Chrono - Development #54994

perf /datetimes de rendez-vous sur un grand nombre de guichets (genre 30) et périodes d'ouverture et exceptions

18 juin 2021 15:13 - Frédéric Péters

Statut:	Fermé	**Début:**	18 juin 2021
Priorité:	Normal	**Echéance:**	
Assigné à:		**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	Non

Description

Des guichets qui ont des horaires d'ouverture genre lundi 9h->10h30, puis 11h->12h30, 13h->14h30, 15h->16h30, 17h→18h30, et merdi les mêmes et mercredi et jeudi etc. Sur tous les guichets les mêmes exceptions, typiquement une par jour (exceptions qui retirent chaque fois un créneau).

Un type de rendez-vous, 90 minutes. Et une réservation max à 16 jours.

Ça produit, au moins quand presque tous les créneaux sont réservés, un retour de /datetimes trop lent. (de l'ordre de 35 secondes pour le cas précis rencontré).

Révisions associées

Révision 4d9c0330 - 19 juin 2021 07:54 - Frédéric Péters

api: prefetch exception desk when getting available slots (#54994)

Historique

#1 - 18 juin 2021 17:03 - Benjamin Dauvergne

~~Combien de guichets ? Dans les deux derniers tickets du genre (#42169 et #42142) on avait la même mais sur un agenda virtuel avec beaucoup d'agendas, mais je ne sais plus combien.~~ Apprendre à lire le titre.

#2 - 18 juin 2021 17:05 - Benjamin Dauvergne

C'est bizarre parce que ça ne fait pas beaucoup de créneaux, avec des RdV de 90 minutes et des plages d'autant ça fait juste 5 créneaux par jour et par guichets donc dans les 12j ouvrables * 5 créneaux * 30 guichets = 1800 créneaux, ça devrait passer.

