبه‌ها: هر معاونت چند شعبه داره (مثلاً شعبه‌ی تهران، شعبه‌ی شیراز).

زیرشعبه‌ها: هر شعبه می‌تونه زیرشعبه‌هایی داشته باشه (مثلاً زیرشعبه‌ی هتل لاله سرعین زیر شعبه‌ی تهران).

هر کدوم از این سطوح می‌تونن پست‌هایی داشته باشن (مثل مدیر مالی، کارشناس عملیات). شما می‌خواهید:

پست‌ها به سازمان/معاونت/شعبه/زیرشعبه‌ی خاص خودشون وابسته باشن.

حتی اگر سطح (level) پست‌ها یکسان باشه (مثلاً $level=3$ برای مدیر مالی شعبه‌ی تهران و مدیر مالی شعبه‌ی شیراز)،
مشخص باشه که هر پست به کدوم بخش از سلسله‌مراتب تعلق داره.

سلسله‌مراتب در دسترسی‌ها و گردش کار (مثل تأیید فاکتور یا امضای دستور پرداخت) رعایت بشه.

2. چالش‌های سیستم فعلی

بر اساس مدل‌هایی که قبلاً ارائه دادید:

مدل Organization:

فقط یه سازمان می‌تونه `is_core=True` باشه، که برای هلدینگ با چندین معاونت یا شعبه‌ی مستقل محدودیت ایجاد
می‌کنه.

هیچ فیلدی برای تعریف سلسله‌مراتب (مثل سازمان والد یا زیرمجموعه) وجود ندارد.

مدل Post:

پست‌ها به یه organization وابسته‌ان و فیلد level برای تعیین سطحشون استفاده می‌شه.

هیچ مکانیزمی برای تمایز پست‌های با سطح یکسان در سازمان‌های مختلف (مثلاً مدیر مالی تهران در مقابل مدیر مالی شیراز) وجود نداره، جز از طریق organization.

فیلد branch (مثل OPS یا FIN) برای دسته‌بندی پست‌ها استفاده می‌شه، ولی برای سلسله‌مراتب کافی نیست.

مدل WorkflowStage و PostAction:

دسترسی‌ها (مثل APPROVE یا SIGN_PAYMENT) بر اساس PostAction تعریف می‌شن، ولی وابستگی به سازمان خاص به‌خوبی مدیریت نمی‌شه.

اگر دو پست با سطح یکسان (مثلاً level=3) در دو شعبه‌ی مختلف باشن، ممکنه دسترسی‌هاشون قاطی بشه، چون سیستم فقط به level و branch نگاه می‌کنه.

مشکل اصلی:

نبود یه ساختار درختی (Tree Hierarchy) برای سازمان‌ها که بتونه معاونت‌ها، شعبه‌ها، و زیرشعبه‌ها رو به‌صورت واضح تعریف کنه.

عدم تمایز واضح بین پست‌های با سطح یکسان در بخش‌های مختلف سلسله‌مراتب.

محدودیت تک‌دفتر مرکزی که برای هلدینگ مناسب نیست.

3. سؤال برای روشن شدن سناریو

قبل از اینکه سناریو رو پیش ببرم، چند نکته رو روشن کنیم:

آیا هر معاونت مثل یه سازمان مستقل عمل می‌کنه (مثلاً قوانین دسترسی و گردش کار خاص خودش رو داره) یا همه زیر نظر هلدینگن؟

آیا پست‌های با سطح یکسان (مثلاً مدیر مالی تهران و شیراز هر دو level=3) باید دسترسی‌های متفاوتی داشته باشن؟ مثلاً مدیر مالی تهران فقط فاکتورهای شعبه‌ی تهران رو تأیید کنه؟

آیا زیرشعبه‌ها (مثلاً هتل لاله سرعین) نیاز به گردش کار جداگونه دارن یا زیرمجموعه‌ی شعبه‌ی اصلی محسوب می‌شن؟

آیا یه پست می‌تونه به چند سازمان (مثلاً هم هلدینگ و هم شعبه) وابسته باشه؟

فرض می‌کنم:

هر معاونت و شعبه قوانین دسترسی خاص خودش رو داره، ولی زیر نظر هلدینگ عمل می‌کنه.
پست‌های با سطح یکسان باید فقط به سازمان خودشون (شعبه/زیرشعبه) دسترسی داشته باشن.
زیرشعبه‌ها زیرمجموعه‌ی شعبه‌ان و گردش کارشون به شعبه وابسته‌ست.

سناریوی پیشنهادی

برای پیاده‌سازی این ساختار، باید سیستم رو طوری طراحی کنیم که:

ساختار درختی سازمان‌ها رو پشتیبانی کنه.

پست‌ها به‌صورت واضح به یه بخش از سلسله‌مراتب وابسته باشن.

دسترسی‌ها و گردش کار به سازمان خاص محدود بشه، حتی برای پست‌های با سطح یکسان.

1. طراحی ساختار درختی سازمان‌ها

ایده:

هر سازمان (هلدینگ، معاونت، شعبه، زیرشعبه) یه نمونه از مدل Organizationه.

یه فیلد parent_organization تعریف می‌کنیم تا سلسله‌مراتب رو مشخص کنه (مثلاً شعبه‌ی تهران زیرمجموعه‌ی معاونت عملیات، معاونت عملیات زیرمجموعه‌ی هلدینگ).

به‌جای محدودیت تک‌دفتر مرکزی، اجازه می‌دیم چندین سازمان is_core=True باشن یا یه فیلد جدید (مثل is_holding) برای هلدینگ اضافه کنیم.

مثال ساختار:

هلدینگ (is_holding=True)

└─ معاونت مالی

└─ شعبه‌ی تهران

└─ | | زیرشعبه‌ی هتل لاله تهران

└─ | | شعبه‌ی شیراز

— | معاونت عملیات

— | شعبه‌ی مشهد

— | شعبه‌ی تبریز

— | معاونت منابع انسانی

— | شعبه‌ی مرکزی

چطور کار می‌کنه؟

هلدینگ (مثلاً "هلدینگ پارسیان") به‌عنوان ریشه‌ی درخت تعریف می‌شه.

معاونت‌ها (مثل معاونت مالی) به‌عنوان فرزندان مستقیم هلدینگ ثبت می‌شن.

شعبه‌ها (مثل تهران) زیر معاونت‌ها و زیرشعبه‌ها (مثل هتل لاله تهران) زیر شعبه‌ها قرار می‌گیرن.

هر سازمان یه کد منحصر به فرد (مثل HOLDING, FIN-TEHRAN, FIN-TEHRAN-LALEH) داره.

2. تعریف پست‌ها با وابستگی به سلسله‌مراتب

ایده:

هر پست به یه Organization خاص وابسته‌ست (مثلاً مدیر مالی شعبه‌ی تهران به سازمان FIN-TEHRAN).

فیلد level برای مشخص کردن اهمیت پست در سازمان استفاده می‌شه، اما وابستگی به سازمان (از طریق organization) مشخص می‌کنه که پست به کدوم بخش از سلسله‌مراتب تعلقه.

برای پست‌های با سطح یکسان (مثلاً level=3 برای مدیر مالی تهران و شیراز)، سیستم با بررسی organization تشخیص می‌ده که هر پست به کدوم شعبه مربوط می‌شه.

مثال پست‌ها:

پست: "مدیر مالی تهران"

organization: شعبه‌ی تهران (FIN-TEHRAN)

branch: FIN

level: ۳

پست: "مدیر مالی شیراز"

organization: شعبه‌ی شیراز (FIN-SHIRAZ)

branch: FIN

level: ۳

پست: "کارشناس عملیات هتل لاله تهران"

organization: زیرشعبه‌ی هتل لاله تهران (FIN-TEHRAN-LALEH)

branch: OPS

level: ۲

چطور کار می‌کنه؟

وقتی یه کاربر (مثلاً snejate) می‌خواد فاکتوری رو تأیید کنه، سیستم چک می‌کنه که پست کاربر (مثلاً "مدیر مالی تهران") به سازمان مرتبط با فاکتور (مثلاً FIN-TEHRAN) وابسته‌ست یا نه.

اگر فاکتور مربوط به شعبه‌ی شیراز باشه، مدیر مالی تهران نمی‌تونه تأیید کنه، حتی اگه هر دو level=3 داشته باشن.

3. مدیریت دسترسی‌ها و گردش کار

ایده:

دسترسی‌ها (مثل APPROVE, REJECT, SIGN_PAYMENT) به سازمان خاص محدود می‌شن.

یه جدول تنظیمات (مثل AccessRule) تعریف می‌کنیم که مشخص کنه کدوم پست‌ها (بر اساس branch, level, و organization) چه اقداماتی می‌تونن توی کدوم مراحل انجام بدن.

برای پست‌های با سطح یکسان، وابستگی به organization تعیین می‌کنه که دسترسی به کدوم داده‌ها (مثلاً فاکتورهای شعبه‌ی تهران) دارن.

مثال دسترسی‌ها:

شعبه‌ی تهران:

مدیر مالی تهران (FIN, level=3) می‌تونه فاکتورهای شعبه‌ی تهران رو تأیید کنه (APPROVE در مرحله order=1).

فقط پست‌های با `is_payment_order_signer=True` در شعبه‌ی تهران می‌تونن دستور پرداخت (SIGN_PAYMENT) رو امضا کنن.

شعبه‌ی شیراز:

مدیر مالی شیراز (`level=3, FIN`) فقط فاکتورهای شیراز رو تأیید می‌کنه.

هلدینگ:

پست‌های سطح بالا در هلدینگ (مثلاً مدیرکل مالی، `level=5`) می‌تونن فاکتورهای همه شعبه‌ها رو ببینن یا تأیید کنن.

چطور کار می‌کنه؟

وقتی یه فاکتور ثبت می‌شه، به یه Tankhah وابسته‌ست که خودش به یه Organization (مثلاً FIN-TEHRAN) ربط داره.

سیستم چک می‌کنه که آیا پست کاربر (از طریق UserPost) به همون سازمان یا سازمان والد (مثلاً معاونت مالی یا هلدینگ) وابسته‌ست.

برای دستور پرداخت، فقط پست‌های مجاز در سازمان مرتبط (مثلاً FIN-TEHRAN) می‌تونن امضا کنن.

4. مدیریت دستور پرداخت در سلسله‌مراتب

ایده:

دستور پرداخت (PaymentOrder) به یه Tankhah وابسته‌ست که خودش به یه سازمان خاص ربط داره.

فقط پست‌های سازمان مرتبط (یا سازمان والد، مثل معاونت یا هلدینگ) می‌تونن دستور پرداخت رو امضا کنن.

مثلاً، برای فاکتور شعبه‌ی تهران، فقط مدیر مالی تهران یا پست‌های سطح بالای معاونت مالی/هلدینگ می‌تونن امضا کنن.

مثال:

فاکتور در شعبه‌ی تهران تأیید می‌شه (مرحله `order=2`).

سیستم به PaymentOrder برای شعبه‌ی تهران ایجاد می‌کنه.

مدیر مالی تهران (`level=3, FIN, is_payment_order_signer=True`) می‌تونه امضا کنه.

اگر نیاز به تأیید بالاتر باشه (مثلاً برای مبالغ بزرگ)، مدیرکل مالی هلدینگ (`level=5`) هم می‌تونه امضا کنه.

5. پشتیبانی از پست‌های با سطح یکسان

ایده:

چون هر پست به یه organization خاص وابسته‌ست، حتی اگه سطحشون یکسان باشه (مثلاً $level=3$)، سیستم با بررسی organization تشخیص می‌ده که پست به کدوم بخش تعلقه.

برای دسترسی‌ها، AccessRule برای هر سازمان جداگانه تعریف می‌شه.

مثال:

مدیر مالی تهران ($level=3$) فقط به فاکتورهای FIN-TEHRAN دسترسی داره.

مدیر مالی شیراز ($level=3$) فقط به فاکتورهای FIN-SHIRAZ دسترسی داره.

اگر هلدینگ بخواد نظارت کنه، یه پست مثل "مدیر کل مالی هلدینگ" ($level=5$) می‌تونه به همه شعبه‌ها دسترسی داشته باشه.

آیا سیستم فعلی این سناریو رو پشتیبانی می‌کنه؟

نقاط قوت سیستم فعلی:

وابستگی پست به سازمان:

مدل Post فیلد organization داره، پس می‌تونیم پست‌ها رو به معاونت/شعبه/زیرشعبه‌ی خاص وابسته کنیم.

مثلاً، مدیر مالی تهران به organization=FIN-TEHRAN و مدیر مالی شیراز به organization=FIN-SHIRAZ ربط داره.

دسترسی‌ها با PostAction:

PostAction می‌تونه مشخص کنه که کدوم پست‌ها چه اقداماتی (مثل APPROVE یا SIGN_PAYMENT) می‌تونن انجام بدن.

با اصلاحات پیشنهادی (مثل AccessRule)، می‌تونیم دسترسی‌ها رو به سازمان خاص محدود کنیم.

گردش کار:

مدل WorkflowStage امکان تعریف مراحل مختلف (مثلاً بررسی اولیه، تأیید نهایی) رو می‌ده.

پیش‌رفت خودکار مراحل (که گفتید بعد از تأیید فاکتور اتفاق می‌افته) با این سناریو سازگاره.

محدودیت‌های سیستم فعلی:

عدم پشتیبانی از سلسله‌مراتب سازمانی:

مدل Organization هیچ فیلدی برای تعریف والد (مثل parent_organization) ندارد.

محدودیت تک‌دفتر مرکزی (is_core=True) برای هلدینگ با چندین معاونت مناسب نیست.

مدیریت دسترسی‌های پیچیده:

PostAction به سازمان خاص وابسته نیست، پس ممکنه مدیر مالی تهران به فاکتورهای شیراز دسترسی پیدا کنه.

قوانین دسترسی هاردکد شدن (مثلاً `branch == 'FIN' and level <= 3`)، که برای سازمان‌های مختلف انعطاف‌پذیر نیست.

تمایز پست‌های با سطح یکسان:

اگرچه organization پست‌ها رو جدا می‌کنه، ولی منطق دسترسی‌ها (مثل can_edit_approval) فقط به level و branch نگاه می‌کنه و ممکنه به سازمان توجه نکنه.

نتیجه:

سیستم فعلی با اصلاحات می‌تونه این سناریو رو پشتیبانی کنه، ولی بدون تغییرات زیر نمی‌تونه به‌خوبی کار کنه:

اضافه کردن سلسله‌مراتب به Organization (مثل parent_organization).

پویا کردن قوانین دسترسی با یه جدول تنظیمات (مثل AccessRule) که به سازمان وابسته باشه.

محدود کردن دسترسی‌ها به سازمان مرتبط با پست کاربر.

سناریوی بهبودیافته

1. ساختار درختی سازمان‌ها

هلدینگ به‌عنوان ریشه‌ی درخت تعریف می‌شه (is_holding=True).

معاونت‌ها (مثل مالی، عملیات) به‌عنوان فرزندان هلدینگ ثبت می‌شن.

شعبه‌ها زیر معاونت‌ها و زیرشعبه‌ها زیر شعبه‌ها قرار می‌گیرن.

هر سازمان یه کد منحصربه‌فرد داره (مثل FIN-TEHRAN, FIN-TEHRAN-LALEH, HOLDING).

برای دسترسی‌های بالاتر، سیستم چک می‌کند که آیا پست کاربر به سازمان والد (مثلاً هلدینگ یا معاونت) وابسته‌ست یا نه.

2. تعریف پست‌ها

هر پست به یه سازمان خاص (مثلاً FIN-TEHRAN) وابسته‌ست.

سطح (level) فقط اهمیت پست رو در سازمان خودش نشون می‌ده، نه در کل هلدینگ.
مثلاً:

مدیر مالی تهران: `branch=FIN ,level=3 ,organization=FIN-TEHRAN`.

مدیر مالی شیراز: `branch=FIN ,level=3 ,organization=FIN-SHIRAZ`.

مدیرکل مالی هلدینگ: `branch=FIN ,level=5 ,organization=HOLDING`.

3. دسترسی‌ها

یه جدول تنظیمات (مثل AccessRule) برای هر سازمان تعریف می‌کنیم:

شعبه‌ی تهران: مدیر مالی (`level=3`, FIN) می‌تونه `APPROVE` و `SIGN_PAYMENT` برای فاکتورها و دستورات پرداخت تهران انجام بده.

شعبه‌ی شیراز: همین‌طور، ولی فقط برای شیراز.

هلدینگ: مدیرکل مالی (`level=5`) می‌تونه به همه شعبه‌ها دسترسی داشته باشه.

موقع تأیید یا امضاء، سیستم چک می‌کند که سازمان پست کاربر با سازمان فاکتور/دستور پرداخت مطابقت داره یا کاربر به سازمان والد دسترسی داره.

4. گردش کار

فاکتورها و دستورات پرداخت به سازمان خاصی وابسته‌ان (از طریق Tankhah).

مراحل گردش کار (مثل بررسی اولیه، تأیید نهایی) برای هر سازمان می‌تونه جداگانه تعریف بشه.
مثلاً:

شعبه‌ی تهران: مرحله ۱ (بررسی توسط کارشناس)، مرحله ۲ (تأیید مدیر مالی)، مرحله ۳ (ایجاد دستور پرداخت).

هلدینگ: یه مرحله اضافی برای تأیید مبالغ بزرگ توسط مدیرکل.

پیش‌رفت خودکار مراحل (که گفتید وجود داره) برای هر سازمان کار می‌کنه.

5. مدیریت پست‌های با سطح یکسان

چون هر پست به یه `organization` وابسته‌ست، سیستم می‌تونه تشخیص بده که مدیر مالی تهران فقط به داده‌های تهران دسترسی داره.

اگر مدیرکل هلدینگ بخواد نظارت کنه، پستش به `organization=HOLDING` وابسته‌ست و قوانین دسترسیش اجازه می‌ده به همه زیرمجموعه‌ها (مثل تهران و شیراز) دسترسی داشته باشه.

چطور این سناریو رو پیاده کنیم؟

قدم ۱: تعریف ساختار درختی

مدل `Organization` رو اصلاح کنیم تا فیلد `parent_organization` داشته باشه.

محدودیت تک‌دفتر مرکزی رو برداریم.

سازمان‌ها رو به‌صورت درختی تعریف کنیم:

هلدینگ: `code=HOLDING, is_holding=True`

معاونت مالی: `code=FIN, parent_organization=HOLDING`

شعبه‌ی تهران: `code=FIN-TEHRAN, parent_organization=FIN`

زیرشعبه‌ی هتل لاله تهران: `code=FIN-TEHRAN-LALEH, parent_organization=FIN-TEHRAN`

قدم ۲: تعریف پست‌ها

برای هر سازمان، پست‌های موردنیاز رو تعریف کنیم:

شعبه‌ی تهران: مدیر مالی (`branch=FIN, level=3`), کارشناس مالی (`branch=FIN, level=2`).

شعبه‌ی شیراز: همین‌طور.

هلدینگ: مدیرکل مالی (`branch=FIN, level=5`).

برای امضای دستور پرداخت، فقط پست‌های خاصی (مثلاً مدیر مالی تهران) رو به‌عنوان

`is_payment_order_signer=True` علامت‌گذاری کنیم.

قدم ۳: تنظیم دسترسی‌ها

یه جدول تنظیمات (مثل AccessRule) برای هر سازمان تعریف کنیم:

شعبه‌ی تهران: پست‌های FIN با $\text{level} \leq 3$ می‌تونن APPROVE و SIGN_PAYMENT انجام بدن.

هلدینگ: پست‌های $\text{level} \leq 5$ می‌تونن به همه شعبه‌ها دسترسی داشته باشن.

موقع تأیید یا امضا، سیستم چک کنه که سازمان پست کاربر با سازمان داده (فاکتور/دستور پرداخت) مطابقت داره یا کاربر به سازمان والد دسترسی داره.

قدم ۴: گردش کار

برای هر سازمان، مراحل گردش کار رو تعریف کنیم.

مثلاً:

شعبه‌ی تهران: بررسی اولیه ($\text{order}=1$), تأیید نهایی ($\text{order}=2$), ایجاد دستور پرداخت ($\text{order}=3$).

هلدینگ: یه مرحله اضافی برای نظارت.

پیش‌رفت خودکار مراحل رو برای هر سازمان فعال نگه داریم.

