Chrono - Bug #18495

Résa multi-guichets ne fonctionne pas sur le premier jour ouvert à la réservation

06 septembre 2017 14:44 - Brice Mallet

Statut:	Fermé	**Début:**	06 septembre 2017
Priorité:	Normal	**Echéance:**	
Assigné à:	Josué Kouka	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	

Description

Vu à l'instant avec Josué, une réservation posée le premier jour de la période ouvert à la réservation n'est pas prise en compte dans le cas du multi-guichet.
https://demarches2016.alfortville.fr/backoffice/management/test-rdv-version-sept17/26/

Proposition : si cela était plus simple, il serait possible de fonctionner en jour complet i.e. si réservation à J+1, il est envisageable de décider que tous les créneaux de demain sont bloqués et non pas seulement ceux avant 14h44.

Révisions associées

Révision b7211601 - 08 septembre 2017 17:18 - Frédéric Péters

fix next day but less than 24h delta booking (#18495)

Historique

#1 - 06 septembre 2017 16:01 - Frédéric Péters

> pas prise en compte

Besoin de précision là-dessus, le créneau est affiché comme pas disponible alors qu'il l'est ? le créneau peut être choisi mais la réservation échoue ? Autre chose ?

> Proposition : si cela était plus simple, il serait possible de fonctionner en jour complet i.e. si réservation à J+1, il est envisageable de décider que tous les créneaux de demain sont bloqués et non pas seulement ceux avant 14h44.

Sauf erreur c'est le comportement présent pour les agendas d'événements, si la réservation minimale est à J+1 alors on peut réserver tous les événements du lendemain, même ceux à 10 heures du matin alors qu'il est 14h44; sans même poser la question du plus simple ou pas, avoir le même comportement pour les rendez-vous me semble souhaitable.

#2 - 06 septembre 2017 16:37 - Brice Mallet

Frédéric Péters a écrit :

> > pas prise en compte
>
> Besoin de précision là-dessus, le créneau est affiché comme pas disponible alors qu'il l'est ? le créneau peut être choisi mais la réservation échoue ? Autre chose ?

Le créneau est disponible et réservable n fois alors que , dans ce cas de 2 guichet, devrait ne plus apparaître après avoir été réservé 2 fois.
Mais donc du coup le problème est plus, il me semble : ces créneaux sont affichés alors qu'ils ne le devraient pas (sont hors des limites réservables)

> > Proposition : si cela était plus simple, il serait possible de fonctionner en jour complet i.e. si réservation à J+1, il est envisageable de décider que tous les créneaux de demain sont bloqués et non pas seulement ceux avant 14h44.
>
> Sauf erreur c'est le comportement présent pour les agendas d'événements, si la réservation minimale est à J+1 alors on peut réserver tous les événements du lendemain, même ceux à 10 heures du matin alors qu'il est 14h44; sans même poser la question du plus simple ou pas, avoir le même comportement pour les rendez-vous me semble souhaitable.

OK

#3 - 06 septembre 2017 16:42 - Frédéric Péters

Josué, le ticket t'es assigné, tu t'en occupes pour de vrai, tu as une échéance ?

#4 - 06 septembre 2017 16:52 - Frédéric Péters

> Le créneau est disponible et réservable n fois alors que , dans ce cas de 2 guichet, devrait ne plus apparaître après avoir été réservé 2 fois. Mais donc du coup le problème est plus, il me semble : ces créneaux sont affichés alors qu'ils ne le devraient pas (sont hors des limites réservables)

Ça sonne pour moi comme deux problèmes différents; d'une part un horaire trop proche ne devrait pas être réservable; mais d'autre part, si jamais il s'est trouvé réservable, le nombre de réservations ne peut pas dépasser le nombre de places.

#5 - 06 septembre 2017 16:54 - Josué Kouka

Frédéric Péters a écrit :

> Josué, le ticket t'es assigné, tu t'en occupes pour de vrai, tu as une échéance ?