#3 - 18 juin 2021 17:17 - Frédéric Péters

D'un test local, log SQL sur un appel /datetimes :

```
2021-06-18 17:12:43 CEST [2160423-252484] fred@chrono-test- LOG:  statement: SELECT "agendas_desk"."id", "agen
das_desk"."agenda_id", "agendas_desk"."label", "agendas_desk"."slug" FROM "agendas_desk" WHERE "agendas_desk".
"id" = 1
2021-06-18 17:12:43 CEST [2160423-252485] fred@chrono-test- LOG:  statement: SELECT "agendas_desk"."id", "agen
das_desk"."agenda_id", "agendas_desk"."label", "agendas_desk"."slug" FROM "agendas_desk" WHERE "agendas_desk".
"id" = 1
2021-06-18 17:12:43 CEST [2160423-252486] fred@chrono-test- LOG:  statement: SELECT "agendas_desk"."id", "agen
das_desk"."agenda_id", "agendas_desk"."label", "agendas_desk"."slug" FROM "agendas_desk" WHERE "agendas_desk".
"id" = 1
2021-06-18 17:12:43 CEST [2160423-252487] fred@chrono-test- LOG:  statement: SELECT "agendas_desk"."id", "agen
das_desk"."agenda_id", "agendas_desk"."label", "agendas_desk"."slug" FROM "agendas_desk" WHERE "agendas_desk".
"id" = 1
2021-06-18 17:12:43 CEST [2160423-252488] fred@chrono-test- LOG:  statement: SELECT "agendas_desk"."id", "agen
das_desk"."agenda_id", "agendas_desk"."label", "agendas_desk"."slug" FROM "agendas_desk" WHERE "agendas_desk".
"id" = 1
2021-06-18 17:12:43 CEST [2160423-252489] fred@chrono-test- LOG:  statement: SELECT "agendas_desk"."id", "agen
das_desk"."agenda_id", "agendas_desk"."label", "agendas_desk"."slug" FROM "agendas_desk" WHERE "agendas_desk".
"id" = 1
2021-06-18 17:12:43 CEST [2160423-252490] fred@chrono-test- LOG:  statement: SELECT "agendas_desk"."id", "agen
das_desk"."agenda_id", "agendas_desk"."label", "agendas_desk"."slug" FROM "agendas_desk" WHERE "agendas_desk".
"id" = 1
2021-06-18 17:12:43 CEST [2160423-252491] fred@chrono-test- LOG:  statement: SELECT "agendas_desk"."id", "agen
das_desk"."agenda_id", "agendas_desk"."label", "agendas_desk"."slug" FROM "agendas_desk" WHERE "agendas_desk".
"id" = 1
```

```
2021-06-18 17:12:43 CEST [2160423-252492] fred@chrono-test- LOG:  statement: SELECT "agendas_desk"."id", "agendas_desk"."agenda_id", "agendas_desk"."label", "agendas_desk"."slug" FROM "agendas_desk" WHERE "agendas_desk"."id" = 1
2021-06-18 17:12:43 CEST [2160423-252493] fred@chrono-test- LOG:  statement: SELECT "agendas_desk"."id", "agendas_desk"."agenda_id", "agendas_desk"."label", "agendas_desk"."slug" FROM "agendas_desk" WHERE "agendas_desk"."id" = 1
2021-06-18 17:12:43 CEST [2160423-252494] fred@chrono-test- LOG:  statement: SELECT "agendas_desk"."id", "agendas_desk"."agenda_id", "agendas_desk"."label", "agendas_desk"."slug" FROM "agendas_desk" WHERE "agendas_desk"."id" = 1
2021-06-18 17:12:43 CEST [2160423-252495] fred@chrono-test- LOG:  statement: SELECT "agendas_desk"."id", "agendas_desk"."agenda_id", "agendas_desk"."label", "agendas_desk"."slug" FROM "agendas_desk" WHERE "agendas_desk"."id" = 1
2021-06-18 17:12:43 CEST [2160423-252496] fred@chrono-test- LOG:  statement: SELECT "agendas_desk"."id", "agendas_desk"."agenda_id", "agendas_desk"."label", "agendas_desk"."slug" FROM "agendas_desk" WHERE "agendas_desk"."id" = 1
2021-06-18 17:12:43 CEST [2160423-252497] fred@chrono-test- LOG:  statement: SELECT "agendas_desk"."id", "agendas_desk"."agenda_id", "agendas_desk"."label", "agendas_desk"."slug" FROM "agendas_desk" WHERE "agendas_desk"."id" = 1
2021-06-18 17:12:43 CEST [2160423-252498] fred@chrono-test- LOG:  statement: SELECT "agendas_desk"."id", "agendas_desk"."agenda_id", "agendas_desk"."label", "agendas_desk"."slug" FROM "agendas_desk" WHERE "agendas_desk"."id" = 1
2021-06-18 17:12:43 CEST [2160423-252499] fred@chrono-test- LOG:  statement: SELECT "agendas_desk"."id", "agendas_desk"."agenda_id", "agendas_desk"."label", "agendas_desk"."slug" FROM "agendas_desk" WHERE "agendas_desk"."id" = 1
2021-06-18 17:12:43 CEST [2160423-252500] fred@chrono-test- LOG:  statement: SELECT "agendas_desk"."id", "agendas_desk"."agenda_id", "agendas_desk"."label", "agendas_desk"."slug" FROM "agendas_desk" WHERE "agendas_desk"."id" = 1
2021-06-18 17:12:43 CEST [2160423-252501] fred@chrono-test- LOG:  statement: SELECT "agendas_desk"."id", "agendas_desk"."agenda_id", "agendas_desk"."label", "agendas_desk"."slug" FROM "agendas_desk" WHERE "agendas_desk"."id" = 1
2021-06-18 17:12:43 CEST [2160423-252502] fred@chrono-test- LOG:  statement: SELECT "agendas_desk"."id", "agendas_desk"."agenda_id", "agendas_desk"."label", "agendas_desk"."slug" FROM "agendas_desk" WHERE "agendas_desk"."id" = 1
2021-06-18 17:12:43 CEST [2160423-252503] fred@chrono-test- LOG:  statement: SELECT "agendas_desk"."id", "agendas_desk"."agenda_id", "agendas_desk"."label", "agendas_desk"."slug" FROM "agendas_desk" WHERE "agendas_desk"."id" = 1
2021-06-18 17:12:43 CEST [2160423-252504] fred@chrono-test- LOG:  statement: SELECT "agendas_desk"."id", "agendas_desk"."agenda_id", "agendas_desk"."label", "agendas_desk"."slug" FROM "agendas_desk" WHERE "agendas_desk"."id" = 2
2021-06-18 17:12:43 CEST [2160423-252505] fred@chrono-test- LOG:  statement: SELECT "agendas_desk"."id", "agendas_desk"."agenda_id", "agendas_desk"."label", "agendas_desk"."slug" FROM "agendas_desk" WHERE "agendas_desk"."id" = 2
[...]
```