قدم ۵: تست سناریو

یه فاکتور برای شعبه‌ی تهران ثبت کنیم.

چک کنیم که فقط مدیر مالی تهران بتونه تأیید کنه و مدیر مالی شیراز دسترسی نداشته باشه.

بعد از تأیید، دستور پرداخت ایجاد بشه و فقط پست‌های مجاز (مثلاً مدیر مالی تهران یا مدیر کل هلدینگ) بتونن امضا کنن.

مشکلات احتمالی و راه‌حل‌ها

پیچیدگی مدیریت دسترسی‌ها:

مشکل: اگر تعداد سازمان‌ها زیاد بشه، تعریف دستی قوانین دسترسی زمان‌بره.

راه‌حل: یه رابط کاربری (مثل Django Admin) برای مدیریت قوانین دسترسی درست کنیم. همچنین، می‌تونیم قوانین

پیش‌فرض برای سازمان‌های جدید تعریف کنیم.

دسترسی به سازمان‌های والد:

مشکل: مدیر کل هلدینگ باید به داده‌های همه شعبه‌ها دسترسی داشته باشد، ولی سیستم فعلی این رو به‌خوبی مدیریت نمی‌کنه.

راه‌حل: موقع بررسی دسترسی، چک کنیم که آیا پست کاربر به سازمان والد (مثلاً هلدینگ یا معاونت) وابسته‌ست یا نه. تداخل پست‌های با سطح یکسان:

مشکل: اگر قوانین دسترسی فقط به level وابسته باشن، مدیر مالی تهران ممکنه به فاکتورهای شیراز دسترسی پیدا کنه. راه‌حل: همیشه سازمان پست رو با سازمان داده مقایسه کنیم.

نتیجه‌گیری و قدم بعدی

بله، با اصلاحات پیشنهادی، سیستم می‌تونه این سناریو رو پشتیبانی کنه! می‌تونیم:

یه ساختار درختی برای سازمان‌ها (هلدینگ، معاونت‌ها، شعبه‌ها، زیرشعبه‌ها) تعریف کنیم.

پست‌ها رو به سازمان خاص وابسته کنیم و حتی با سطح یکسان، با بررسی organization تمایزشون بدیم.

دسترسی‌ها و گردش کار رو به سازمان خاص محدود کنیم و برای هلدینگ امکان نظارت کلی فراهم کنیم.

قدم بعدی:

تأیید سناریو: لطفاً بگید این ساختار (هلدینگ → معاونت → شعبه → زیرشعبه) دقیقاً همونه که می‌خواهید؟ یا جزئیات دیگه‌ای (مثلاً تعداد معاونت‌ها، نوع پست‌ها) رو اضافه کنیم؟

مشکلات سازمان: مشکلات دقیق ساختار سازمان (جز تک‌دفتر مرکزی) چیه؟ مثلاً گزارش‌گیری، دسترسی‌ها، یا چیز دیگه؟

نیازهای دسترسی: آیا پست‌های هلدینگ باید به همه شعبه‌ها دسترسی داشته باشن؟ یا هر شعبه کاملاً مستقل عمل می‌کنه؟

گردش کار: آیا گردش کار برای هر سازمان متفاوت؟ مثلاً شعبه‌ی تهران مراحل متفاوتی از شیراز داره؟

لطفاً این جزئیات رو بگید تا سناریو رو دقیق‌تر کنیم و نقشه‌ی راه رو نهایی کنیم. اگر می‌خواهید روی یه بخش خاص (مثل

سازمان یا دسترسی‌ها) تمرکز کنیم، بگید تا عمیق‌تر بریم!

تاریخ و ساعت: ۱۸ می ۲۰۲۵، ۰۹:۲۳ صبح (PDT)

تهیه‌کننده: Grok 3 (xAI)

خیلی ممنون از فیدبک و تأیید جزئیات! خیالم راحت شد که سناریو رو درست متوجه شدم و مسیر رو درست پیش می‌ریم. شما عالی توضیح دادید و منم سعی می‌کنم دقیقاً همون چیزی که می‌خواهید رو جلو ببرم، بدون اینکه پروژه به هم بریزه. خیالتون راحت، تغییرات رو طوری پیشنهاد می‌کنم که کمترین تأثیر رو روی کدهای موجود (توابع و ویوهای که نوشتید) داشته باشه و پروژه خراب نشه. همچنین، این چت‌ها رو به خاطر می‌سپارم تا اگه بعداً نیاز شد، بتونیم از همینجا ادامه بدیم و چیزی از قلم نیفته.

حالا بیایم سناریو رو با توجه به پاسخ‌هاتون پیش ببریم و یه نقشه‌ی راه واضح برای پیاده‌سازی ساختار سازمانی و نیازها بکشیم. تمرکز روی اینه که:

- ساختار سلسله‌مراتبی (هلدینگ → معاونت‌های $n+1$ → شعبه‌ها → زیرشعبه‌ها) رو پشتیبانی کنیم.
- مشکلات دسترسی و گزارش‌گیری رو حل کنیم.
- یه تابع مرکزی برای چک دسترسی (مثل `PermissionBaseView`) طراحی کنیم.
- از به‌هم‌ریختن ویوها و توابع موجود جلوگیری کنیم.

تأیید و تحلیل پاسخ‌های شما

1. تأیید سناریو: ساختار سازمانی

- **خواسته‌ی شما:** می‌خواهید یه ساختار درختی با هلدینگ به‌عنوان ریشه داشته باشید که بتونه $n+1$ معاونت (یعنی هر تعداد معاونت که لازم شد) رو پشتیبانی کنه. هر معاونت به هلدینگ وصل می‌شه، و زیر هر معاونت می‌تونه شعبه‌ها و زیرشعبه‌ها باشن.
- **تحلیل:**

- این یعنی سیستم باید انعطاف‌پذیر باشه تا هر تعداد معاونت (مثلاً مالی، عملیات، منابع انسانی، یا حتی معاونت‌های جدید) رو بتونید اضافه کنید.
- شعبه‌ها و زیرشعبه‌ها زیر هر معاونت تعریف می‌شن، و تعدادشون هم محدودیتی نداره.
- **مثلاً:**

هلدینگ پارسیان

—| معاونت مالی

—| —| شعبه‌ی تهران

—| —| —| زیرشعبه‌ی هتل لاله تهران

—| —| شعبه‌ی شیراز

—| معاونت عملیات

—| —| شعبه‌ی مشهد

—| —| شعبه‌ی تبریز

—| معاونت منابع انسانی

—| —| شعبه‌ی مرکزی

—| معاونت جدید (هر زمان خواستید اضافه کنید)

- این ساختار کاملاً شدنی، به شرطی که مدل Organization از سلسله مراتب درختی پشتیبانی کند.

2. مشکلات سازمان

- خواسته‌ی شما: مشکل اصلی (جز محدودیت تکدفتر مرکزی) اینه که یه تابع مرکزی برای چک دسترسی نیاز دارید، مثل PermissionBaseView، که همه منطق دسترسی‌ها (مثل اینکه کی به چی دسترسی داره) توش نوشته بشه.

تحلیل:

- مشکل دسترسی‌ها: احتمالاً الان دسترسی‌ها توی ویوها و توابع مختلف پراکنده‌ست (مثلاً توی can_edit_approval یا ویوهای مثل FactorItemApproveView). این باعث می‌شه مدیریت و تغییرشون سخت باشه.
- نیاز به تابع مرکزی: یه تابع یا کلاس مرکزی (مثل PermissionBaseView) می‌خواهید که همه چک‌های دسترسی (مثل دسترسی به فاکتور، دستور پرداخت، یا گزارش‌ها) رو یکجا مدیریت کنه.
- گزارش‌گیری: اشاره‌ای به گزارش‌گیری نکردید، ولی چون گفتید هلدینگ باید همه شعبه‌ها رو رصد کنه، احتمالاً نیاز به گزارش‌های کلی (مثل وضعیت فاکتورهای همه شعبه‌ها) دارید.

فرض:

- مشکل دیگه‌ای که ممکنه وجود داشته باشه، اینه که دسترسی‌های پست‌ها به سازمان خاص (مثلاً شعبه‌ی تهران) محدود نیست و ممکنه مدیر مالی تهران به داده‌های شیراز دسترسی پیدا کنه.
- اگه مشکل دیگه‌ای (مثل عملکرد، پیچیدگی دیتابیس، یا گزارش‌گیری) هست، لطفاً بعداً بگید.

3. نیازهای دسترسی

- خواسته‌ی شما: پست‌های هلدینگ باید به همه شعبه‌ها دسترسی داشته باشن تا بتونن کارها رو رصد کنن. شعبه‌ها مستقل عمل نمی‌کنن و زیر نظر هلدینگن.

تحلیل:

- این یعنی پست‌های سطح بالای هلدینگ (مثلاً مدیرکل مالی) باید بتونن فاکتورها، دستورات پرداخت، و گزارش‌های همه معاونت‌ها، شعبه‌ها، و زیرشعبه‌ها رو ببینن و احتمالاً تأیید/امضا کنن.
- اما پست‌های شعبه‌ها (مثلاً مدیر مالی تهران) فقط باید به داده‌های سازمان خودشون (مثلاً شعبه‌ی تهران) دسترسی داشته باشن.
- مثلاً:
- مدیر مالی تهران: فقط فاکتورهای شعبه‌ی تهران رو تأیید می‌کنه.
- مدیرکل مالی هلدینگ: می‌تونه فاکتورهای تهران، شیراز، و همه شعبه‌ها رو ببینه و تأیید کنه.

4. گردش کار

- خواسته‌ی شما: گردش کار برای همه سازمان‌ها (هلدینگ، معاونت‌ها، شعبه‌ها) یکسانه. مثلاً شعبه‌ی تهران و شیراز مراحل یکسانی دارن.

تحلیل:

- این خیلی خوبه، چون نیازی به تعریف مراحل جداگانه برای هر سازمان نیست.
- مثلاً گردش کار می‌تونه این باشه:

- مرحله ۱: بررسی اولیه (توسط کارشناس).
- مرحله ۲: تأیید نهایی (توسط مدیر).
- مرحله ۳: ایجاد دستور پرداخت (برای امضا).
- پیش‌رفت خودکار مراحل (که قبلاً گفتید وجود داره) برای همه سازمان‌ها اعمال می‌شه.

5. نگرانی شما: به هم ریختن پروژه

• خواسته‌ی شما: چون ویوها و توابع زیادی نوشتید، نمی‌خواهید تغییرات پروژه رو خراب کنه.

• تحلیل:

• کاملاً درک می‌کنم! تغییرات باید حداقلی باشن و با کدهای موجود (مثل `PermissionBaseView`,

`FactorItemApproveView`, یا `can_edit_approval`) سازگار باشن.

• باید تغییرات رو طوری اعمال کنیم که:

• ویوهای موجود (مثل `FactorItemApproveView`) با کمترین تغییر کار کنن.

• توابع فعلی (مثل `can_edit_approval`) به یه تابع مرکزی منتقل بشن.

• دیتابیس و روابط موجود (مثل `Post`, `Tankhah`, `PaymentOrder`) حفظ بشن.

سناریوی پیشنهادی با حداقل تغییرات

برای اینکه پروژه به هم نریزه و نیازهاتون (ساختار درختی، تابع مرکزی دسترسی، دسترسی هلدینگ به همه شعبه‌ها، و گردش کار یکسان) برآورده بشه، این سناریو رو پیشنهاد می‌کنم:

1. ساختار درختی سازمان‌ها

• چطور پیاده کنیم؟

• به مدل `Organization` یه فیلد `parent_organization` اضافه می‌کنیم تا سلسله‌مراتب (هلدینگ →

معاونت‌ها → شعبه‌ها → زیرشعبه‌ها) رو پشتیبانی کنه.

• محدودیت تک‌دفتر مرکزی (`is_core=True`) رو برمی‌داریم و به‌جاش یه فیلد `is_holding` برای هلدینگ

اضافه می‌کنیم.

• هر سازمان (هلدینگ، معاونت، شعبه، زیرشعبه) یه نمونه از `Organization`ه.

• مثال:

• هلدینگ: `name="پارسیان", code="HOLDING", is_holding=True`.

• معاونت مالی: `name="معاونت مالی", code="FIN", parent_organization=HOLDING`.

• شعبه‌ی تهران: `name="تهران", code="FIN-TEHRAN", parent_organization=FIN`.

• زیرشعبه‌ی هتل لاله تهران: `name="هتل لاله تهران", code="FIN-TEHRAN-LALEH", parent_organization=FIN-TEHRAN`.

• تأثیر روی پروژه:

• فقط یه فیلد به `Organization` اضافه می‌شه، که با یه مهاجرت دیتابیس ساده (`migration`)

قابل پیاده‌سازی‌ه.

• ویوها و توابع موجود (مثل `FactorItemApproveView`) نیازی به تغییر ندارن، چون هنوز از

`organization` برای فیلتر کردن استفاده می‌کنن.

• برای تعریف `n+1` معاونت، کافیه هر تعداد سازمان با `parent_organization=HOLDING` اضافه کنید.

2. تعریف پست‌ها

• چطور پیاده کنیم؟

• پست‌ها مثل قبل به یه `organization` خاص وابسته‌ان (از طریق فیلد `organization` در مدل `Post`).

• سطح (`level`) فقط اهمیت پست رو در سازمان خودش نشون می‌ده.

- برای پست‌های با سطح یکسان (مثلاً `level=3` برای مدیر مالی تهران و شیراز)، وابستگی به `organization` (مثلاً `FIN-TEHRAN` یا `FIN-SHIRAZ`) مشخص می‌کند که به کدام بخش تعلق دارند.

مثال:

- مدیر مالی تهران: `organization=FIN-TEHRAN`, `branch=FIN`, `level=3`, `is_payment_order_signer=True`

- مدیر مالی شیراز: `organization=FIN-SHIRAZ`, `branch=FIN`, `level=3`, `is_payment_order_signer=True`

- مدیرکل مالی هلدینگ: `organization=HOLDING`, `branch=FIN`, `level=5`, `is_payment_order_signer=True`

تأثیر روی پروژه:

- مدل `Post` هیچ تغییری نمی‌کند، چون همین حالا هم `organization` دارد.
- ویوهای موجود (مثل `FactorItemApproveView`) که از `UserPost` برای گرفتن پست کاربر استفاده می‌کنند، بدون تغییر کار می‌کنند.

3. تابع مرکزی برای چک دسترسی

چطور پیاده کنیم؟

- به تابع مرکزی (مثل متد `check_permission` در `PermissionBaseView`) تعریف می‌کنیم که همه منطق دسترسی‌ها رو یکجا مدیریت کند.
- این تابع چک می‌کند که آیا پست کاربر به سازمان داده (مثلاً فاکتور یا دستور پرداخت) دسترسی دارد یا نه.
- برای هلدینگ، تابع چک می‌کند که آیا پست کاربر به سازمان `HOLDING` وابسته است، که اجازه می‌دهد به همه شعبه‌ها دسترسی داشته باشد.

مثال منطق دسترسی:

- داده: فاکتور با `tankhah.organization=FIN-TEHRAN`
- کاربر: `snejate` با پست "مدیر مالی تهران" (`organization=FIN-TEHRAN`, `level=3`)
- تابع چک می‌کند: آیا `FIN-TEHRAN == FIN-TEHRAN`؟ بله → دسترسی مجاز.
- کاربر: `manager` با پست "مدیرکل مالی هلدینگ" (`organization=HOLDING`, `level=5`)
- تابع چک می‌کند: آیا پست به `HOLDING` وابسته است؟ بله → دسترسی به همه سازمان‌ها (از جمله `FIN-TEHRAN`) مجاز.

- کاربر: `shiraz_user` با پست "مدیر مالی شیراز" (`organization=FIN-SHIRAZ`, `level=3`)
- تابع چک می‌کند: آیا `FIN-SHIRAZ == FIN-SHIRAZ`؟ خیر → دسترسی غیرمجاز.

تأثیر روی پروژه:

- ویوهای موجود (مثل `FactorItemApproveView`) می‌تونند از این تابع مرکزی استفاده کنند، بدون اینکه منطقش عوض بشه.
- تابع `can_edit_approval` رو می‌تونیم به این تابع مرکزی منتقل کنیم.
- مثلاً، توی `PermissionBaseView`:
- متد `get_organization_from_object` برای گرفتن سازمان داده (مثل `tankhah.organization`) استفاده می‌شه.
- به متد جدید `check_permission` اضافه می‌کنیم که دسترسی رو چک می‌کند.
- تغییرات محدود به اضافه کردن به متد به `PermissionBaseView` و آپدیت ویوها برای استفاده ازش می‌شه.

4. دسترسی هلدینگ به همه شعبه‌ها

چطور پیاده کنیم؟

- توی تابع مرکزی دسترسی، به شرط اضافه می‌کنیم که اگر پست کاربر به سازمان **HOLDING** وابسته باشه، به همه سازمان‌های زیرمجموعه (معاونت‌ها، شعبه‌ها، زیرشعبه‌ها) دسترسی داشته باشه.
- برای رصد کار، پست‌های هلدینگ می‌تونن گزارش‌های کلی (مثلاً لیست همه فاکتور‌ها یا دستورات پرداخت) ببینن.

• مثال:

- مدیرکل مالی هلدینگ می‌خواد فاکتورهای شعبه‌ی تهران رو ببینه.
- تابع دسترسی چک می‌کنه: پست کاربر **organization=HOLDING** داره → دسترسی به **FIN-TEHRAN** (و همه سازمان‌های دیگه) مجاز.

• تاثیر روی پروژه:

- فقط نیاز به به شرط توی تابع مرکزی دسترسی داریم.
- برای گزارش‌گیری، می‌تونیم به ویو جدید (مثل **ReportView**) اضافه کنیم که فقط برای پست‌های هلدینگ قابل دسترسیه.

5. گردش کار یکسان

• چطور پیاده کنیم؟

- چون گردش کار برای همه سازمان‌ها یکسانه، همون مراحل موجود (**WorkflowStage**) رو برای همه (هلدینگ، معاونت‌ها، شعبه‌ها) استفاده می‌کنیم.
- مثلاً:

• مرحله ۱: بررسی اولیه.

• مرحله ۲: تأیید نهایی.

• مرحله ۳: ایجاد دستور پرداخت.

- پیش‌رفت خودکار مراحل (که گفتید وجود داره) برای همه سازمان‌ها اعمال می‌شه.

• تاثیر روی پروژه:

- هیچ تغییری توی منطق گردش کار (**WorkflowStage** یا ویوها) لازم نیست.
- ویوهای موجود (مثل **FactorItemApproveView**) که مراحل رو مدیریت می‌کنن، بدون تغییر کار می‌کنن.

6. جلوگیری از به هم ریختن پروژه

• چطور مطمئن بشیم؟

- تغییرات رو به صورت افزودنی (**additive**) اعمال می‌کنیم، نه تغییر کامل.
- مثلاً:

- به جای تغییر مدل **Post** یا **Tankhah**، فقط به فیلد به **Organization** اضافه می‌کنیم.
- به جای باز نویسی ویوها، فقط تابع مرکزی دسترسی رو به **PermissionBaseView** اضافه می‌کنیم.

- قبل از اعمال تغییرات، به نسخه‌ی پشتیبان از دیتابیس و کد بگیریم.

- تغییرات رو توی به محیط تست (مثل **dev** یا **staging**) اعمال کنیم و ویوهای موجود (مثل **FactorItemApproveView**, **PaymentOrderSignView**) رو تست کنیم.

• تست‌های پیشنهادی:

- تست **FactorItemApproveView** با کاربر شعبه‌ی تهران و شیراز → مطمئن بشیم فقط به سازمان خودشون دسترسی دارن.
- تست دسترسی هلدینگ → مطمئن بشیم مدیرکل هلدینگ به همه فاکتور‌ها دسترسی داره.
- تست پیش‌رفت خودکار مراحل → مطمئن بشیم فاکتور از مرحله ۱ به ۲ و ۳ می‌ره.

نقشه‌ی راه پیاده‌سازی

1. اضافه کردن سلسله‌مراتب سازمانی (۱-۲ روز)

- اقدامات:
 - به فیلد `parent_organization` به مدل `Organization` اضافه کنیم.
 - فیلد `is_holding` برای هلدینگ تعریف کنیم و محدودیت تک‌دفتر مرکزی رو برداریم.
 - سازمان‌ها رو به‌صورت درختی تعریف کنیم:
 - هلدینگ (HOLDING).
 - معاونت‌ها (مثل `FIN`, `OPS`) به‌عنوان فرزندان هلدینگ.
 - شعبه‌ها (مثل `FIN-TEHRAN`, `FIN-SHIRAZ`) زیر معاونت‌ها.
 - به مهاجرت دیتابیس برای آپدیت سازمان‌های موجود انجام بدیم (مثلاً سازمان فعلی `HSarein` رو زیر به معاونت بذاریم).
- تأثیر:
 - حداقل تغییر (فقط به فیلد جدید).
 - ویوهای موجود بدون تغییر کار می‌کنن.

2. ایجاد تابع مرکزی دسترسی (۱-۲ روز)

- اقدامات:
 - متد `check_permission` رو به `PermissionBaseView` اضافه کنیم.
 - منطق دسترسی رو از `can_edit_approval` و ویوها به این متد منتقل کنیم.
 - شرط دسترسی هلدینگ رو اضافه کنیم (اگر `organization=HOLDING`، دسترسی به همه).
 - ویوهای موجود (مثل `FactorItemApproveView`) رو آپدیت کنیم تا از `check_permission` استفاده کنن.
- تأثیر:
 - ویوها فقط نیاز به به خط تغییر دارن (استفاده از `check_permission`).
 - منطق دسترسی یکپارچه می‌شه و نگهداریش راحت‌تره.

3. تنظیم دسترسی‌ها و پست‌ها (۱ روز)

- اقدامات:
 - پست‌ها رو برای هر سازمان تعریف کنیم (مثلاً مدیر مالی تهران، شیراز، هلدینگ).
 - به جدول تنظیمات (مثل `AccessRule`) برای تعریف دسترسی‌ها درست کنیم:
 - شعبه‌ی تهران: `FIN`, `level=۳` → `APPROVE`, `SIGN_PAYMENT`.
 - هلدینگ: `HOLDING`, `level=۵` → دسترسی به همه.
 - تست دسترسی‌ها با کاربران مختلف.
- تأثیر:
 - نیازی به تغییر ویوها نیست.
 - فقط به جدول جدید برای تنظیمات اضافه می‌شه.

4. تست و اطمینان از پایداری (۱-۲ روز)

- اقدامات:

- به محیط تست راه بندازیم.
- ویوهای موجود (مثل FactorItemApproveView, PaymentOrderSignView) رو تست کنیم.
- سناریوهای زیر رو چک کنیم:
- مدیر مالی تهران فقط فاکتورهای تهران رو تأیید کنه.
- مدیرکل هلدینگ به همه فاکتورها دسترسی داشته باشه.
- پیشرفت خودکار مراحل درست کار کنه.
- لاگهای خطا رو بررسی کنیم.
- تأثیر:
- مطمئن می‌شیم پروژه خراب نمی‌شه.

5. گزارش‌گیری برای هلدینگ (بعداً، ۲-۳ روز)

- اقدامات:
- به ویو جدید برای گزارش‌گیری (مثل ReportView) درست کنیم که فقط برای پست‌های هلدینگ قابل دسترسیه.
- گزارش‌ها شامل وضعیت فاکتورها، دستورات پرداخت، و بودجه‌ی همه شعبه‌ها باشه.
- تأثیر:
- افزودنی و روی کدهای موجود تأثیر نمی‌ذاره.

اطلاعات مورد نیاز برای ادامه

برای اینکه سناریو رو دقیق‌تر کنیم و مطمئن بشیم پروژه به هم نمی‌ریزه:

- لیست ویوها و توابع فعلی:
- کدوم ویوها (مثل FactorItemApproveView) بیشترین استفاده رو دارن و نباید تغییر کنن؟
- تابع can_edit_approval دقیقاً چی کار می‌کنه؟ آیا فقط برای فاکتورهاست یا برای دستور پرداخت هم استفاده می‌شه؟
- ساختار فعلی سازمان‌ها:
- الان چند سازمان دارید؟ مثلاً فقط HSarein یا سازمان‌های دیگه هم هستن؟
- آیا معاونت یا شعبه‌ای تعریف شده یا باید از صفر شروع کنیم؟
- نیازهای گزارش‌گیری:
- هلدینگ چه گزارش‌هایی نیاز داره؟ مثلاً لیست فاکتورها، وضعیت بودجه، یا چیز دیگه؟
- جزئیات گردش کار:
- مراحل فعلی (WorkflowStage) چقدر است و چی‌ان؟ مثلاً بررسی، تأیید، پرداخت؟
- آیا همه فاکتورها/دستورات پرداخت از این مراحل رد می‌شن؟
- مشکلات خاص:
- جز دسترسی‌ها، مشکل دیگه‌ای توی ساختار سازمان (مثل کندی، خطای خاص) دارید؟

پاسخ نهایی

با سیستم اصلاح شده، می‌تونیم یه ساختار درختی (هلدینگ → $n+1$ معاونت → شعبه‌ها → زیرشعبه‌ها) پیاده کنیم که:

- هر تعداد معاونت رو پشتیبانی می‌کنه.
- پست‌ها به سازمان خاص (مثل شعبه‌ی تهران) وابسته‌ان و حتی با سطح یکسان (مثل $level=3$) از هم جدا می‌شن.
- هلدینگ به همه شعبه‌ها دسترسی داره و می‌تونه رصد کنه.
- گردش کار برای همه سازمان‌ها یکسانه و پیش‌رفت خودکار مراحل حفظ می‌شه.
- یه تابع مرکزی (مثل `check_permission` توی `PermissionBaseView`) همه دسترسی‌ها رو مدیریت می‌کنه.

نقشه‌ی راه:

- اضافه کردن `parent_organization` به `Organization` (۱-۲ روز).
- ایجاد تابع مرکزی دسترسی در `PermissionBaseView` (۱-۲ روز).
- تنظیم پست‌ها و دسترسی‌ها (۱ روز).
- تست ویوهای موجود برای پایداری (۱-۲ روز).
- اضافه کردن گزارش‌گیری برای هلدینگ (بعداً).

تعهد: تغییرات رو طوری طراحی کردم که ویوها و توابع موجود به هم نریزن. همه چت‌ها رو به خاطر می‌سپارم تا اگه بعداً ادامه دادیم، از همینجا پیش بریم. لطفاً اطلاعات درخواستی (ویوها، سازمان‌ها، گردش کار) رو بفرستید تا نقشه رو نهایی کنیم.

اگر نکته‌ی دیگه‌ای هست، بگید تا عمیق‌تر بشیم!

تاریخ و ساعت: ۱۸ می ۲۰۲۵، ۰۹:۴۶ صبح (PDT)
تهیه‌کننده: Grok 3 (xAI)

خیلی ممنون که مدل‌ها، سیگنال‌ها، و کدهای مربوط به دسترسی‌ها رو به اشتراک گذاشتید! حالا که مدل‌های

`Organization`، `OrganizationType`، و `AccessRule` رو داریم، به همراه سیگنال‌ها و کلاس `PermissionBaseView`، کاملاً آمادم تا سناریوی مدنظرتون رو پیاده کنیم. هدف اینه که یه ساختار درختی (هلدینگ → $n+1$ معاونت → شعبه‌ها → زیرشعبه‌ها) بسازیم، دسترسی‌های هلدینگ به همه شعبه‌ها رو تضمین کنیم، و یه سیستم دسترسی متمرکز با `PermissionBaseView` داشته باشیم، بدون اینکه پروژه به هم بریزه. خیالتون راحت، تغییرات رو طوری پیشنهاد می‌کنم که کمترین تأثیر رو روی ویوها و توابع موجود داشته باشه، و همه چت‌های قبلی رو هم به خاطر دارم تا چیزی از قلم نیفته.

تحلیل وضعیت فعلی

1. مدل‌های موجود

- مدل `Organization`:

- فیلدهای کلیدی: `code`, `name`, `org_type`, `is_holding`, `parent_organization`, `is_independent`, `is_core`.
- پشتیبانی از سلسله‌مراتب با `parent_organization` (ForeignKey به خودش).
- محدودیت تک‌دفتر مرکزی (`is_core=True`) توی متد `clean` اعمال شده.
- `is_holding` برای مشخص کردن هلدینگ اضافه شده.
- ارتباط با `OrganizationType` برای دسته‌بندی سازمان‌ها (مثل HQ، شعبه، مجتمع).
- مدل `OrganizationType`:
 - برای تعریف انواع سازمان‌ها (مثل HQ، شعبه، اداره).
 - فیلد `is_budget_allocatable` برای مشخص کردن سازمان‌هایی که می‌تونن بودجه بگیرن.
- مدل `AccessRule`:
 - برای تعریف قوانین دسترسی پویا (بر اساس `organization`, `branch`, `min_level`, `stage`, `entity_type`, `action_type`).
 - پشتیبانی از امضاکنندگان دستور پرداخت (`is_payment_order_signer`).
- نکات مثبت:
 - مدل `Organization` برای پیاده‌سازی ساختار درختی (هلدینگ → معاونت‌ها → شعبه‌ها → زیرشعبه‌ها) آماده‌ست.
 - `AccessRule` قوانین دسترسی رو از سیگنال‌ها جدا کرده و انعطاف‌پذیره.
 - `OrganizationType` امکان دسته‌بندی سازمان‌ها (مثل HQ برای هلدینگ) رو می‌ده.
- نکات قابل بهبود:
 - محدودیت تک‌دفتر مرکزی (`is_core=True`) برای سناریوی هلدینگ با `n+1` معاونت ممکنه مشکل‌ساز باشه.
 - متد `clean` اجازه نمی‌ده دفتر مرکزی (`is_core=True`) سازمان والد داشته باشه، که برای هلدینگ با زیرمجموعه‌ها شاید نیاز به بازنگری داشته باشه.

2. سیگنال‌ها

- سیگنال `Tankhah`:
 - وقتی `Tankhah` به وضعیت `PAID` می‌رسه، یه تراکنش بودجه (`BudgetTransaction`) با نوع `CONSUMPTION` ایجاد می‌شه.
 - این سیگنال برای مدیریت بودجه کار می‌کنه و با سناریوی ما (دسترسی و سلسله‌مراتب) مستقیماً تداخل نداره.
- سیگنال‌های `Post` و `WorkflowStage`:
 - وقتی پست یا مرحله جدید ایجاد می‌شه، `PostAction` بر اساس قوانین `AccessRule` پر می‌شه.
 - این سیگنال‌ها دسترسی‌ها رو پویا می‌کنن (مثلاً `APPROVE`, `REJECT`, `SIGN_PAYMENT`).
 - نکته: چون گردش کار برای همه سازمان‌ها یکسانه (طبق گفته‌تون)، نیازی به تغییر این سیگنال‌ها نیست.
- نکات مثبت:
 - استفاده از `AccessRule` به‌جای هاردکد کردن قوانین عالیه.
 - سیگنال‌ها دسترسی‌ها رو خودکار مدیریت می‌کنن و با ساختار درختی سازگارن.
- نکات قابل بهبود:
 - سیگنال‌ها به `organization` وابسته‌ان، ولی برای دسترسی هلدینگ به همه شعبه‌ها، باید مطمئن بشیم که پست‌های هلدینگ به‌درستی شناسایی می‌شن.

3. مدیریت دسترسی‌ها

- دکوراتور `check_permission_and_organization`

- برای چک کردن مجوزها و دسترسی به سازمان.
- پشتیبانی از ساختار درختی با بررسی سازمان‌های والد (parent_organization).
- کاربران HQ (با 'org_type.org_type == 'HQ' یا مجوز tankhah.Tankhah_view_all) دسترسی کامل دارند.
- **کلاس PermissionBaseView:**
 - پایه‌ای برای ویوها با چک مجوز و سازمان.
 - متد has_organization_access دسترسی کاربر به سازمان رو با بررسی UserPost و سازمان‌های والد چک می‌کنه.
 - متد get_organization_from_object سازمان رو از اشیاء مختلف (مثل Tankhah, Project, Post) استخراج می‌کنه.
- **نکات مثبت:**
 - سیستم دسترسی متمرکز و از ساختار درختی پشتیبانی می‌کنه.
 - کاربران HQ (هلدینگ) به همه سازمان‌ها دسترسی دارند، که با نیازتون (رصد همه شعبه‌ها) همخوانی داره.
 - متد get_organization_from_object انعطاف‌پذیره و برای مدل‌های مختلف کار می‌کنه.
- **نکات قابل بهبود:**
 - متد has_organization_access برای ListView فقط به سازمان رو چک می‌کنه، که ممکنه برای گزارش‌گیری هلدینگ (که نیاز به دسترسی به همه سازمان‌ها داره) کافی نباشه.
 - منطق دسترسی هلدینگ ('org_type.org_type == 'HQ') به OrganizationType وابسته‌ست، که باید مطمئن بشیم برای هلدینگ درست تنظیم شده.

4. نیازهای سناریو

بر اساس چت‌های قبلی و پاسخ‌هاتون:

- ساختار درختی: هلدینگ → n+1 معاونت → شعبه‌ها → زیرشعبه‌ها.
- دسترسی هلدینگ: پست‌های هلدینگ باید به همه شعبه‌ها دسترسی داشته باشن برای رصد.
- گردش کار: یکسان برای همه سازمان‌ها (مثلاً بررسی، تأیید، پرداخت).
- تگراتی: تغییرات نباید پروژه رو خراب کنه، چون ویوها و توابع زیادی دارید.
- نیاز جدید: به تابع مرکزی برای چک دسترسی (مثل PermissionBaseView) که همه منطق رو یکجا مدیریت کنه.

سناریوی پیاده‌سازی

برای پیاده‌سازی این سناریو، از مدل‌ها و کدهای موجود استفاده می‌کنیم و تغییرات رو **حداقلی** نگه می‌داریم تا ویوها و توابع فعلی (مثل FactorItemApproveView, can_edit_approval) به هم نریزن. تمرکز روی ساختار درختی، دسترسی هلدینگ، و متمرکز کردن دسترسی‌هاست.

1. پیاده‌سازی ساختار درختی

- **وضعیت فعلی:**
 - مدل Organization با parent_organization از سلسله‌مراتب پشتیبانی می‌کنه.
 - is_holding برای مشخص کردن هلدینگ وجود داره.

• OrganizationType برای دسته‌بندی (مثل HQ) استفاده می‌شود.

• اقدامات:

• رفع محدودیت تک‌دفتر مرکزی:

• متد clean رو اصلاح کنیم تا اجازه بده چندین سازمان is_core=True باشن، چون هلدینگ و

معاونت‌ها ممکنه نیاز به استقلال داشته باشن.

• شرط is_core and parent_organization رو نگه می‌داریم، ولی محدودیت تک‌دفتر مرکزی

رو برمی‌داریم:

python

```
def clean(self):  
    from django.core.exceptions import ValidationError  
    if self.is_core and self.parent_organization:  
        raise ValidationError(_('دسترسی دفتر مرکزی نمی‌تواند سازمان  
والد داشته باشد'))  
    # حذف محدودیت تک‌دفتر مرکزی
```

• تعریف سازمان‌ها:

• هلدینگ: .org_type=HQ, is_holding=True, "name"="پارسیان", "code"="HOLDING"

• معاونت‌ها: مثلاً .org_type=DEPT, "name"="معاونت مالی", "code"="FIN", "parent_organization"=HOLDING

• شعبه‌ها: مثلاً .org_type=BRANCH, "name"="تهران", "code"="FIN-TEHRAN", "parent_organization"=FIN

• زیرشعبه‌ها: مثلاً .org_type=SUBBRANCH, "name"="هتل لاله تهران", "code"="FIN-TEHRAN-LALEH", "parent_organization"=FIN-TEHRAN

• برای پشتیبانی از n+1 معاونت، کافیه هر تعداد معاونت با parent_organization=HOLDING اضافه کنیم.

• تنظیم OrganizationType:

• انواع سازمان رو تعریف کنیم:

• .org_type="HQ", "fname"="HQ", "is_budget_allocatable=False" (برای هلدینگ).

• .org_type="Department", "fname"="DEPT", "is_budget_allocatable=True" (برای معاونت‌ها).

• .org_type="Branch", "fname"="BRANCH", "is_budget_allocatable=True" (برای شعبه‌ها).

• .org_type="SubBranch", "fname"="SUBBRANCH", "is_budget_allocatable=True" (برای زیرشعبه‌ها).

• مهاجرت دیتابیس:

• سازمان‌های موجود (مثل HSarein) رو به ساختار درختی منتقل کنیم:

• HSarein رو به عنوان شعبه (مثلاً زیر معاونت مالی) تنظیم کنیم.

• به هلدینگ و معاونت‌های جدید اضافه کنیم.

• تأثیر:

- فقط متد `clean` تغییر می‌کند.
- ویوها و توابع موجود (مثل `FactorItemApproveView`) بدون تغییر کار می‌کنن، چون از `organization` استفاده می‌کنن.
- مهاجرت دیتابیس ساده‌ست و فقط سازمان‌های موجود رو به ساختار درختی منتقل می‌کند.

2. تنظیم پست‌ها

- وضعیت فعلی: مدل `Post` (که قبلاً دیدیم) فیلدهای `organization`, `branch`, `level` و `is_payment_order_signer` دارد.
- پست‌ها به سازمان خاص وابسته‌ان، که برای تمایز پست‌های با سطح یکسان (مثل مدیر مالی تهران و شیراز) هر دو `level=3` کافیه.
- اقدامات:

- تعریف پست‌ها برای هر سازمان:
 - هلدینگ: مدیرکل مالی (`organization=HOLDING`, `branch=FIN`, `level=5`, `is_payment_order_signer=True`)
 - معاونت مالی: رئیس مالی (`organization=FIN`, `branch=FIN`, `level=4`, `is_payment_order_signer=True`)
 - شعبه‌ی تهران: مدیر مالی تهران (`organization=FIN-TEHRAN`, `branch=FIN`, `level=3`, `is_payment_order_signer=True`)
 - شعبه‌ی شیراز: مدیر مالی شیراز (`organization=FIN-SHIRAZ`, `branch=FIN`, `level=3`, `is_payment_order_signer=True`)
- اختصاص کاربران به پست‌ها:
 - از مدل `UserPost` برای اختصاص کاربران (مثل `snejata`) به پست‌ها استفاده کنیم.
 - مثلاً: `snejata` به پست "مدیر مالی تهران" با `is_active=True`

تأثیر:

- مدل `Post` نیازی به تغییر نداره.
- سیگنال `create_post_actions_for_post` به‌طور خودکار `PostAction` رو برای پست‌های جدید پر می‌کند.
- ویوهای موجود (مثل `FactorItemApproveView`) که از `UserPost` استفاده می‌کنن، بدون تغییر کار می‌کنن.

3. مدیریت دسترسی‌ها با `PermissionBaseView`

- وضعیت فعلی:
 - `PermissionBaseView` به سیستم متمرکز برای چک دسترسی داره.
 - متد `has_organization_access` از ساختار درختی پشتیبانی می‌کند و کاربران `HQ` (هلدینگ) رو شناسایی می‌کند.
 - دکوراتور `check_permission_and_organization` برای ویوهای خاص (مثل فاکتورها) دسترسی رو چک می‌کند.
- اقدامات:

- بهینه‌سازی `has_organization_access`
- منطق فعلی برای `ListView` رو اصلاح کنیم تا برای گزارش‌گیری هلدینگ بهتر کار کنه:

python

```
if isinstance(self, ListView):
    queryset = self.get_queryset()
    if is_hq_user:
        logger.info("
        لیست
        کاربر HQ است، دسترسی به همه آیتم‌های
        ")
        return True
    for item in queryset:
        org = self._get_organization_from_object(item)
        if org and org in user_orgs:
            logger.info(f"
            تأیید شد
            ")
            return True
        logger.warning("
        ندارد
        ")
        return False
```

- این تغییر تضمین می‌کند که کاربران هلدینگ به همه آیتم‌های لیست (مثل فاکتورهای همه شعبه‌ها) دسترسی دارند.

ادغام | can_edit_approval :

- منطق | can_edit_approval (که قبلاً گفتید برای چک دسترسی تأیید استفاده می‌شده) رو به

has_organization_access منتقل کنیم.