pour référence le code du test pour arriver au pdb d'où j'ai lancé l'appel et regardé les logs :

```
@pytest.mark.freeze_time('2017-05-20')
def test_liege(app, user):
    app.authorization = ('Basic', ('john.doe', 'password'))

    agenda = Agenda(
        label=u'Foo bar Meeting', kind='meetings', minimal_booking_delay=0, maximal_booking_delay=16
    )
    agenda.save()
    meeting_type = MeetingType(agenda=agenda, label='Blah', duration=90)
    meeting_type.save()

    desks = {}
    for x in range(30):
        desk, created = Desk.objects.get_or_create(agenda=agenda, label='Desk %s' % x)
        desks[desk.id] = desk

    for x in range(7):
        for desk in desks.values():
            for hour_delta in range(0, 8, 2):
                time_period = TimePeriod(
                    weekday=x,
                    start_time=datetime.time(8 + hour_delta, 0),
                    end_time=datetime.time(9 + hour_delta, 30),
                    desk=desk,
                )
                time_period.save()

    api_url = '/api/agenda/%s/meetings/%s/datetimes/' % (agenda.slug, meeting_type.slug)
    resp = app.get(api_url)
```

```
    assert len(resp.json['data']) == 64

    for x in range(20):
        for desk in desks.values():
            TimePeriodException.objects.create(
                desk=desk,
                start_datetime=make_aware(datetime.datetime(2017, 5, 22, 8, 0) + datetime.timedelta(days=x)),
                end_datetime=make_aware(datetime.datetime(2017, 5, 22, 9, 30) + datetime.timedelta(days=x)),
                )

    resp = app.get(api_url)
    assert len(resp.json['data']) == 50

    for i in range(400):
        #resp = app.get(api_url)
        fillslot_url = random.choice(resp.json['data'])['api']['fillslot_url']
        resp2 = app.post(fillslot_url)
        #assert resp.json['err'] == 0

    import pdb; pdb.set_trace()
    pass
```

mais ça n'est pas lent du tout.

#4 - 19 juin 2021 08:00 - Frédéric Péters

- Fichier 0001-api-prefetch-exception-desk-when-getting-available-s.patch ajouté

- Statut changé de Nouveau à Solution proposée

- Patch proposed changé de Non à Oui

J'ai tracé le problème sur place et tout vient du temps passé sur la récup des exceptions,

```
    desks_exceptions = {
        time_period_desk: IntervalSet.from_ordered(
            map(TimePeriodException.as_interval, time_period_exceptions)
        )
        for time_period_desk, time_period_exceptions in itertools.groupby(
            TimePeriodException.objects.filter(desk__agenda__in=agendas).order_by(
                'desk_id', 'start_datetime', 'end_datetime'
            ),
            key=lambda time_period: time_period.desk,
        )
    }
```

À cause du .desk, ce qui correspond bien aux requêtes répétées notées plus haut.

En soit tout me semble pouvoir être réécrit pour juste se baser l'id du guichet mais ça marche aussi tout à fait de juste taper un select_related('desk') là.

#6 - 19 juin 2021 09:36 - Benjamin Dauvergne

- Statut changé de Solution proposée à Solution validée

#7 - 20 juin 2021 18:03 - Frédéric Péters

- Statut changé de Solution validée à Résolu (à déployer)

```
commit 4d9c0330ad01cff3fdf61a6477e1d19e980e18fd
Author: Frédéric Péters <fpeters@entrouvert.com>
Date:   Sat Jun 19 07:54:31 2021 +0200

    api: prefetch exception desk when getting available slots (#54994)
```

#8 - 22 juin 2021 09:17 - Frédéric Péters

- Statut changé de Résolu (à déployer) à Solution déployée

Fichiers

0001-api-prefetch-exception-desk-when-getting-available-s.patch	1,09 ko	19 juin 2021	Frédéric Péters