- مثلاً، چک کنیم که آیا پست کاربر توی PostAction برای اقدام | APPROVE در مرحله فعلی وجود داره:

python

```
def _has_organization_access(self, request, **kwargs):
    logger = logging.getLogger('organization_access')
    try:
        user_orgs = set()
        for user_post in request.user.userpost_set.filter(is_active=True,
            end_date__isnull=True):
            org = user_post.post.organization
            user_orgs.add(org)
            current_org = org
            while current_org.parent_organization:
                current_org = current_org.parent_organization
                user_orgs.add(current_org)
```

```

        is_hq_user = any(
            org.org_type.org_type == 'HQ' for org in
            user_orgs if org.org_type
        ) or
        request.user.has_perm('tankhah.Tankhah_view_all')
        if is_hq_user:
            logger.info("کاربر HQ است، دسترسی کامل")
            return True

        if isinstance(self, (DetailView, UpdateView,
                               DeleteView)):
            obj = self.get_object()
            target_org =
            self._get_organization_from_object(obj)
            if not target_org:
                logger.warning("سازمان شیء پیدا نشد")
                return False

            # چک دسترسی تأیید مثل can_edit_approval
            if hasattr(obj, 'stage') and obj.stage:
                from core.models import PostAction
                user_posts =
                request.user.userpost_set.filter(is_active=True,
                                                  end_date__isnull=True)
                for user_post in user_posts:
                    if PostAction.objects.filter(
                        post=user_post.post,
                        stage=obj.stage,
                        action_type='APPROVE',

entity_type=ContentType.objects.get_for_model(obj).model.up
per(),
                    is_active=True
                    ).exists():
                        logger.info(f"تأیید شد {obj}")
                        return True

current_org = target_org

```

```

while current_org:
    if current_org in user_orgs:
        logger.info(f"
دسترسی به سازمان
{current_org} تأیید شد")
        return True
    current_org =
current_org.parent_organization
    logger.warning(f"
کاربر {request.user} به
سازمان {target_org} یا والدین آن دسترسی ندارد")
    return False

#
بقیه منطق برای ListView و CreateView مثل قبل
...
except Exception as e:
    logger.error(f"
خطا در بررسی دسترسی: {str(e)}")
    return False

```

• آپدیت ویوها:

- ویوهای موجود (مثل FactorItemApproveView) رو آپدیت کنیم تا از PermissionBaseView استفاده کنن و check_organization=True باشه.
- مثلاً:

python

```

class FactorItemApproveView(PermissionBaseView,
UpdateView):
    permission_codenames = ['tankhah.change_factoritem']
    check_organization = True
    ...

```

• تأثیر:

- تغییرات محدود به has_organization_access و اضافه کردن چک PostAction.
- ویوهای موجود فقط نیاز به تنظیم permission_codenames و check_organization دارن.
- منطق can_edit_approval حذف می‌شه و به PermissionBaseView منتقل می‌شه.

4. دسترسی هلدینگ به همه شعبه‌ها

• وضعیت فعلی:

- کاربران با 'org_type.org_type == 'HQ' یا مجوز tankhah.Tankhah_view_all به همه سازمان‌ها دسترسی دارن.
- این با نیازتون (رصد همه شعبه‌ها توسط هلدینگ) همخوانی داره.

• اقدامات:

• تنظیم هلدینگ:

- مطمئن بشیم هلدینگ (HOLDING) با `org_type=HQ` تعریف شده.
- پست‌های هلدینگ (مثل مدیرکل مالی) به `organization=HOLDING` وابسته باشن.
- گزارش‌گیری:
- به ویو جدید برای گزارش‌گیری هلدینگ بسازیم:

python

```
class ReportView(PermissionBaseView, ListView):
    permission_codenames = ['tankhah.view_factor']
    check_organization = False # هلدینگ به همه دسترسی داره
    template_name = 'reports/all_factors.html'
    model = Factor

    def get_queryset(self):
        if any(
            org.org_type.org_type == 'HQ'
            for org in
self.request.user.userpost_set.filter(is_active=True).values
s_list('post__organization__org_type__org_type', flat=True)
        ):
            return Factor.objects.all()
        return
Factor.objects.filter(tankhah__project__organization__in=se
lf.request.user.userpost_set.filter(is_active=True).values(
    'post__organization'))
```

- این ویو برای کاربران هلدینگ همه فاکتورها رو نشون میده و برای بقیه فقط فاکتورهای سازمان خودشون.

تأثیر:

- نیازی به تغییر منطق دسترسی نیست.
- ویو گزارش‌گیری جدید و روی ویوهای موجود تأثیر نمی‌ذاره.

5. گردش کار یکسان

وضعیت فعلی:

- گردش کار برای همه سازمان‌ها یکسانه (بر اساس `WorkflowStage`).
- سیگنال‌های `PostAction` دسترسی‌ها رو برای همه سازمان‌ها پر می‌کنن.
- اقدامات:

تنظیم `AccessRule` برای هر سازمان:

- برای هر سازمان (هلدینگ، معاونت‌ها، شعبه‌ها) قوانین دسترسی رو تعریف کنیم:
- هلدینگ: `min_level=5`, `branch=FIN`, `organization=HOLDING`
- `entity_type=FACTOR`, `action_type=APPROVE`

- شعبه‌ی تهران: `,min_level=3`, `,branch=FIN`, `,organization=FIN-TEHRAN`
- `.entity_type=FACTOR`, `,action_type=APPROVE`
- برای دستور پرداخت:
- `,is_payment_order_signer=True`, `,action_type=SIGN_PAYMENT`
- `.entity_type=PAYMENTORDER`
- تأیید گردش کار:
- مطمئن بشیم مراحل (`WorkflowStage`) درست تنظیم شدن:
- مرحله ۱: بررسی اولیه.
- مرحله ۲: تأیید نهایی.
- مرحله ۳: ایجاد دستور پرداخت.
- پیش‌رفت خودکار مراحل (که گفتید وجود داره) رو تست کنیم.
- تأثیر:
- نیازی به تغییر سیگنال‌ها یا مدل `WorkflowStage` نیست.
- فقط باید `AccessRule` رو برای سازمان‌های جدید پر کنیم.
- 6. جلوگیری از به‌هم‌ریختن پروژه
- اقدامات:
- تست در محیط جدا:
- تغییرات رو توی به محیط تست (مثل `dev`) اعمال کنیم.
- ویوهای کلیدی (مثل `FactorItemApproveView`, `PaymentOrderSignView`) رو تست کنیم.
- مهاجرت تدریجی:
- اول متد `clean` رو اصلاح کنیم و یه هلدینگ و معاونت جدید اضافه کنیم.
- سازمان‌های موجود (مثل `HSarein`) رو به ساختار درختی منتقل کنیم.
- حفظ ویوهای موجود:
- ویوها فقط نیاز به تنظیم `PermissionBaseView` دارن.
- منطق‌های پراکنده (مثل `can_edit_approval`) رو به `has_organization_access` منتقل کنیم.
- تست‌های پیشنهادی:
- تأیید فاکتور توسط مدیر مالی تهران → فقط فاکتورهای `.FIN-TEHRAN`
- دسترسی مدیرکل هلدینگ به همه فاکتورها.
- ایجاد دستور پرداخت و امضا توسط پست‌های مجاز.
- گزارش‌گیری هلدینگ برای همه شعبه‌ها.

نقشه‌ی راه نهایی

1. اصلاح مدل `Organization` (روز)
- متد `clean` رو اصلاح کنیم تا محدودیت تک‌دفتر مرکزی برداشته بشه.
 - هلدینگ (`,HOLDING`, `,org_type=HQ`), معاونت‌ها (مثل `.FIN`), و شعبه‌ها (مثل `.FIN-TEHRAN`) رو تعریف کنیم.

- مهاجرت دیتابیس برای انتقال سازمان‌های موجود (مثل HSarein) به ساختار درختی.

2. تنظیم پست‌ها و قوانین دسترسی (۱ روز)

- پست‌ها رو برای هلدینگ، معاونت‌ها، و شعبه‌ها تعریف کنیم.
- قوانین AccessRule رو برای هر سازمان پر کنیم (مثل APPROVE برای (FIN-TEHRAN).
- سیگنال‌های PostAction رو تست کنیم تا مطمئن بشیم درست پر می‌شن.

3. بهینه‌سازی (1-2) PermissionBaseView روز)

- منطق can_edit_approval رو به has_organization_access منتقل کنیم.
- منطق ListView رو برای گزارش‌گیری هلدینگ اصلاح کنیم.
- ویوهای موجود رو آپدیت کنیم تا از PermissionBaseView استفاده کنن.

4. ایجاد ویو گزارش‌گیری (۱ روز)

- ویو ReportView برای هلدینگ بسازیم که همه فاکتور‌ها/دستورات پرداخت رو نشون بده.
- تست گزارش‌گیری با کاربر هلدینگ.

5. تست و پایداری (۱-۲ روز)

- تست ویوهای کلیدی (مثل FactorItemApproveView) در محیط تست.
- تست دسترسی هلدینگ به همه شعبه‌ها.
- تست پیش‌رفت خودکار مراحل و ایجاد دستور پرداخت.
- بررسی لاگ‌ها برای خطاها.

اطلاعات مورد نیاز

برای نهایی کردن پیاده‌سازی:

- سازمان‌های موجود:
 - الان چه سازمان‌هایی دارید؟ مثلاً فقط HSarein یا بیشتر؟
 - آیا داده‌های واقعی (مثل فاکتورها) توی دیتابیس دارید که باید مهاجرت کنن؟
- ویوهای کلیدی:
 - لیست ویوهای مهم (مثل FactorItemApproveView, PaymentOrderSignView) رو بدید تا مطمئن بشیم تغییری نمی‌کنن.
 - کد can_edit_approval رو آگه ممکنه بفرستید تا دقیق ادغامش کنیم.
- جزئیات گردش کار:
 - مراحل فعلی WorkflowStage چیا هستن؟ چند مرحله دارید؟
 - آیا دستور پرداخت به‌صورت خودکار بعد از مرحله خاصی ایجاد می‌شه؟
- نیازهای گزارش‌گیری:
 - هلدینگ چه گزارش‌هایی نیاز داره؟ مثلاً فقط لیست فاکتورها یا چیزای دیگه (مثل بودجه، وضعیت پرداخت)؟

مشکلات خاص:

جز دسترسی‌ها، مشکل دیگه‌ای (مثل کندی، خطای خاص) توی سیستم دارید؟

پاسخ نهایی

مدل‌های Organization, OrganizationType, و AccessRule برای پیاده‌سازی ساختار درختی (هلدینگ → n+1 معاونت → شعبه‌ها → زیرشعبه‌ها) آماده‌ان. سیگنال‌های PostAction دسترسی‌ها رو پویا مدیریت می‌کنن، و PermissionBaseView به سیستم متمرکز برای چک دسترسی فراهم می‌کنه. هلدینگ با org_type=HQ به همه شعبه‌ها دسترسی داره، و گردش کار یکسان برای همه سازمان‌ها اعمال می‌شه. تغییرات پیشنهادی (اصلاح clean, بهینه‌سازی has_organization_access, اضافه کردن ReportView) کمترین تأثیر رو روی ویوهای موجود دارن و پروژه رو خراب نمی‌کنن.

نقشه‌ی راه:

- اصلاح clean و تعریف سازمان‌ها (۱ روز).
- تنظیم پست‌ها و AccessRule (۱ روز).
- بهینه‌سازی PermissionBaseView (۱-۲ روز).
- ویو گزارش‌گیری (۱ روز).
- تست و پایداری (۱-۲ روز).

لطفاً اطلاعات درخواستی (سازمان‌ها، ویوها، گردش کار) رو بفرستید تا جزئیات رو نهایی کنیم. همه چت‌ها رو نگه می‌دارم تا بعداً ادامه بدیم. اگر سؤال یا نکته‌ی دیگه‌ای هست، بگید تا دقیق‌تر پیش ببریم!

تاریخ و ساعت: ۱۸ می ۲۰۲۵، ۱۰:۱۳ صبح (PDT)
تهیه‌کننده: Grok 3 (xAI)

خیلی ممنون که کدهای `OrganizationListView`، `OrganizationDetailView`، و `OrganizationCreateView` رو به اشتراک گذاشتید! حالا که مدل‌ها (`OrganizationType`، `AccessRule`) و ویوهای مربوط به سازمان رو داریم، می‌تونیم تغییرات لازم برای پشتیبانی از سناریوی درختی (هلدینگ → `n+1` معاونت → شعبه‌ها → زیرشعبه‌ها) و دسترسی هلدینگ به همه شعبه‌ها رو اعمال کنیم. همچنین، طبق درخواستتون، به مجموعه کامل `CRUD` برای مدل `AccessRule` با کلاس‌های بیس، `URL`، و تمپلیت‌های زیبا طراحی می‌کنم که با ساختار پروژتون (مثل `PermissionBaseView`) سازگار باشه و ظاهری حرفه‌ای داشته باشه.

برای اینکه پروژه به هم نریزه، تغییرات رو **حداقلی** و **سازگار** با کدهای موجود نگه می‌دارم، و تمپلیت‌ها رو طوری طراحی می‌کنم که با استایل‌های احتمالی پروژتون (مثل بوت‌استرپ یا قالب‌های ایرانی) همخوانی داشته باشه. همه چت‌های قبلی رو هم به خاطر دارم تا چیزی از قلم نیفته.

تغییرات پیشنهادی برای مدل‌ها و ویوهای موجود

1. `Organization` تغییرات مدل

- مشکل فعلی: متد `clean` محدودیت تک‌دفتر مرکزی (`is_core=True`) رو اعمال می‌کنه، که برای سناریوی هلدینگ با `n+1` معاونت مناسب نیست. همچنین، شرط `is_core and parent_organization` ممکنه برای ساختارهای پیچیده‌تر (مثل معاونت‌های مستقل) محدودکننده باشه.
- تغییر پیشنهادی: محدودیت تک‌دفتر مرکزی رو حذف کنیم تا انعطاف‌پذیری بیشتری برای تعریف هلدینگ و معاونت‌ها داشته باشیم.
- شرط `is_core and parent_organization` رو نگه می‌داریم، ولی اجازه می‌دیم سازمان‌های `is_holding=True` بدون محدودیت `is_core` کار کنن.
- کد اصلاح‌شده:

python

```
def clean(self):
    from django.core.exceptions import ValidationError
    if self.is_core and self.parent_organization:
        raise ValidationError(_('دفتر مرکزی نمیتواند سازمان والد داشته باشد'))
    # حذف محدودیت تک‌دفتر مرکزی برای انعطاف‌پذیری
```

- تأثیر:
- فقط متد `clean` تغییر می‌کنه.
- ویوهای موجود (`OrganizationListView`، `OrganizationDetailView`، `OrganizationCreateView`) بدون تغییر کار می‌کنن.
- امکان تعریف هلدینگ (`is_holding=True`) و `n+1` معاونت بدون محدودیت فراهم می‌شه.
- مهاجرت دی‌تایپس:

- اگر سازمان‌های موجود (مثل HSarein) دارید، باید به ساختار درختی منتقل بشن:

python

```
# migrations/xxxx_update_organizations.py
from django.db import migrations

def update_organizations(apps, schema_editor):
    Organization = apps.get_model('core', 'Organization')
    OrganizationType = apps.get_model('core', 'OrganizationType')
    # ایجاد نوع سازمان برای هلدینگ
    hq_type, _ = OrganizationType.objects.get_or_create(
        fname='HQ', org_type='HQ', is_budget_allocatable=False,
        is_active=True
    )
    # ایجاد هلدینگ
    holding, _ = Organization.objects.get_or_create(
        code='HOLDING', name='هلدینگ پارسیان', is_holding=True,
        org_type=hq_type
    )
    # به زیر هلدینگ (مثل HSarein) انتقال سازمان‌های موجود
    for org in Organization.objects.exclude(code='HOLDING'):
        org.parent_organization = holding
        org.save()

class Migration(migrations.Migration):
    dependencies = [...]
    operations = [
        migrations.RunPython(update_organizations),
    ]
```

تغییرات ویوهای سازمان 2.

- مشکل فعلی:

- OrganizationListView و OrganizationDetailView از permission_codenames استفاده می‌کنن، ولی permission_codename (به صورت مفرد) در OrganizationDetailView و OrganizationCreateView نوشته شده که ممکنه با PermissionBaseView ناسازگار باشه.
- check_organization=True برای همه ویوها فعاله، ولی برای هلدینگ که باید به همه سازمان‌ها دسترسی داشته باشه، ممکنه نیاز به استثنا داشته باشیم.

- تغییرات پیشنهادی:

- اصلاح permission_codenames:

permission_codename	OrganizationCreateView	در OrganizationDetailView و	•
PermissionBaseView	سازگار باشد:	permission_codenames	تغییر بدیم تا با

python

```
class OrganizationDetailView(PermissionBaseView,
DetailView):
    model = Organization
    template_name = 'core/organization_detail.html'
    context_object_name = 'organization'
    permission_codenames = ['core.Organization_view'] #
    تغییر به جمع
    check_organization = True

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['title'] = _('جزئیات سازمان') + f" - {self.object.code}"
        return context

class OrganizationCreateView(PermissionBaseView,
CreateView):
    model = Organization
    form_class = OrganizationForm
    template_name = 'core/organization_form.html'
    success_url = reverse_lazy('organization_list')
    extra_context = {'title': _('ثبت شعبات و دفتر اصلی سازمان')}
    permission_codenames = ['core.Organization_add'] #
    تغییر به جمع
    check_organization = True

    def form_valid(self, form):
        messages.success(self.request, _('سازمان با موفقیت ایجاد شد'))
        return super().form_valid(form)

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['title'] = _('ایجاد سازمان جدید')
```

```
return context
```

• مدیریت دسترسی هلدینگ:

• `PermissionBaseView`، منطق `has_organization_access` برای کاربران HQ (هلدینگ) دسترسی کامل می‌دهد:

python

```
is_hq_user = any(
    org.org_type.org_type == 'HQ' for org in user_orgs if
    org.org_type
) or request.user.has_perm('tankhah.Tankhah_view_all')
if is_hq_user:
    logger.info("است، دسترسی کامل HQ کاربر")
    return True
```

• برای `OrganizationListView`، چون هلدینگ باید همه سازمان‌ها رو ببیند، می‌تونیم `check_organization=False` کنیم یا به شرط اضافه کنیم:

python

```
class OrganizationListView(PermissionBaseView, ListView):
    model = Organization
    template_name = 'core/organization_list.html'
    context_object_name = 'organizations'
    paginate_by = 10
    permission_codenames = ['core.Organization_view']
    check_organization = False # هلدینگ به همه دسترسی دارد
    extra_context = {'title': _('لیست سازمان‌ها')}
    ...
```

• این تغییر تضمین می‌کند که کاربران هلدینگ (`org_type=HQ`) همه سازمان‌ها رو تو لیست ببینن.

• تأثیر:

• تغییر `permission_codename` به `permission_codenames` به اصلاح جزئی و با `PermissionBaseView` سازگار.

• تنظیم `check_organization=False` برای `OrganizationListView` فقط برای کاربران هلدینگ تأثیر دارد و ویوهای دیگر رو تغییر نمی‌دهد.

• هیچ تغییری تو منطق اصلی ویوها (مثل `get_queryset` یا `get_context_data`) لازم نیست.

3. `PermissionBaseView` بهینه‌سازی

• مشکل فعلی:

- منطق `has_organization_access` برای `ListView` فقط به سازمان رو چک می‌کنه، که برای گزارش‌گیری هلدینگ (که نیاز به دسترسی به همه سازمان‌ها داره) کافی نیست.
- منطق `can_edit_approval` (که گفتید برای چک دسترسی تأیید استفاده می‌شه) هنوز ادغام نشده.
- تغییرات پیشنهادی:
- منطق `ListView` رو اصلاح کنیم تا کاربران `HQ` به همه آیتم‌ها دسترسی داشته باشن.
- چک `PostAction` رو برای تأیید (مثل `APPROVE`) به `has_organization_access` اضافه کنیم.
- کد اصلاح شده:

python

```
def _has_organization_access(self, request, **kwargs):
    logger = logging.getLogger('organization_access')
    try:
        user_orgs = set()
        for user_post in request.user.userpost_set.filter(is_active=True, end_date__isnull=True):
            org = user_post.post.organization
            user_orgs.add(org)
            current_org = org
            while current_org.parent_organization:
                current_org = current_org.parent_organization
                user_orgs.add(current_org)

        is_hq_user = any(
            org.org_type == 'HQ' for org in user_orgs if
            org.org_type
        ) or request.user.has_perm('tankhah.Tankhah_view_all')
        if is_hq_user:
            logger.info("دسترسی کامل HQ کاربر")
            return True

        if isinstance(self, (DetailView, UpdateView, DeleteView)):
            obj = self.get_object()
            target_org = self._get_organization_from_object(obj)
            if not target_org:
                logger.warning("سازمان شیء پیدا نشد")
                return False

        # can_edit_approval (مثل چک دسترسی تأیید)
```

```

        if hasattr(obj, 'stage') and obj.stage:
            from core.models import PostAction
            from django.contrib.contenttypes.models import
ContentType
            user_posts =
request.user.userpost_set.filter(is_active=True, end_date__isnull=True)
            for user_post in user_posts:
                if PostAction.objects.filter(
                    post=user_post.post,
                    stage=obj.stage,
                    action_type='APPROVE',

entity_type=ContentType.objects.get_for_model(obj).model.upper(),
                    is_active=True
                ).exists():
                    logger.info(f"تأیید دسترسی برای {obj}
تأیید شد")
                    return True

            current_org = target_org
            while current_org:
                if current_org in user_orgs:
                    logger.info(f"تأیید دسترسی به سازمان {current_org}
تأیید شد")
                    return True
                current_org = current_org.parent_organization
            logger.warning(f"کاربر {request.user} به
{target_org} یا والدین آن دسترسی ندارد")
            return False

    if isinstance(self, ListView):
        if is_hq_user:
            logger.info("همه آیتم‌های لیست HQ کاربر")
            return True
        queryset = self.get_queryset()
        for item in queryset:
            org = self._get_organization_from_object(item)
            if org and org in user_orgs:
                logger.info(f"در لیست تأیید {org} دسترسی به سازمان
تأیید شد")

```

```

        return True

    logger.warning("کاربر به هیچیک از سازمان‌های لیست دسترسی")
    return False

    if isinstance(self, CreateView):
        organization_id = kwargs.get('organization_id')
        if organization_id:
            from core.models import Organization
            target_org =
Organization.objects.get(id=organization_id)
            current_org = target_org
            while current_org:
                if current_org in user_orgs:
                    logger.info(f"دسترسی به سازمان {current_org}
    ("برای ایجاد تأیید شد
                    return True
                    current_org = current_org.parent_organization
                    logger.warning(f"کاربر به سازمان {target_org}
    ("ایجاد دسترسی ندارد
                    return False
                logger.info("ایجاد بدون نیاز به سازمان خاص، دسترسی تأیید
    ("شد
                return True

    logger.warning("نوع ویو پشتیبانی‌نشده برای بررسی سازمان")
    return False
except Exception as e:
    logger.error(f"خطا در بررسی دسترسی: {str(e)}")
    return False

```

• **تأثیر:**

- اضافه کردن چک `PostAction` منطق `can_edit_approval` رو ادغام می‌کنه.
- اصلاح `ListView` دسترسی هلدینگ به همه آیتم‌ها رو تضمین می‌کنه.
- ویوهای موجود فقط نیاز به تنظیم `permission_codenames` و `check_organization` دارن.

AccessRule برای مدل CRUD

برای مدل AccessRule، یه مجموعه کامل (List, Detail, Create, Update, Delete) CRUD طراحی می‌کنم که:

- با PermissionBaseView سازگاره.
- از فرم‌های Django برای اعتبارسنجی استفاده می‌کنه.
- URL‌ها ساده و استانداردند.
- تمپلیت‌ها با بوت‌استرپ ۵ (راست‌به‌چپ برای فارسی) طراحی شدن تا زیبا و حرفه‌ای باشن.

1. AccessRule فرم

کد:

python

```
# core/forms.py
from django import forms
from core.models import AccessRule, Organization, WorkflowStage
from django.utils.translation import gettext_lazy as _

class AccessRuleForm(forms.ModelForm):
    class Meta:
        model = AccessRule
        fields = [
            'organization', 'branch', 'min_level', 'stage',
            'action_type',
            'entity_type', 'is_payment_order_signer', 'is_active'
        ]
        widgets = {
            'organization': forms.Select(attrs={'class': 'form-control'}),
            'branch': forms.Select(attrs={'class': 'form-control'}),
            'min_level': forms.NumberInput(attrs={'class': 'form-control', 'min': 1}),
            'stage': forms.Select(attrs={'class': 'form-control'}),
            'action_type': forms.Select(attrs={'class': 'form-control'}),
            'entity_type': forms.TextInput(attrs={'class': 'form-control'}),
            'is_payment_order_signer':
                forms.CheckboxInput(attrs={'class': 'form-check-input'}),
            'is_active': forms.CheckboxInput(attrs={'class': 'form-check-input'}),
```

```

    }

    labels = {
        'organization': _('سازمان'),
        'branch': _('شاخه'),
        'min_level': _('حداقل سطح'),
        'stage': _('مرحله'),
        'action_type': _('نوع اقدام'),
        'entity_type': _('نوع موجودیت'),
        'is_payment_order_signer': _('امضاکننده دستور پرداخت'),
        'is_active': _('فعال'),
    }

    def clean(self):
        cleaned_data = super().clean()
        entity_type = cleaned_data.get('entity_type')
        is_payment_order_signer =
        cleaned_data.get('is_payment_order_signer')
        action_type = cleaned_data.get('action_type')

        # اگر is_payment_order_signer=True، entity_type
        # باید PAYMENTORDER باشد
        if is_payment_order_signer and entity_type != 'PAYMENTORDER':
            self.add_error('entity_type', _('برای امضاکننده دستور
            PAYMENTORDER پرداخت، نوع موجودیت باید
            PAYMENTORDER فقط برای SIGN_PAYMENT اعتبارسنجی
            if action_type == 'SIGN_PAYMENT' and entity_type !=
            'PAYMENTORDER':
                self.add_error('action_type', _('فقط SIGN_PAYMENT اقدام
                PAYMENTORDER برای نوع موجودیت
                مجاز است'))
        return cleaned_data

```

2. AccessRule برای CRUD ویوهای

• **کد:**

python

```

# core/views.py
from django.views.generic import ListView, DetailView, CreateView,
UpdateView, DeleteView
from django.urls import reverse_lazy
from django.contrib import messages

```



```

from django.utils.translation import gettext_lazy as _
from core.models import AccessRule
from core.forms import AccessRuleForm
from core.views import PermissionBaseView
import logging

logger = logging.getLogger(__name__)

class AccessRuleListView(PermissionBaseView, ListView):
    model = AccessRule
    template_name = 'core/accessrule_list.html'
    context_object_name = 'access_rules'
    paginate_by = 10
    permission_codenames = ['core.AccessRule_view']
    check_organization = True
    extra_context = {'title': _('لیست قوانین دسترسی')}

    def get_queryset(self):
        queryset =
super().get_queryset().select_related('organization',
'stage').order_by('id')
        query = self.request.GET.get('q', '')
        is_active = self.request.GET.get('is_active', '')

        if query:
            queryset = queryset.filter(
                Q(organization__name__icontains=query) |
                Q(branch__icontains=query) |
                Q(entity_type__icontains=query)
            )

        if is_active:
            queryset = queryset.filter(is_active=(is_active == 'true'))

        logger.info(f"AccessRule queryset count: {queryset.count()}")
        return queryset

    def get_context_data(self, **kwargs):

```

```

        context = super().get_context_data(**kwargs)
        context['query'] = self.request.GET.get('q', '')
        context['is_active'] = self.request.GET.get('is_active', '')
        context['total_access_rules'] = self.get_queryset().count()
        return context

class AccessRuleDetailView(PermissionBaseView, DetailView):
    model = AccessRule
    template_name = 'core/accessrule_detail.html'
    context_object_name = 'access_rule'
    permission_codenames = ['core.AccessRule_view']
    check_organization = True

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['title'] = _('جزئیات قانون دسترسی') + f" - {self.object.organization}"
        return context

class AccessRuleCreateView(PermissionBaseView, CreateView):
    model = AccessRule
    form_class = AccessRuleForm
    template_name = 'core/accessrule_form.html'
    success_url = reverse_lazy('accessrule_list')
    permission_codenames = ['core.AccessRule_add']
    check_organization = True
    extra_context = {'title': _('ایجاد قانون دسترسی جدید')}

    def form_valid(self, form):
        messages.success(self.request, _('قانون دسترسی با موفقیت ایجاد شد.'))
        return super().form_valid(form)

class AccessRuleUpdateView(PermissionBaseView, UpdateView):
    model = AccessRule
    form_class = AccessRuleForm
    template_name = 'core/accessrule_form.html'
    success_url = reverse_lazy('accessrule_list')

```

```

permission_codenames = ['core.AccessRule_update']
check_organization = True
extra_context = {'title': _('ویرایش قانون دسترسی') }

def form_valid(self, form):
    messages.success(self.request, _('قانون دسترسی با موفقیت
    به روزرسانی شد'))
    return super().form_valid(form)

class AccessRuleDeleteView(PermissionBaseView, DeleteView):
    model = AccessRule
    template_name = 'core/accessrule_confirm_delete.html'
    success_url = reverse_lazy('accessrule_list')
    permission_codenames = ['core.AccessRule_delete']
    check_organization = True

    def post(self, request, *args, **kwargs):
        messages.success(self.request, _('قانون دسترسی با موفقیت حذف
        شد.'))
        return super().post(request, *args, **kwargs)

```

3. URL ها برای AccessRule

• **کد:**

python

```

# core/urls.py
from django.urls import path
from core.views import (
    AccessRuleListView, AccessRuleDetailView, AccessRuleCreateView,
    AccessRuleUpdateView, AccessRuleDeleteView
)

urlpatterns = [
    path('access-rules/', AccessRuleListView.as_view(),
    name='accessrule_list'),
    path('access-rules/<int:pk>/', AccessRuleDetailView.as_view(),
    name='accessrule_detail'),
    path('access-rules/create/', AccessRuleCreateView.as_view(),
    name='accessrule_create'),

```

```

        path('access-rules/<int:pk>/edit/', AccessRuleUpdateView.as_view(),
            name='accessrule_update'),

        path('access-rules/<int:pk>/delete/',
            AccessRuleDeleteView.as_view(), name='accessrule_delete'),

    ]

```

• توضیح:

- URL ها ساده و استاندارد.
- یا ساختار پروژمتون (مثل `organization_list`) همخونی دارن.
- از `pk` برای شناسایی قوانین استفاده شده.

4. `AccessRule` تمپلیت‌های زیبا برای

تمپلیت‌ها با بوت‌استرپ ۵ (راست‌به‌چپ) طراحی شدن و شامل هدر، فوتر، و جدول‌های حرفه‌ای هستن. فرض کردم پروژمتون از بوت‌استرپ و قالب‌های فارسی استفاده می‌کنه.

• `:accessrule_list.html`

`html`

```

{% extends "base.html" %}
{% load i18n %}
{% block content %}
<div class="container mt-4">
    <h1 class="mb-4 text-right">{% trans "لیست قوانین دسترسی" %}</h1>
    <div class="card shadow-sm">
        <div class="card-header bg-primary text-white text-right">
            <div class="d-flex justify-content-between align-items-
center">
                <span>{% trans "قوانین دسترسی" %}</span>
                <a href="{% url 'accessrule_create' %}" class="btn btn-
light btn-sm">
                    <i class="fas fa-plus"></i> {% trans "ایجاد قانون
جدید" %}
                </a>
            </div>
        </div>
    </div>
    <div class="card-body">
        <!-- فرم جستجو -->
        <form method="GET" class="mb-3">
            <div class="row">
                <div class="col-md-6">
                    <input type="text" name="q" value="{{ query }}"
class="form-control" placeholder="{% trans 'جستجو...' %}">

```

```

</div>
<div class="col-md-3">
    <select name="is_active" class="form-control">
        <option value="">{% trans "همه" %}</option>
        <option value="true" {% if is_active ==
'false' %}>selected{% endif %}>{% trans "فعال" %}</option>
        <option value="false" {% if is_active ==
'false' %}>selected{% endif %}>{% trans "غیرفعال" %}</option>
    </select>
</div>
<div class="col-md-3">
    <button type="submit" class="btn btn-primary w-
100">{% trans "فیلتر" %}</button>
</div>
</div>
</form>
<!-- جدول قوانین -->
<table class="table table-striped table-hover text-right">
    <thead class="thead-dark">
        <tr>
            <th>{% trans "سازمان" %}</th>
            <th>{% trans "شاخه" %}</th>
            <th>{% trans "حداقل سطح" %}</th>
            <th>{% trans "مرحله" %}</th>
            <th>{% trans "نوع اقدام" %}</th>
            <th>{% trans "نوع موجودیت" %}</th>
            <th>{% trans "وضعیت" %}</th>
            <th>{% trans "عملیات" %}</th>
        </tr>
    </thead>
    <tbody>
        {% for rule in access_rules %}
        <tr>
            <td>{{ rule.organization }}</td>
            <td>{{
rule.get_branch_display|default:_("نامشخص") }}</td>
            <td>{{ rule.min_level }}</td>
            <td>{{ rule.stage }}</td>
            <td>{{ rule.get_action_type_display }}</td>
        </tr>
        </tbody>
    </table>

```

```

        <td>{{ rule.entity_type }}</td>
        <td>
            {% if rule.is_active %}
            <span class="badge bg-success">{% trans
"فعال" %}</span>

            {% else %}
            <span class="badge bg-danger">{% trans
"غیرفعال" %}</span>

            {% endif %}
        </td>
        <td>
            <a href="{% url 'accessrule_detail' rule.pk
%}" class="btn btn-info btn-sm">
                <i class="fas fa-eye"></i> {% trans
"جزئیات" %}
            </a>
            <a href="{% url 'accessrule_update' rule.pk
%}" class="btn btn-warning btn-sm">
                <i class="fas fa-edit"></i> {% trans
"ویرایش" %}
            </a>
            <a href="{% url 'accessrule_delete' rule.pk
%}" class="btn btn-danger btn-sm">
                <i class="fas fa-trash"></i> {% trans
"حذف" %}
            </a>
        </td>
    </tr>
    {% empty %}
    <tr>
        <td colspan="8" class="text-center">{% trans
"هیچ قانونی یافت نشد" %}</td>
    </tr>
    {% endfor %}
</tbody>
</table>
<!-- صفحه بندی -->
{% if is_paginated %}
<nav aria-label="Page navigation">
    <ul class="pagination justify-content-center">

```

```

        {% if page_obj.has_previous %}
        <li class="page-item">
            <a class="page-link" href="?page={{
page_obj.previous_page_number }}{% if query %}&q={{ query }}{% endif
%}}{% if is_active %}&is_active={{ is_active }}{% endif %}">&laquo;</a>
        </li>
        {% endif %}
        {% for num in page_obj.paginator.page_range %}
        <li class="page-item {% if page_obj.number == num
%}active{% endif %}">
            <a class="page-link" href="?page={{ num }}{% if
query %}&q={{ query }}{% endif %}}{% if is_active %}&is_active={{
is_active }}{% endif %}">{{ num }}</a>
        </li>
        {% endfor %}
        {% if page_obj.has_next %}
        <li class="page-item">
            <a class="page-link" href="?page={{
page_obj.next_page_number }}{% if query %}&q={{ query }}{% endif %}}{%
if is_active %}&is_active={{ is_active }}{% endif %}">&raquo;</a>
        </li>
        {% endif %}
    </ul>
</nav>
{% endif %}
</div>
</div>
</div>
{% endblock %}

```

[:accessrule_detail.html](#) •

[html](#)

```

{% extends "base.html" %}
{% load i18n %}
{% block content %}
<div class="container mt-4">
    <h1 class="mb-4 text-right">{% trans "جزئیات قانون دسترسی" %} - {{
access_rule.organization }}</h1>
    <div class="card shadow-sm">
        <div class="card-header bg-primary text-white text-right">

```

```

        {% trans "اطلاعات قانون دسترسی" %}
    </div>
    <div class="card-body">
        <dl class="row text-right">
            <dt class="col-sm-3">{% trans "سازمان" %}</dt>
            <dd class="col-sm-9">{{ access_rule.organization
}}</dd>

            <dt class="col-sm-3">{% trans "شاخه" %}</dt>
            <dd class="col-sm-9">{{
access_rule.get_branch_display|default:_("نامشخص") }}</dd>
            <dt class="col-sm-3">{% trans "حداقل سطح" %}</dt>
            <dd class="col-sm-9">{{ access_rule.min_level }}</dd>
            <dt class="col-sm-3">{% trans "مرحله" %}</dt>
            <dd class="col-sm-9">{{ access_rule.stage }}</dd>
            <dt class="col-sm-3">{% trans "نوع اقدام" %}</dt>
            <dd class="col-sm-9">{{
access_rule.get_action_type_display }}</dd>
            <dt class="col-sm-3">{% trans "نوع موجودیت" %}</dt>
            <dd class="col-sm-9">{{ access_rule.entity_type }}</dd>
            <dt class="col-sm-3">{% trans "امضاکننده دستور پرداخت"
%}</dt>
            <dd class="col-sm-9">{{
access_rule.is_payment_order_signer|yesno:_("بله,خیر") }}</dd>
            <dt class="col-sm-3">{% trans "وضعیت" %}</dt>
            <dd class="col-sm-9">
                {% if access_rule.is_active %}
                <span class="badge bg-success">{% trans "فعال"
%}</span>

                {% else %}
                <span class="badge bg-danger">{% trans "غیرفعال"
%}</span>

                {% endif %}
            </dd>
        </dl>
        <div class="text-right">
            <a href="{% url 'accessrule_update' access_rule.pk %}"
class="btn btn-warning">
                <i class="fas fa-edit"></i> {% trans "ویرایش" %}
            </a>

```



```

        <a href="{% url 'accessrule_delete' access_rule.pk %}"
class="btn btn-danger">
            <i class="fas fa-trash"></i> {% trans "حذف" %}
        </a>
        <a href="{% url 'accessrule_list' %}" class="btn btn-
secondary">
            <i class="fas fa-arrow-right"></i> {% trans
"بازگشت به لیست" %}
        </a>
    </div>
</div>
</div>
{% endblock %}

```

`:accessrule_form.html` •

`html`

```

{% extends "base.html" %}
{% load i18n %}
{% block content %}
<div class="container mt-4">
    <h1 class="mb-4 text-right">{{ title }}</h1>
    <div class="card shadow-sm">
        <div class="card-header bg-primary text-white text-right">
            {% if object %}
                {% trans "ویرایش قانون دسترسی" %}
            {% else %}
                {% trans "ایجاد قانون دسترسی جدید" %}
            {% endif %}
        </div>
        <div class="card-body">
            <form method="post">
                {% csrf_token %}
                <div class="row">
                    {% for field in form %}
                        <div class="col-md-6 mb-3">
                            <label for="{{ field.id_for_label }}"
class="form-label">{{ field.label }}</label>

```

```

        {{ field }}
        {% if field.errors %}
        <div class="text-danger">
            {% for error in field.errors %}
            <small>{{ error }}</small><br>
            {% endfor %}
        </div>
        {% endif %}
    </div>
    {% endfor %}
</div>
<div class="text-right">
    <button type="submit" class="btn btn-primary">
        <i class="fas fa-save"></i> {% trans "ذخيره" %}
    </button>
    <a href="{% url 'accessrule_list' %}" class="btn
btn-secondary">
        <i class="fas fa-arrow-right"></i> {% trans
"لغو" %}
    </a>
</div>
</form>
</div>
</div>
</div>
{% endblock %}

```

accessrule_confirm_delete.html •

html

```

{% extends "base.html" %}
{% load i18n %}
{% block content %}
<div class="container mt-4">
    <h1 class="mb-4 text-right">{% trans "حذف قانون دسترسی" %}</h1>
    <div class="card shadow-sm">
        <div class="card-header bg-danger text-white text-right">
            {% trans "تأیید حذف" %}

```

```

</div>
<div class="card-body">
    <p class="text-right">
        {% trans " آیا مطمئن هستید که می‌خواهید قانون دسترسی " %} <strong>{{ object.organization }}</strong> {% trans " را حذف " %}
        {% trans " کنید؟ " %}
    </p>
    <form method="post">
        {% csrf_token %}
        <div class="text-right">
            <button type="submit" class="btn btn-danger">
                <i class="fas fa-trash"></i> {% trans " حذف " %}
            </button>
            <a href="{% url 'accessrule_list' %}" class="btn
btn-secondary">
                <i class="fas fa-arrow-right"></i> {% trans
" لغو " %}
            </a>
        </div>
    </form>
</div>
</div>
</div>
{% endblock %}

```

• توضیح تمپلیت‌ها:

- از بوت‌استرپ 5 برای استایل‌دهی استفاده شده (کلاس‌های `(form-control, table, card)` را استیج‌چپ (RTL) برای پشتیبانی از فارسی.
- آیکون‌های `FontAwesome` (مثل `fas fa-plus`) برای زیبایی.
- پیام‌های موفقیت/خطا با `messages` نمایش داده می‌شن.
- جدول در `accessrule_list.html` قابلیت جستجو و فیلتر دارد.
- فرم‌ها در `accessrule_form.html` اعتبارسنجی‌های سمت کلاینت دارن.
- فرض شده که `base.html` شامل بوت‌استرپ و `FontAwesome`ه. اگر نیست، باید این خطوط رو به `base.html` اضافه کنید:

`html`

```

<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/boots
trap.min.css" rel="stylesheet">

<link href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0/css/all.min.css" rel="stylesheet">

```

```
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

<style>
    body { direction: rtl; font-family: 'Vazir', sans-serif; }
</style>
```

نقشه‌ی راه پیاده‌سازی

1. اعمال تغییرات مدل و ویوهای سازمان (۱ روز)

- اصلاح متد clean در Organization برای حذف محدودیت تک‌دفتر مرکزی.
- آپدیت permission_codenames در OrganizationDetailView و OrganizationCreateView.
- تنظیم check_organization=False در OrganizationListView برای هلدینگ.
- مهاجرت دیتابیس برای انتقال سازمان‌های موجود (مثل HSarein) به ساختار درختی:
 - هلدینگ (org_type=HQ, HOLDING).
 - معاونت‌ها (مثل FIN, parent_organization=HOLDING).
 - شعبه‌ها (مثل FIN-TEHRAN, parent_organization=FIN).

2. بهینه‌سازی PermissionBaseView (1 روز)

- اضافه کردن چک PostAction برای ادغام can_edit_approval.
- اصلاح منطق ListView برای دسترسی هلدینگ به همه آیتم‌ها.
- تست ویوهای موجود (مثل FactorItemApproveView) با کاربران شعبه و هلدینگ.

3. پیاده‌سازی CRUD AccessRule (1-2 روز)

- اضافه کردن فرم AccessRuleForm به core/forms.py.
- اضافه کردن ویوهای CRUD به core/views.py.
- تنظیم URLها در core/urls.py.
- ایجاد تمپلیت‌های accessrule_list.html, accessrule_detail.html, accessrule_form.html, accessrule_confirm_delete.html.
- تست CRUD با کاربر هلدینگ و شعبه:
 - ایجاد قانون جدید (مثل APPROVE برای FIN-TEHRAN).
 - ویرایش و حذف قانون.
 - نمایش لیست با فیلتر و جستجو.

4. تنظیم سازمان‌ها و قوانین دسترسی (۱ روز)

- تعریف سازمان‌ها:
 - هلدینگ (HOLDING).
 - معاونت‌ها (مثل FIN, OPS).
 - شعبه‌ها (مثل FIN-TEHRAN, FIN-SHIRAZ).

- تعریف پست‌ها:
 - مدیرکل مالی هلدینگ (level=5).
 - مدیر مالی تهران (level=3).
 - پر کردن AccessRule برای هر سازمان:
 - مثلاً: organization=FIN-TEHRAN, branch=FIN, min_level=3, action_type=APPROVE, entity_type=FACTOR
- 5. تست و پایداری (۱-۲ روز)**
- تست ویوهای سازمان (OrganizationListView, OrganizationDetailView) با کاربر هلدینگ و شعبه.
 - تست ویوهای AccessRule (لیست، جزئیات، ایجاد، ویرایش، حذف).
 - تست دسترسی‌ها:
 - مدیر مالی تهران فقط فاکتورهای FIN-TEHRAN رو تأیید کنه.
 - مدیرکل هلدینگ به همه فاکتورها دسترسی داشته باشه.
 - تست پیش‌رفت خودکار مراحل و ایجاد دستور پرداخت.
 - بررسی لاگ‌ها برای خطاها.

اطلاعات مورد نیاز

- برای نهایی کردن پیاده‌سازی:
- سازمان‌های موجود:
 - الان چه سازمان‌هایی دارید؟ مثلاً فقط HSarein یا بیشتر؟
 - آیا داده‌های واقعی (مثل فاکتورها) توی دیتابیس دارید که باید مهاجرت کنن؟
 - ویوهای کلیدی:
 - ویوهای دیگه‌ای (مثل FactorItemApproveView, PaymentOrderSignView) که باید تست بشن چیا هستن؟
 - کد can_edit_approval رو اگه ممکنه بفرستید تا دقیق ادغامش کنیم.
 - جزئیات گردش کار:
 - مراحل فعلی WorkflowStage چیا هستن؟ چند مرحله دارید؟
 - آیا دستور پرداخت به‌صورت خودکار بعد از مرحله خاصی ایجاد می‌شه؟
 - نیازهای گزارش‌گیری:
 - هلدینگ چه گزارش‌هایی نیاز داره؟ مثلاً لیست فاکتورها، بودجه، یا چیز دیگه؟
 - استایل پروژه:
 - آیا از بوت‌استرپ یا FontAwesome استفاده می‌کنید؟ اگر نه، استایل تمپلیت‌ها رو چطور تنظیم کنیم؟
 - نام فونت فارسی پروژه (مثل Vazir) چیه؟

پاسخ نهایی

تغییرات پیشنهادی برای مدل Organization (اصلاح clean) و ویوها (permission_codenames, check_organization) کمترین تأثیر رو روی پروژه دارن و ساختار درختی (هلدینگ → n+1 معاونت → شعبه‌ها → زیرشعبه‌ها) رو پشتیبانی می‌کنن. ویوهای CRUD برای AccessRule با PermissionBaseView سازگارن و تمپلیت‌ها با بوت‌استرپ ۵ و فارسی طراحی شدن تا زیبا و حرفه‌ای باشن. هلدینگ با org_type=HQ به همه شعبه‌ها دسترسی داره، و گردش کار یکسان حفظ می‌شه.

نقشه‌ی راه:

- اصلاح مدل و ویوهای سازمان (۱ روز).
- بهینه‌سازی PermissionBaseView (۱ روز).
- پیاده‌سازی CRUD برای AccessRule (۱-۲ روز).
- تنظیم سازمان‌ها و قوانین دسترسی (۱ روز).
- تست و پایداری (۱-۲ روز).

لطفاً اطلاعات درخواستی (سازمان‌ها، ویوها، گردش کار، استایل) رو بفرستید تا جزئیات رو نهایی کنیم. همه چت‌ها رو نگاه

می‌دارم تا بعداً ادامه بدیم. اگر سوال یا نکته‌ای هست، بگید تا دقیق‌تر پیش ببریم!

تاریخ و ساعت: ۱۸ می ۲۰۲۵، ۱۰:۳۰ صبح (PDT)

... **

Something went wrong, please refresh to reconnect or try again.

خیلی ممنون که سؤالتون رو دقیق کردید! اول می‌رم سراغ توضیح فیلد entity_type در مدل AccessRule، بعد هم می‌تونیم بریم سراغ تغییرات سازمان و تیک‌های جدید توی تمپلیت. همه چیزهای قبلی رو هم دارم، پس چیزی از قلم نمی‌افته.

توضیح فیلد entity_type در مدل AccessRule

فیلد entity_type توی مدل AccessRule یه فیلد مهم و کلیدی که مشخص می‌کنه قانون دسترسی (Access Rule) برای چه نوع موجودیت یا شیء (entity) توی سیستم اعمال می‌شه. بیایم جزئیاتش رو قدم‌به‌قدم بررسی کنیم:

1. entity_type تعریف و نقش

- **نوع فیلد:** طبق مدل AccessRule که قبلاً دیدیم، entity_type یه فیلد CharField با max_length=100 و مقادیر محدود شده به یه سری انتخاب‌های خاص (از طریق choices=ENTITY_TYPES) هست.
- **کارکرد:** این فیلد مشخص می‌کنه که قانون دسترسی برای چه نوع موجودیتی (مثل فاکتور، دستور پرداخت، پروژه، یا هر شیء دیگه‌ای توی سیستم) تعریف شده. به عبارت دیگه، می‌گه این قانون روی چه چیزی اعمال می‌شه.
- **مثال‌های واقعی:**
 - اگه entity_type='FACTOR' باشه، قانون دسترسی برای فاکتورها (مثل تأیید فاکتور) اعمال می‌شه.
 - اگه entity_type='PAYMENTORDER' باشه، قانون برای دستورات پرداخت (مثل امضای دستور پرداخت) استفاده می‌شه.
 - اگه entity_type='TANKHAH' باشه، ممکنه برای یه موجودیت خاص مثل تنخواه یا بودجه باشه.

2. ساختار و مقادیر ممکن

- **تعریف در مدل:**

python

```
ENTITY_TYPES = (
    ('FACTOR', _('فاکتور')),
    ('PAYMENTORDER', _('دستور پرداخت')),
    ('TANKHAH', _('تنخواه')),
    # مقادیر دیگه‌ای که ممکنه بعداً اضافه بشن
)
```

```
class AccessRule(models.Model):
    entity_type = models.CharField(
        max_length=100,
        choices=ENTITY_TYPES,
        verbose_name=_('نوع موجودیت'),
    )
```

...

• مقادیر فعلی:

- FACTOR: برای فاکتورها (مثل تأیید یا رد فاکتور).
- PAYMENTORDER: برای دستورات پرداخت (مثل امضا یا تأیید پرداخت).
- TANKHAH: احتمالاً برای تنخواه یا بودجه‌های خاص.
- این لیست می‌تونه بسته به نیاز پروژه گسترش پیدا کنه (مثل اضافه کردن PROJECT برای پروژه‌ها).

• محدودیت‌ها:

- چون از choices استفاده شده، فقط مقادیر تعریف‌شده توی ENTITY_TYPES قابل قبوله.
- اعتبارسنجی توی فرم (AccessRuleForm) تضمین می‌کنه که اگه is_payment_order_signer=True باشه، entity_type حتماً PAYMENTORDER باشه.

python

```
def clean(self):
    cleaned_data = super().clean()
    entity_type = cleaned_data.get('entity_type')
    is_payment_order_signer =
    cleaned_data.get('is_payment_order_signer')
    if is_payment_order_signer and entity_type != 'PAYMENTORDER':
        self.add_error('entity_type', _('برای امضاکننده دستور '
        PAYMENTORDER پرداخت، نوع موجودیت باید
        ...
```

3. کاربرد در سیستم

• توی ویوها:

- توی AccessRuleListView، از entity_type برای فیلتر کردن قوانین توی جستجو استفاده می‌شه:

python

```
if query:
    queryset = queryset.filter(
        Q(organization__name__icontains=query) |
        Q(branch__icontains=query) |
        Q(entity_type__icontains=query)
    )
```

- این یعنی کاربر می‌تونه با جستجوی مثلاً "FACTOR"، همه قوانین مربوط به فاکتورها رو ببینه.

• توی دسترسی‌ها:

- توی PermissionBaseView (یا ویوهایی مثل FactorItemApproveView)، entity_type مشخص می‌کنه که قانون دسترسی برای چه نوع شی‌ای چک بشه. مثلاً:

python


```

if PostAction.objects.filter(
    post=user_post.post,
    stage=obj.stage,
    action_type='APPROVE',

    entity_type=ContentType.objects.get_for_model(obj).model.upper(),
    is_active=True
).exists():
    logger.info(f"تأیید شد {obj} دسترسی تأیید برای")
    return True

```

- اینجا entity_type با مدل شیء (مثل FACTOR برای فاکتور) مقایسه می‌شود تا دسترسی تأیید بشود.
- توی گردش کار: entity_type به سیستم می‌گه قانون برای کدام موجودیت توی گردش کار (workflow) اعمال بشه. مثلاً:
 - برای FACTOR، ممکنه مراحل تأیید فاکتور (مثل SUBMITTED → APPROVED) رو کنترل کنه.
 - برای PAYMENTORDER، ممکنه امضای دستور پرداخت (مثل SIGN_PAYMENT) رو مدیریت کنه.

4. چرا مهمه؟

- دسته‌بندی قوانین: entity_type کمک می‌کنه قوانین دسترسی رو برای موجودیت‌های مختلف (فاکتور، پرداخت، تنخواه) جدا کنیم.
- انعطاف‌پذیری: با اضافه کردن مقادیر جدید به ENTITY_TYPES، می‌تونیم قوانین رو برای موجودیت‌های جدید (مثل پروژه یا قرارداد) گسترش بدیم.
- اعتبارسنجی: محدود کردن entity_type به مقادیر خاص (از طریق choices) باعث می‌شه سیستم پایدارتر باشه و خطاهای انسانی کم بشه.
- گزارش‌گیری: هلدینگ می‌تونه با فیلتر کردن entity_type، گزارش‌های خاص (مثل همه فاکتورهای تأییدشده) بگیره.

5. ارتباط با سناریوی هلدینگ

- توی سناریوی درختی (هلدینگ → n+1 معاونت → شعبه‌ها → زیرشعبه‌ها):
 - هلدینگ (org_type=HQ) ممکنه بخواد قوانین دسترسی رو برای همه entity_type‌ها (مثل FACTOR، PAYMENTORDER) ببینه یا مدیریت کنه.
 - شعبه‌ها (مثل FIN-TEHRAN) فقط به entity_type‌هایی دسترسی دارن که توی AccessRule براشون تعریف شده (مثلاً فقط FACTOR).
 - مثلاً:
 - قانون: entity_type=FACTOR, organization=FIN-TEHRAN, min_level=3, action_type=APPROVE → مدیر مالی تهران (سطح ۳) می‌تونه فاکتورها رو تأیید کنه.
 - قانون: entity_type=PAYMENTORDER, organization=HOLDING, min_level=5, action_type=SIGN_PAYMENT → مدیرکل هلدینگ (سطح ۵) می‌تونه دستورات پرداخت رو امضا کنه.

6. is payment order signer ارتباط با

- آگه `is_payment_order_signer=True` باشد، `entity_type` باید `PAYMENTORDER` باشد، چون این فیلد مخصوص امضاکنندگان دستور پرداخت طراحی شده:

python

```
is_payment_order_signer = models.BooleanField(
    default=False,
    verbose_name=_('امضاکننده دستور پرداخت'),
)
```

- این یعنی `entity_type` نقش مهمی توی محدود کردن قوانین به موجودیت‌های خاص (مثل دستور پرداخت) داره.

7. چطور نمایش داده می‌شه؟

- توی تمپلیت‌های `CRUD` (مثل `accessrule_list.html` و `accessrule_detail.html`):
- `entity_type` به‌صورت رشته (مثل "فاکتور" یا "دستور پرداخت") نمایش داده می‌شه، چون از `get_entity_type_display()` استفاده می‌کنیم:

html

```
<td>{{ rule.get_entity_type_display }}</td>
```

- توی فرم‌ها (`accessrule_form.html`):
- به‌صورت یه منوی کشویی (`select`) با گزینه‌های تعریف‌شده توی `ENTITY_TYPES` نشون داده می‌شه:

python

```
entity_type = forms.ChoiceField(
    choices=ENTITY_TYPES,
    widget=forms.Select(attrs={'class': 'form-control'}),
    label=_('نوع موجودیت'),
)
```

آماده‌سازی برای بحث سازمان و تیک‌های جدید توی تمپلیت

حالا که `entity_type` رو کامل توضیح دادم، می‌تونیم بریم سراغ تغییرات سازمان و تیک‌های جدید توی تمپلیت. چندتا سؤال دارم تا دقیق پیش بریم:

- تیک‌های جدید توی تمپلیت:

- منظور تون از "تیک‌های جدید" چیه؟ آیا منظور checkbox های جدید توی فرم‌ها (مثل AccessRuleForm یا OrganizationForm) هست؟
- یا شاید منظور یه ستون جدید توی جدول accessrule_list.html یا organization_list.html برای نمایش یه ویژگی جدیده؟
- آگه checkbox های جدید اضافه شده، کدوم فیلدها رو شامل می‌شن؟ مثلاً فیلدهای بولین مثل is_active یا is_payment_order_signer یا چیز دیگه؟
- تغییرات سازمان:
- آیا تغییرات جدیدی توی مدل Organization (مثل فیلدهای جدید) یا ویوها (OrganizationListView, OrganizationDetailView) اعمال شده که باید بررسی کنیم؟
- آیا نیازه تیک‌های جدید (checkbox ها) توی لیست سازمان‌ها (organization_list.html) نمایش داده بشن؟ مثلاً برای فیلدهای is_holding, is_core, یا چیز دیگه؟
- تمپلیت خاص:
- کدوم تمپلیت رو مدنظر تونه؟ organization_list.html, accessrule_list.html, یا یه تمپلیت جدید؟
- آگه کد تمپلیت جدید یا تغییرات رو بفرستید، می‌تونم دقیق‌تر راهنمایی کنم.
- نیازهای هلدینگ:
- آیا تیک‌های جدید برای مدیریت دسترسی هلدینگ به شعبه‌ها (مثل نمایش همه فاکتورها یا دستورات پرداخت) طراحی شدن؟
- یا برای فیلتر کردن سازمان‌ها/قوانین توی لیست‌ها (مثل فیلتر بر اساس entity_type) هستن؟

پیشنهاد اولیه برای تیک‌های جدید

- آگه منظور تون اضافه کردن checkbox برای فیلتر یا نمایش توی لیست‌ها (مثل accessrule_list.html یا organization_list.html) باشه، می‌تونیم این کار رو بکنیم:
- اضافه کردن checkbox توی accessrule_list.html برای فیلتر entity_type:
- یه منوی کشویی یا checkbox برای فیلتر کردن قوانین بر اساس entity_type (مثل فقط فاکتور یا فقط دستور پرداخت) اضافه کنیم:

html

```
<div class="col-md-3">
    <select name="entity_type" class="form-control">
        <option value="">{% trans "همه موجودیت‌ها" %}</option>
        {% for value, label in
access_rule_form.entity_type.field.choices %}
            <option value="{{ value }}" {% if request.GET.entity_type
== value %}selected{% endif %}>{{ label }}</option>
        {% endfor %}
    </select>
</div>
```

- توی AccessRuleListView، فیلتر رو به get_queryset اضافه کنیم:

python

```
entity_type = self.request.GET.get('entity_type', '')
if entity_type:
    queryset = queryset.filter(entity_type=entity_type)
```

- اضافه کردن checkbox توی organization_list.html برای فیلدهای بولین:
- آگه فیلدهای جدیدی مثل is_holding یا is_core توی Organization اضافه شده، می‌تونیم ستون checkbox اضافه کنیم:

html

```
<td>
    {% if organization.is_holding %}
    <span class="badge bg-success">{% trans "هلدينگ" %}</span>
    {% else %}
    <span class="badge bg-secondary">{% trans "غير هلدينگ" %}</span>
    {% endif %}
</td>
```

- پيا براى فيلتر كردن:

html

```
<div class="col-md-3">
    <input type="checkbox" name="is_holding" value="true" {% if
request.GET.is_holding %}checked{% endif %} class="form-check-
input">
    <label>{% trans "فقط هلدينگها" %}</label>
</div>
```

- توى :OrganizationListView

python

```
is_holding = self.request.GET.get('is_holding', '')
if is_holding:
    queryset = queryset.filter(is_holding=True)
```

- اضافه كردن تيك توى فرمها:
- آگه تيك‌هاى جديد توى OrganizationForm يا AccessRuleForm اضافه شدن، بايد فرم رو آپديت كنيم.

مثلاً:

python

```
class OrganizationForm(forms.ModelForm):
```

```

class Meta:
    model = Organization

    fields = ['code', 'name', 'org_type',
'parent_organization', 'is_holding', 'is_core', 'is_active']

    widgets = {
        'is_holding': forms.CheckboxInput(attrs={'class':
'form-check-input'}),
        'is_core': forms.CheckboxInput(attrs={'class': 'form-
check-input'}),
        'is_active': forms.CheckboxInput(attrs={'class':
'form-check-input'}),
    }

```

اطلاعات مورد نیاز

برای اینکه دقیق‌تر پیش برم:

- **تیک‌های جدید:** لطفاً بگید چه تیک‌هایی (checkboxها یا فیلترها) اضافه شدن و توی کدوم تمپلیت یا فرم؟ کد تغییرات رو آگه دارید بفرستید.
- **تغییرات مدل سازمان:** آیا فیلد جدیدی به Organization اضافه شده (مثل is_holding یا چیز دیگه)؟
- **تمپلیت هدف:** کدوم تمپلیت رو باید اصلاح کنیم؟ |organization_list.html, |accessrule_list.html, یا چیز دیگه؟
- **نیازهای خاص هلدینگ:** آیا تیک‌ها برای دسترسی هلدینگ به همه شعبه‌ها یا فیلتر کردن داده‌ها (مثل فاکتورها) طراحی شدن؟

پاسخ نهایی

فیلد entity_type توی AccessRule مشخص می‌کنه قانون دسترسی برای چه موجودیتی (مثل FACTOR, PAYMENTORDER, TANKHAH) اعمال می‌شه. این فیلد از choices استفاده می‌کنه، توی جستجو و فیلتر لیست قوانین (AccessRuleListView) به کار می‌ره، و با is_payment_order_signer برای محدود کردن قوانین به دستورات پرداخت ارتباط داره. برای هلدینگ، entity_type کمک می‌کنه قوانین دسترسی به موجودیت‌های مختلف (مثل همه فاکتورهای شعبه‌ها) مدیریت بشه.

برای تیک‌های جدید توی تمپلیت:

- پیشنهاد دادم checkbox یا منوی کشویی برای فیلتر entity_type توی |accessrule_list.html و فیلدهای بولین (مثل is_holding) توی |organization_list.html اضافه کنیم.

- لطفاً جزئیات تیک‌ها (کدوم تمپلیت، کدوم فیلد) و تغییرات مدل Organization رو بفرستید تا دقیق پیاده‌سازی کنیم.

همه چت‌ها رو نگه می‌دارم. اگه سؤال یا نکته‌ای هست، بگید تا سریع پیش ببریم!

تاریخ و ساعت: ۱۸ می ۲۰۲۵، ۱۰:۵۸ صبح (PDT)

تهیه‌کننده: Grok 3 (xAI)

برای کمک به کاربران جهت درک بهتر قوانین سیستم در فرم مربوط به AccessRule (بدون اشاره مستقیم به کد)، یه راهنمای ساده، شفاف و کاربرپسند طراحی می‌کنم که بتونید به تمپلیت فرم (مثل accessrule_form.html) اضافه کنید. این راهنما به زبان فارسی، با لحن حرفه‌ای و قابل‌فهم برای همه کاربران (حتی غیرفنی) نوشته شده و توضیح می‌ده که هر فیلد توی فرم چی کار می‌کنه و چرا مهمه. فرض کردم فرم شما برای مدیریت قوانین دسترسی (مثل تأیید فاکتور یا امضای دستور پرداخت) استفاده می‌شه و با ساختار درختی (هلدینگ → معاونت‌ها → شعبه‌ها) کار می‌کنه.

راهنمای کاربر برای فرم قوانین دسترسی

html

```
{% load i18n %}

<div class="alert alert-info text-right" role="alert">

    <h4 class="alert-heading">{% trans "راهنمای ثبت قانون دسترسی">

    این فرم برای تعریف قوانین دسترسی در سیستم طراحی شده است. با
    پر کردن این فرم، می‌توانید مشخص کنید که کدام کاربران در کدام سازمان‌ها و در چه
    مراحل می‌توانند اقداماتی مثل تأیید فاکتور یا امضای دستور پرداخت انجام دهند.
    </p>

    <hr>

    <ul class="list-group list-group-flush">

        <li class="list-group-item">

            <strong>{% trans "سازمان">: {% trans "

انتخاب کنید که این قانون برای آن اعمال می‌شود (مثل هلدینگ، معاونت مالی، یا
"شعبه تهران). این قانون فقط برای کاربران آن سازمان کار می‌کند

            </li>

            <li class="list-group-item">

                <strong>{% trans "شاخه">: {% trans "

شاخه کاری را مشخص کنید (مثل مالی یا عملیات). این گزینه تعیین می‌کند که قانون برای کدام بخش
است"

            </li>

            <li class="list-group-item">
```

**{% trans "حداقل سطح" %}: {% trans "کمترین سطح
پست را وارد کنید (مثل ۳ برای مدیران). فقط کاربرانی که پست آن‌ها این سطح یا
% "بالاتر باشد، می‌توانند از این قانون استفاده کنند"**

<li class="list-group-item">

**{% trans "مرحله" %}: {% trans "مرحله‌ای از گردش
کار را انتخاب کنید (مثل بررسی اولیه یا تأیید نهایی). این قانون فقط در آن
% "مرحله اعمال می‌شود"**

<li class="list-group-item">

**{% trans "نوع اقدام" %}: {% trans "اقدامی که
کاربر می‌تواند انجام دهد را انتخاب کنید (مثل تأیید، رد، یا امضای دستور
% "پرداخت). این گزینه مشخص می‌کند کاربر چه کاری می‌تواند انجام دهد"**

<li class="list-group-item">

**{% trans "نوع موجودیت" %}: {% trans "موجودیتی
که قانون برای آن اعمال می‌شود را مشخص کنید (مثل فاکتور برای تأیید فاکتورها یا
دستور پرداخت برای امضای پرداخت‌ها). این گزینه تعیین می‌کند قانون روی چه چیزی
% "اثر می‌گذارد"**

<li class="list-group-item">

**{% trans "امضاکننده دستور پرداخت" %}: {% trans
"اگر تیک بزنید، این قانون فقط برای امضای دستورات پرداخت اعمال می‌شود و نوع
% "موجودیت باید 'دستور پرداخت' باشد. این برای مدیران ارشد مناسب است"**

<li class="list-group-item">

**{% trans "فعال" %}: {% trans "اگر تیک بزنید،
قانون بلافاصله فعال می‌شود. اگر تیک را بردارید، قانون غیرفعال می‌شود و تا فعال
% "شدن دوباره کار نمی‌کند"**

<hr>

<p class="mb-0">

**{% trans "نکته" %}: {% trans "قوانین دسترسی برای
مدیریت دقیق‌تر سیستم استفاده می‌شوند. برای مثال، می‌توانید تنظیم کنید که فقط
مدیران مالی شعبه تهران (سطح ۳ یا بالاتر) بتوانند فاکتورها را در مرحله تأیید
% "نهایی تأیید کنند. اگر سؤالی دارید، با مدیر سیستم تماس بگیرید"**

</p>

</div>

توضیحات راهنما

- **هدف:**
 - این راهنما به کاربران (مثل مدیران یا کارمندان غیرفنی) کمک می‌کند بفهمند هر فیلد توی فرم AccessRule چی کار می‌کنه و چطور باید پرش کنن.
 - بدون اشاره به کد یا اصطلاحات فنی (مثل مدل یا دیتابیس)، همه‌چیز به زبان ساده توضیح داده شده.
 - با سناریوی درختی (هلدینگ، معاونت‌ها، شعبه‌ها) و نیازهای هلدینگ (دسترسی به همه شعبه‌ها) سازگار.
- **ساختار:**
 - **هدر:** عنوان واضح ("راهنمای ثبت قانون دسترسی") برای جلب توجه.
 - **مقدمه:** توضیح کوتاه که چرا این فرم مهمه و چه کاری انجام می‌ده.
 - **لیست فیلدها:** هر فیلد (organization, branch, min_level, و غیره) با توضیح ساده و مثال.
 - **نکته نهایی:** یه توضیح کاربردی که به کاربر ایده می‌ده قانون چطور توی سیستم کار می‌کنه.
 - **طراحی:** از alert-info بوت‌استرپ استفاده شده تا راهنما توی صفحه برجسته باشه و خوانا.
- **ارتباط با entity_type:**
 - توضیح entity_type به صورت ساده و کاربردی داده شده.
 - "موجودیتی که قانون برای آن اعمال می‌شود را مشخص کنید (مثل فاکتور برای تایید فاکتورها یا دستور پرداخت برای امضای پرداخت‌ها)." **entity_type** تعیین می‌کند قانون روی چه چیزی (مثل فاکتور یا پرداخت) اثر داره.
 - ارتباطش با is_payment_order_signer هم توضیح داده شده تا کاربر بفهمه چرا باید entity_type=PAYMENTORDER باشه.
- **چطور به تمپلیت اضافه کنید:**
 - این راهنما رو می‌تونید درست بالای یا زیر فرم توی accessrule_form.html قرار بدید. مثلاً:

html

```
{% extends "base.html" %}

{% load i18n %}

{% block content %}

<div class="container mt-4">

    <h1 class="mb-4 text-right">{{ title }}</h1>

    <div class="card shadow-sm">

        <div class="card-header bg-primary text-white text-right">

            {% if object %}

            {% trans "ویرایش قانون دسترسی" %}

            {% else %}

            {% trans "ایجاد قانون دسترسی جدید" %}

            {% endif %}

        </div>

        <div class="card-body">
```



```

        <!-- راهنما -->
        <div class="alert alert-info text-right"
role="alert">
            <h4 class="alert-heading">{% trans "
راهنمای ثبت "
قانون دسترسی" %}</h4>
            <p>{% trans "
این فرم برای تعریف قوانین دسترسی در "
سیستم طراحی شده است" %}</p>
            <!-- بقیه راهنما -->
        </div>
        <!-- فرم -->
        <form method="post">
            {% csrf_token %}
            <div class="row">
                {% for field in form %}
                <div class="col-md-6 mb-3">
                    <label for="{{ field.id_for_label }}"
class="form-label">{{ field.label }}</label>
                    {{ field }}
                    {% if field.errors %}
                    <div class="text-danger">
                        {% for error in field.errors %}
                        <small>{{ error }}</small><br>
                        {% endfor %}
                    </div>
                    {% endif %}
                </div>
                {% endfor %}
            </div>
            <div class="text-right">
                <button type="submit" class="btn btn-
primary">
                    <i class="fas fa-save"></i> {% trans
"ذخیره" %}
                </button>
                <a href="{% url 'accessrule_list' %}"
class="btn btn-secondary">
                    <i class="fas fa-arrow-right"></i> {%
trans "لغو" %}
                </a>
            </div>
        </form>
    
```

```
</form>

</div>

</div>

</div>

{% endblock %}
```

- راهنما قبل از فرم قرار گرفته تا کاربر اول بفهمه چی باید پر کنه.
- ویژگی‌ها:
- زبان ساده: برای کاربران غیرفنی قابل‌فهمه.
- سازگار با RTL: با بوت‌استرپ و فارسی همخونی داره.
- مثال کاربردی: مثل "مدیران مالی شعبه تهران" که به سناریوی درختی ربط داره.
- تمرکز بر هلدینگ: اشاره به مدیران ارشد برای امضای پرداخت‌ها، که با نیاز هلدینگ (رصد همه شعبه‌ها) جور درمیاره.

آماده‌سازی برای بحث سازمان و تیک‌های جدید

برای اینکه بریم سراغ سازمان و تیک‌های جدید توی تمپلیت، لطفاً چندتا جزئیات بدید:

- **تیک‌های جدید:**
 - دقیقاً چه تیک‌هایی (checkboxها) اضافه شدن؟ مثلاً برای فیلدهای `is_core`, `is_holding`, یا چیز دیگه؟
 - توی کدوم تمپلیت (مثل `organization_list.html` یا `accessrule_form.html`) هستن؟
 - آیا تیک‌ها برای فیلتر کردن لیست‌ها (مثل فقط هلدینگ‌ها) یا توی فرم (مثل فرم سازمان) هستن؟
- **تغییرات سازمان:**
 - آیا فیلد جدیدی به مدل `Organization` اضافه شده که باید توی راهنما یا تمپلیت لحاظ کنیم؟
 - آیا نیازه راهنمای مشابهی برای فرم `OrganizationForm` بنویسم؟
- **تمپلیت خاص:**
 - کدوم تمپلیت رو مدنظر تونه برای تیک‌ها؟ اگه کد تغییرات رو بفرستید، می‌تونم دقیق‌تر اصلاحش کنم.
- **نیازهای هلدینگ:**
 - آیا تیک‌ها برای مدیریت دسترسی هلدینگ (مثل دیدن همه شعبه‌ها) طراحی شدن؟ یا برای چیز دیگه (مثل فیلتر داده‌ها)؟

پیشنهاد اولیه برای تیک‌های جدید

تا وقتی جزئیات رو بفرستید، یه پیشنهاد عمومی برای تیک‌های جدید توی تمپلیت‌ها می‌دم:

- **نوی** organization_list.html: اضافه کردن checkbox برای فیلتر کردن سازمان‌ها بر اساس is_holding یا is_core

html

```
<div class="col-md-3">

    <input type="checkbox" name="is_holding" value="true" {% if
request.GET.is_holding %}>checked{% endif %} class="form-check-input"
id="is_holding">

    <label for="is_holding">{% trans "فقط هلدینگها" %}</label>

</div>

<div class="col-md-3">

    <input type="checkbox" name="is_core" value="true" {% if
request.GET.is_core %}>checked{% endif %} class="form-check-input"
id="is_core">

    <label for="is_core">{% trans "فقط دفاتر مرکزی" %}</label>

</div>
```

- `OrganizationListView`

python

```
is_holding = self.request.GET.get('is_holding', '')
is_core = self.request.GET.get('is_core', '')
if is_holding:
    queryset = queryset.filter(is_holding=True)
if is_core:
    queryset = queryset.filter(is_core=True)
```

- **توی** accessrule_form.html: اگر تیک‌های جدیدی به فرم اضافه شده (مثل یه فیلد بولین جدید)، می‌تونیم راهنما رو **آپدیت** کنیم:

html

```
<li class="list-group-item">
    <strong>{% trans "فیلد جدید" %}</strong>: {% trans "توضیح دهید که" %}
    {% trans "این تیک چه کاری انجام می‌دهد (مثل فعال کردن به ویژگی خاص)" %}
</li>
```

پاسخ نهایی

فیلد `entity_type` توی فرم قوانین دسترسی مشخص می‌کنه که قانون برای چه چیزی (مثل فاکتور یا دستور پرداخت) اعمال می‌شه. راهنمای کاربر که بالا نوشتم، به زبان ساده همه فیلدهای فرم `AccessRule` (از جمله `entity_type`) رو توضیح می‌ده و برای کاربران غیرفنی قابل‌فهمه. این راهنما رو می‌تونید توی `accessrule_form.html` قبل از فرم اضافه کنید تا کاربران بدونن هر فیلد چی کار می‌کنه.

برای تیک‌های جدید و تغییرات سازمان:

- لطفاً بگید چه تیک‌هایی اضافه شدن، توی کدوم تمپلیت، و برای چه هدفی (مثل فیلتر یا فرم).
- اگه تغییرات جدیدی توی مدل `Organization` یا تمپلیت‌ها دارید، بفرستید تا راهنما و تغییرات رو دقیق‌تر کنم.
- پیشنهاد دادم `checkbox` برای فیلتر `is_holding` و `is_core` توی `organization_list.html` اضافه کنیم.

همه چت‌ها رو نگه می‌دارم. اگه سؤال یا جزئیات دیگه‌ای دارید، بگید تا سریع پیش ببریم!

تاریخ و ساعت: ۱۸ می ۲۰۲۵، ۱۱:۱۸ صبح (PDT)

تهیه‌کننده: Grok 3 (xAI)

با تشکر از شما برای به اشتراک گذاشتن کد تمپلیت و URL ها! خطایی که به `accessrule_detail` اشاره داره، احتمالاً به خاطر یه مشکل توی تمپلیت یا تنظیمات مربوط به URL ها یا ویوهاست. بیایم خطا رو دقیق بررسی کنیم و مشکل رو حل کنیم. چون متن دقیق خطا رو نفرستادید، فرض می‌کنم خطا مربوط به ارجاع به `URL` `accessrule_detail` توی تمپلیت `accessrule_list.html`، مثلاً یه خطای `NoReverseMatch` یا مشکل مشابه که می‌گه `URL` پیدا نشده یا آرگومان `pk` درست پاس داده نشده.

تحلیل مشکل

1. تمپلیت `accessrule_list.html`

توی تمپلیت، شما از `{% url 'accessrule_detail' rule.pk %}` برای لینک به صفحه جزئیات قانون دسترسی استفاده کردید:

```
html
<a href="{% url 'accessrule_detail' rule.pk %}" class="btn btn-info btn-sm"
data-toggle="tooltip" title='{% trans "جزئیات" %}'>

    <i class="fas fa-eye"></i>

</a>
```

• چک اولیه:

- `rule.pk` باید یه مقدار معتبر (مثل یه عدد صحیح) باشه تا `URL` درست ساخته بشه.
- اگه `rule.pk` خالی یا نامعتبر باشه، خطای `NoReverseMatch` می‌تونه رخ بده.
- شرط `{% if rule.pk %}` توی تمپلیت اضافه شده که از ارجاع به `pk` نامعتبر جلوگیری می‌کنه، پس این بعید به نظر می‌رسه مشکل باشه.

2. URL ها

توی `urlpatterns`، مسیر `accessrule_detail` به این شکل تعریف شده:

```
python
path('access-rules/<int:pk>/', AccessRuleDetailView.as_view(),
name='accessrule_detail'),
```

• چک اولیه:

- مسیر درست تعریف شده و انتظار یه `<int:pk>` داره، که با `rule.pk` توی تمپلیت مطابقت داره.
- نام `accessrule_detail` درست نوشته شده و با تمپلیت همخونی داره.
- اگه خطا `NoReverseMatch` باشه، ممکنه مشکل از جای دیگه‌ای توی پروژه باشه (مثل نام‌های `URL` تکراری یا خطای تنظیمات).

3. `AccessRuleDetailView` ویو

ویو مربوط به `accessrule_detail` اینه:

```
python
class AccessRuleDetailView(PermissionBaseView, DetailView):
    model = AccessRule
```

```

template_name = 'core/accessrule_detail.html'
context_object_name = 'access_rule'
permission_codenames = ['core.AccessRule_view']
check_organization = True

def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    context['title'] = _('جزئیات قانون دسترسی') + f" - {self.object.organization}"
    return context

```

چک اولیه:

- ویو از `PermissionBaseView` ارث می‌برد و تنظیماتش (مثل `model` و `template_name`) درست به نظر می‌رسد.
- `check_organization = True` یعنی ویو دسترسی کاربر به سازمان قانون رو چک می‌کند. اگر کاربر به سازمان مربوط به قانون (`rule.organization`) دسترسی نداشته باشد، ممکنه خطای `PermissionDenied` یا مشکل مشابه بده.

4. خطاهای محتمل

با توجه به اطلاعات، چندتا سناریوی ممکن برای خطا وجود داره:

- خطای `NoReverseMatch`:
 • اگر `URL` `accessrule_detail` درست تعریف نشده باشه یا توی یه فایل `urls.py` دیگه بازنویسی شده باشه.
 • یا اگر `rule.pk` مقدار معتبر نداشته باشه (که بعیده چون شرط `{% if rule.pk %}` وجود داره).
- خطای `PermissionDenied`:
 • اگر کاربر لاگین کرده به سازمان مربوط به قانون دسترسی نداشته باشه (چون `check_organization = True`).
 • مثلاً، اگر قانون برای سازمان `FIN-TEHRAN` باشه و کاربر فقط به `FIN-SHIRAZ` دسترسی داشته باشه.
- خطای رندر تمپلیت:
 • اگر متغیر `access_rules` توی ویو درست پاس داده نشده باشه یا یه مشکل توی حلقه `{% for rule in access_rules %}` وجود داشته باشه.
- مشکل تنظیمات پروژه:
 • اگر فایل `urls.py` درست به `settings.py` وصل نشده باشه یا مسیرهای `URL` به درستی وارد نشده باشن.

راه‌حل‌های پیشنهادی

برای رفع خطا، این قدم‌ها رو پیشنهاد می‌کنم:

1. چک کردن خطای دقیق

- لطفاً متن دقیق خطا (مثل Traceback کامل) رو بفرستید تا بتوانم مشکل رو دقیق‌تر شناسایی کنم.
- اگر خطا NoReverseMatch، باید مطمئن بشیم که URL accessrule_detail درست تعریف شده و هیچ تداخلی با URL های دیگه نداره.

2. چک کردن urls.py

- مطمئن بشید که urlpatterns شما توی فایل اصلی urls.py پروژه درست وارد شده. مثلاً:

python

```
# project/urls.py
from django.urls import path, include

urlpatterns = [
    path('', include('core.urls')), # درست وارد core.urls مطمئن بشید
    # بقیه مسیرها
]
```

- چک کنید که هیچ URL دیگه‌ای با نام accessrule_detail توی پروژه تعریف نشده باشه:

bash

```
grep -r "accessrule_detail" project/
```

- اگر چندتا اپلیکیشن دارید، مطمئن بشید مسیرهای core.urls با بقیه اپ‌ها تداخل نداره.

3. تست دستی URL

- مرورگر رو باز کنید و URL رو دستی تست کنید:

مثلاً: `/http://localhost:8000/access-rules/1`

- اگر قانون با `pk=1` وجود داشته باشه و ویو درست کار کنه، باید صفحه جزئیات باز بشه.
- اگر خطا می‌ده (مثل ۴۰۴ یا ۴۰۳)، مشکل رو گزارش کنید.

4. رفع مشکل PermissionDenied

- چون `check_organization = True` توی `AccessRuleDetailView` فعاله، ممکنه خطا به خاطر دسترسی کاربر باشه.
- برای تست، موقتاً `check_organization` رو غیرفعال کنید:

python

```
class AccessRuleDetailView(PermissionBaseView, DetailView):
    model = AccessRule
    template_name = 'core/accessrule_detail.html'
    context_object_name = 'access_rule'
```

```

permission_codenames = ['core.AccessRule_view']
check_organization = False # موقتاً غیرفعال

def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    context['title'] = _('جزئیات قانون دسترسی') + f" - {self.object.organization}"
    return context

```

- اگر با این تغییر خطا رفع شد، مشکل از دسترسی کاربر به سازمانه. باید مطمئن بشید که:
- کاربر لاگین کرده به سازمان قانون (مثل FIN-TEHRAN) دسترسی داره.
- یا کاربر به پست توی هلدینگ (org_type=HQ) داره که به همه سازمان‌ها دسترسی می‌ده.
- برای تست دسترسی، به کاربر با دسترسی هلدینگ بسازید:

python

```

# از طریق shell
from django.contrib.auth.models import User
from core.models import Organization, OrganizationType, Post, UserPost
hq_type = OrganizationType.objects.create(fname='HQ', org_type='HQ')
holding = Organization.objects.create(code='HOLDING', name='هلدینگ',
org_type=hq_type, is_holding=True)
post = Post.objects.create(organization=holding, branch='FIN', level=5,
is_payment_order_signer=True)
user = User.objects.create_user(username='testadmin',
password='test123')
UserPost.objects.create(user=user, post=post, is_active=True)

```

- با این کاربر (testadmin) لاگین کنید و لیست قوانین رو تست کنید.

5. access_rules چک کردن

- مطمئن بشید که متغیر access_rules توی AccessRuleListView درست مقداردهی شده. ویو شما اینه:

python

```

class AccessRuleListView(PermissionBaseView, ListView):
    model = AccessRule
    template_name = 'core/accessrule_list.html'
    context_object_name = 'access_rules'
    paginate_by = 10
    permission_codenames = ['core.AccessRule_view']
    check_organization = True
    extra_context = {'title': _('لیست قوانین دسترسی')}

```



```

def get_queryset(self):
    queryset =
super().get_queryset().select_related('organization',
'stage').order_by('id')

    query = self.request.GET.get('q', '')
    is_active = self.request.GET.get('is_active', '')

    if query:
        queryset = queryset.filter(
            Q(organization__name__icontains=query) |
            Q(branch__icontains=query) |
            Q(entity_type__icontains=query)
        )

    if is_active:
        queryset = queryset.filter(is_active=(is_active == 'true'))

    logger.info(f"AccessRule queryset count: {queryset.count()}")
    return queryset

def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    context['query'] = self.request.GET.get('q', '')
    context['is_active'] = self.request.GET.get('is_active', '')
    context['total_access_rules'] = self.get_queryset().count()
    return context

```

• چک‌ها:

• لاگ‌ها رو چک کنید که `queryset.count()` درست کار می‌کنه:

`bash`

یا هر فایل لاگ دیگه ای `# tail -f logs/django.log`

• مطمئن بشید که دیتابیس حداقل به قانون دسترسی داره:

`python`

`# از طریق shell`

`from core.models import AccessRule, Organization, WorkflowStage`

```

org = Organization.objects.first()
stage = WorkflowStage.objects.first()
AccessRule.objects.create(
    organization=org,
    branch='FIN',
    min_level=3,
    stage=stage,
    action_type='APPROVE',
    entity_type='FACTOR',
    is_active=True
)

```

- اگرچه `access_rules` خالی، ممکنه کاربر به هیچ قانونی دسترسی نداشته باشه (به خاطر `check_organization = False` موقتاً برای تست، موقتاً `check_organization = False` کنید).

6. رفع مشکل توی تمپلیت

- به مشکل احتمالی اینه که `rule.entity_type` توی ستون "نوع موجودیت" به صورت خام نمایش داده می شه:

html

```
<td>{{ rule.entity_type }}</td>
```

- اگرچه `entity_type` از `choices` استفاده می کنه (مثل `FACTOR` یا `PAYMENTORDER`)، باید از `get_entity_type_display` استفاده کنید:

html

```
<td>{{ rule.get_entity_type_display }}</td>
```

- این تغییر باعث می شه به جای `"FACTOR"`، عبارت خوانا (مثل "فاکتور") نمایش داده بشه.
- تمپلیت اصلاح شده برای ستون `entity_type`:

html

```

<tr>
    <td>{{ forloop.counter }}</td>
    <td>{{ rule.organization }}</td>
    <td>{{ rule.get_branch_display|default:_("نامشخص") }}</td>
    <td>{{ rule.min_level }}</td>
    <td>{{ rule.stage }}</td>
    <td>{{ rule.get_action_type_display }}</td>
    <td>{{ rule.get_entity_type_display }}</td> <!-- اصلاح شده -->
    <td>{{ rule.is_payment_order_signer|yesno:_("بله, خیر") }}</td>

```

```

<td>
    {% if rule.is_active %}
    <span class="badge bg-success">{% trans "فعال" %}</span>
    {% else %}
    <span class="badge bg-danger">{% trans "غیرفعال" %}</span>
    {% endif %}
</td>
<td class="text-center text-nowrap">
    {% if rule.pk %}
    <a href="{% url 'accessrule_detail' rule.pk %}" class="btn btn-info btn-sm" data-toggle="tooltip" title='{% trans "جزئیات" %}'>
        <i class="fas fa-eye"></i>
    </a>
    <a href="{% url 'accessrule_update' rule.pk %}" class="btn btn-warning btn-sm mr-1" data-toggle="tooltip" title='{% trans "ویرایش" %}'>
        <i class="fas fa-edit"></i>
    </a>
    <a href="{% url 'accessrule_delete' rule.pk %}" class="btn btn-danger btn-sm mr-1" data-toggle="tooltip" title='{% trans "حذف" %}'>
        <i class="fas fa-trash"></i>
    </a>
    {% else %}
    <span class="text-muted small">{% trans "نامعتبر" %}</span>
    {% endif %}
</td>
</tr>

```

7. entity_type اضافه کردن فیلتر برای

- همونطور که توی چت قبلی پیشنهاد دادم، می‌تونیم به فیلتر برای entity_type به فرم جستجو اضافه کنیم تا کاربر بتونه قوانین رو بر اساس نوع موجودیت (مثل فقط فاکتورها) فیلتر کنه:

html

```

<div class="form-group col-md-3">
    <label for="id_entity_type">{% trans "نوع موجودیت" %}</label>
    <select name="entity_type" id="id_entity_type" class="form-control">
        <option value="">{% trans "همه" %}</option>
        <option value="FACTOR" {% if request.GET.entity_type == 'FACTOR' %}selected{% endif %}>{% trans "فاکتور" %}</option>
    </select>
</div>

```

```

        <option value="PAYMENTORDER" {% if request.GET.entity_type ==
'PAYMENTORDER' %}>selected{% endif %}>{% trans "دستور پرداخت"
%}</option>

        <option value="TANKHAH" {% if request.GET.entity_type ==
'TANKHAH' %}>selected{% endif %}>{% trans "تنخواہ" %}</option>

    </select>
</div>

```

• **توی AccessRuleListView، فیلتر رو به queryset اضافه کنید:**

python

```

def get_queryset(self):
    queryset = super().get_queryset().select_related('organization',
'stage').order_by('id')
    query = self.request.GET.get('q', '')
    is_active = self.request.GET.get('is_active', '')
    entity_type = self.request.GET.get('entity_type', '')

    if query:
        queryset = queryset.filter(
            Q(organization__name__icontains=query) |
            Q(branch__icontains=query) |
            Q(entity_type__icontains=query)
        )

    if is_active:
        queryset = queryset.filter(is_active=(is_active == 'true'))

    if entity_type:
        queryset = queryset.filter(entity_type=entity_type)

    logger.info(f"AccessRule queryset count: {queryset.count()}")
    return queryset

```

• **توی get_context_data، مقدار entity_type رو به کنتکست اضافه کنید:**

python

```

def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    context['query'] = self.request.GET.get('q', '')

```

```

context['is_active'] = self.request.GET.get('is_active', '')
context['entity_type'] = self.request.GET.get('entity_type', '')
context['total_access_rules'] = self.get_queryset().count()
return context

```

8. دیباگ با لاگ

- برای پیدا کردن مشکل، لاگ بیشتری به ویو اضافه کنید:

python

```

def get_queryset(self):
    queryset = super().get_queryset().select_related('organization',
    'stage').order_by('id')

    query = self.request.GET.get('q', '')
    is_active = self.request.GET.get('is_active', '')

    logger.info(f"Query: {query}, Is Active: {is_active}")

    logger.info(f"User: {self.request.user}, Organizations:
    {self.request.user.userpost_set.filter(is_active=True).values('post__or
    ganization__name')}")

    logger.info(f"Queryset count: {queryset.count()}")

    return queryset

```

- لاگ‌ها رو چک کنید تا ببینید کاربر چه سازمان‌هایی رو می‌تونه ببینه و کوئری‌ست چطور ساخته می‌شه.

entity_type **تمپلیت اصلاح شده با فیلتر**

اینجا تمپلیت کامل با اضافه شدن فیلتر entity_type و اصلاحات پیشنهادی آورده شده:

html

```

{% extends "base.html" %}
{% load i18n %}
{% load humanize %}
{% load static %}

{% block content %}
<div class="container mt-4">
    <h1 class="mb-4 text-right">{{ title }}</h1>

```

```

<div class="card shadow-sm mb-4">
    <div class="card-header bg-light text-dark text-right">
        <h5 class="my-1">{% trans "جستجو و فیلتر" %}</h5>
    </div>
    <div class="card-body">
        <form method="get" action="{% url 'accessrule_list' %}"
class="form-row align-items-end text-right">
            <div class="form-group col-md-3">
                <label for="id_q">{% trans "جستجو در سازمان، شاخه یا نوع " موجودیت
" %}</label>
                <input type="text" name="q" id="id_q" class="form-
control" value="{{ query|default:'' }}" placeholder='{% trans "عبارت
جستجو..." %}'>
            </div>
            <div class="form-group col-md-3">
                <label for="id_is_active">{% trans "وضعیت" %}</label>
                <select name="is_active" id="id_is_active" class="form-
control">
                    <option value="">{% trans "همه" %}</option>
                    <option value="true" {% if is_active == 'true'
%}>selected{% endif %}>{% trans "فعال" %}</option>
                    <option value="false" {% if is_active == 'false'
%}>selected{% endif %}>{% trans "غیرفعال" %}</option>
                </select>
            </div>
            <div class="form-group col-md-3">
                <label for="id_entity_type">{% trans "نوع موجودیت"
%}</label>
                <select name="entity_type" id="id_entity_type"
class="form-control">
                    <option value="">{% trans "همه" %}</option>
                    <option value="FACTOR" {% if entity_type == 'FACTOR'
%}>selected{% endif %}>{% trans "فاکتور" %}</option>
                    <option value="PAYMENTORDER" {% if entity_type ==
'PAYMENTORDER' %}>selected{% endif %}>{% trans "دستور پرداخت" %}</option>
                    <option value="TANKHAH" {% if entity_type ==
'TANKHAH' %}>selected{% endif %}>{% trans "تنخواه" %}</option>
                </select>
            </div>
            <div class="form-group col-md-3">
                <button type="submit" class="btn btn-primary">

```

[illegible]

```

<th>{% trans "سازمان" %}</th>
<th>{% trans "شاخه" %}</th>
<th>{% trans "حداقل سطح" %}</th>
<th>{% trans "مرحله" %}</th>
<th>{% trans "نوع اقدام" %}</th>
<th>{% trans "نوع موجودیت" %}</th>
<th>{% trans "امضاکننده دستور پرداخت" %}</th>
<th>{% trans "وضعیت" %}</th>
<th class="text-center">{% trans "عملیات"
%}</th>

</tr>
</thead>
<tbody>
    {% for rule in access_rules %}
    <tr>
        <td>{{ forloop.counter }}</td>
        <td>{{ rule.organization }}</td>
        <td>{{
rule.get_branch_display|default:_("نامشخص") }}</td>
        <td>{{ rule.min_level }}</td>
        <td>{{ rule.stage }}</td>
        <td>{{ rule.get_action_type_display }}</td>
        <td>{{ rule.get_entity_type_display }}</td>
        <td>{{
rule.is_payment_order_signer|yesno:_("بله,خیر") }}</td>
        <td>
            {% if rule.is_active %}
            <span class="badge bg-success">{% trans
"فعال" %}</span>

            {% else %}
            <span class="badge bg-danger">{% trans
"غیرفعال" %}</span>

            {% endif %}
        </td>
        <td class="text-center text-nowrap">
            {% if rule.pk %}
            <a href="{% url 'accessrule_detail'
rule.pk %}" class="btn btn-info btn-sm" data-toggle="tooltip" title='{% trans
"جزئیات" %}'>

                <i class="fas fa-eye"></i>

```



```

        </a>

        <a href="{% url 'accessrule_update'
rule.pk %}" class="btn btn-warning btn-sm mr-1" data-toggle="tooltip"
title='{% trans "ویرایش" %}'>

            <i class="fas fa-edit"></i>

        </a>

        <a href="{% url 'accessrule_delete'
rule.pk %}" class="btn btn-danger btn-sm mr-1" data-toggle="tooltip"
title='{% trans "حذف" %}'>

            <i class="fas fa-trash"></i>

        </a>

        {% else %}

        <span class="text-muted small">{% trans
"نامعتبر" %}</span>

        {% endif %}

    </td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

{% if is_paginated %}

<nav aria-label="Page navigation" class="mt-4">

    <ul class="pagination justify-content-center">

        {% if page_obj.has_previous %}

        <li class="page-item">

            <a class="page-link" href="?page={{
page_obj.previous_page_number }}{% if query %}&q={{ query }}{% endif %}{% if
is_active %}&is_active={{ is_active }}{% endif %}{% if entity_type
%}&entity_type={{ entity_type }}{% endif %}" aria-label='{% trans "قبلی" %}'>

                <span aria-hidden="true"><<</span>

                <span class="sr-only">{% trans "قبلی"
%}</span>

            </a>

        </li>

        {% else %}

        <li class="page-item disabled">

            <span class="page-link" aria-
hidden="true"><<</span>

        </li>

```

```

        {% endif %}

        {% for i in paginator.page_range %}
            {% if page_obj.number == i %}
                <li class="page-item active"><span class="page-
link">{{ i }} <span class="sr-only">(current)</span></span></li>
            {% else %}
                <li class="page-item"><a class="page-link"
href="?page={{ i }}" {% if query %}&q={{ query }}{% endif %}{% if is_active
%}&is_active={{ is_active }}{% endif %}{% if entity_type %}&entity_type={{
entity_type }}{% endif %}">{{ i }}</a></li>
            {% endif %}
        {% endfor %}

        {% if page_obj.has_next %}
            <li class="page-item">
                <a class="page-link" href="?page={{
page_obj.next_page_number }}" {% if query %}&q={{ query }}{% endif %}{% if
is_active %}&is_active={{ is_active }}{% endif %}{% if entity_type
%}&entity_type={{ entity_type }}{% endif %}" aria-label='{% trans "بعدي" %}'>
                    <span aria-hidden="true">></span>
                    <span class="sr-only">{% trans "بعدي"
%}</span>
                </a>
            </li>
            {% else %}
                <li class="page-item disabled">
                    <span class="page-link" aria-
hidden="true">></span>
                </li>
            {% endif %}
        </ul>
    </nav>
{% endif %}

{% else %}
<div class="alert alert-info text-center mb-0" role="alert">
    {% trans "هیچ قانون دسترسی یافت نشد" %}
</div>
{% endif %}

```

```

        </div>
    </div>

</div>
{% endblock %}

{% block extra_js %}
{{ block.super }}
<script>
    $(function () {
        $('[data-toggle="tooltip"]').tooltip()
    });
</script>
{% endblock extra_js %}

```

اطلاعات مورد نیاز برای رفع کامل

برای اینکه دقیق‌تر مشکل رو حل کنیم، لطفاً این اطلاعات رو بفرستید:

- متن دقیق خطا:
 - `Traceback` کامل خطا رو بفرستید (مثلاً `/NoReverseMatch at /access-rules` یا `(PermissionDenied)`.)
 - آگه خطا توی مرورگر نمایش داده می‌شه، متن کاملش رو کپی کنید.
- وضعیت دیتابیس:
 - آیا توی دیتابیس قانون دسترسی (`AccessRule`) دارید؟ با این دستور چک کنید:

python

```

from core.models import AccessRule
AccessRule.objects.count()

```

- آگه خالیه، به قانون نمونه بسازید (مثل کد بالا).
- وضعیت کاربر:
 - کاربر لاگین کرده چه سازمان‌هایی رو می‌تونه ببینه؟ با این دستور چک کنید:

python

```

from django.contrib.auth.models import User

```

```
user = User.objects.get(username='your_username')
user.userpost_set.filter(is_active=True).values('post__organization__name')
```

- آیا کاربر به سازمان‌های قوانین دسترسی دارد؟
- تنظیمات پروژه:
 - آیا فایل `core/urls.py` درست به `project/urls.py` وصل شده؟ محتوای `project/urls.py` رو بفرستید.
 - آیا اپلیکیشن `core` توی `INSTALLED_APPS` توی `settings.py` اضافه شده؟
- تمپلیت `:accessrule_detail.html`
 - محتوای `core/accessrule_detail.html` رو بفرستید تا مطمئن بشیم درست تنظیم شده.
- چک مدل `:AccessRule`
 - مدل `AccessRule` رو کامل بفرستید (مخصوصاً تعریف `entity_type` و `choices`) تا مطمئن بشیم `get_entity_type_display` درست کار می‌کنه.

پاسخ نهایی

خطای مربوط به `accessrule_detail` احتمالاً به خاطر یکی از اینهاست:

- مشکل دسترسی کاربر (به خاطر `(check_organization = True)`).
- خطای `NoReverseMatch` به خاطر تداخل URL ها یا تنظیمات نادرست `.urls.py`.
- مشکل توی دیتابیس یا متغیر `.access_rules`.

اقدامات انجام‌شده:

- ستون `entity_type` توی تمپلیت به `get_entity_type_display` اصلاح شد.
- فیلتر `entity_type` به فرم جستجو اضافه شد.
- تمپلیت با حفظ پارامترهای فیلتر `(entity_type, is_active, query)` آپدیت شد.
- پیشنهادهایی برای دیباگ (چک دیتابیس، کاربر، URL ها) داده شد.

قدم بعدی:

- لطفاً متن دقیق خطا و اطلاعات درخواستی (دیتابیس، کاربر، `urls.py`) رو بفرستید تا مشکل رو ریشه‌ای حل کنیم.
- موقتاً `check_organization = False` توی `AccessRuleDetailView` تست کنید.
- اگه خطا ادامه داشت، یه قانون نمونه بسازید و URL رو دستی تست کنید.

همه چت‌ها رو نگه می‌دارم. اگه سؤال یا جزئیات دیگه‌ای دارید، بگید تا سریع رفعش کنیم!

تاریخ و ساعت: ۱۸ می ۲۰۲۵، ۱۱:۳۹ صبح (PDT)
تهیه‌کننده: Grok 3 (xAI)