Je vais m'en occuper. Dès que je finis le patch sur les exceptions (A priori d'ici 1 heure ou deux)

#6 - 06 septembre 2017 16:56 - Frédéric Péters

> Mais donc du coup le problème est plus, il me semble : ces créneaux sont affichés alors qu'ils ne le devraient pas (sont hors des limites réservables)

L'agenda https://chrono-alfortville.test.entrouvert.org/manage/agendas/11/ est configuré pour autoriser les réservations du lendemain (j+1), et le premier jour que https://demarches2016.alfortville.fr/formation/test-rdv-version-sept17/ affiche est bien le lendemain.

#7 - 06 septembre 2017 17:55 - Josué Kouka

J'ai trouvé pourquoi.

```
min_datetime = now() + datetime.timedelta(days=agenda.minimal_booking_delay)
for event in agenda.event_set.filter(
   agenda=agenda, start_datetime__gte=min_datetime,
   start_datetime__lte=max_datetime + datetime.timedelta(meeting_type.duration)).select_related(
               'meeting_type').extra(
                   select={
                       'booking_count': """SELECT COUNT(*) FROM agendas_booking
                                           WHERE agendas_booking.event_id = agendas_event.id
                                           AND agendas_booking.cancellation_datetime IS NOT NULL"""}):
    if event.booking_count:
       continue
    open_slots_by_desk[event.desk_id].remove_overlap(event.start_datetime, event.end_datetime)
```

Au moment de la recherche des résas déja effectuées faut remettre le min_datetime en debut de journée.
Je fais le patch et le test

#8 - 06 septembre 2017 18:05 - Josué Kouka

- Statut changé de Nouveau à En cours

#9 - 06 septembre 2017 19:24 - Josué Kouka

- Fichier 0001-fix-multiple-desk-booking-fetching-range-18495.patch ajouté

- Patch proposed changé de Non à Oui

#10 - 06 septembre 2017 20:31 - Josué Kouka

- Fichier 0001-fix-multiple-desk-booking-fetching-range-18495.patch ajouté

suppresion d'une ligne vide

#11 - 06 septembre 2017 23:12 - Frédéric Péters

J'ai du mal à piger, peut-être parce que la même variable se trouve utilisée avec des sens différents entre deux moments.

```
min_datetime = now() + datetime.timedelta(days=agenda.minimal_booking_delay)
```

Cette valeur est passée time_period.get_time_slots, mais cette méthode produira aussi un résultat pour demain 10h, s'il est aujourd'hui 17h et que le délai est à 1 jour ? C'est-à-dire, à 2017-09-07 17:00, min_datetime vaudra 2017-09-08 17:00, et pourtant on attend que le créneau 2017-09-08 10:00 soit ouvert; du nom du filtre je dirais qu'en toute logique ça ne le serait pas alors que ça devrait l'être.

Sur la même idée, si min_datetime = min_datetime.replace(hour=0, minute=0) a lieu dès le début, ça change quoi ?

max_datetime = min_datetime.replace(hour=23, minute=59); ça devrait plutôt marcher avec "lendemain minuit", plutôt que "aujourd'hui 23h59". (s'il y a jeu comme ça sur des minutes je t'invite à faire des tests où les créneaux durent une minute, pour pouvoir comparer).

def test_agenda_meeting_api_multiple_desk_datetime_range(app, user):; cela n'exécute pas les tests dans les conditions de timezone et horaire différents. (ajouter meetings_agenda en paramètre doit être suffisant).

> Le créneau est disponible et réservable n fois alors que , dans ce cas de 2 guichet, devrait ne plus apparaître après avoir été réservé 2 fois.
> Mais donc du coup le problème est plus, il me semble : ces créneaux sont affichés alors qu'ils ne le devraient pas (sont hors des limites réservables)

Ça sonne pour moi comme deux problèmes différents; d'une part un horaire trop proche ne devrait pas être réservable; mais d'autre part, si jamais il s'est trouvé réservable, le nombre de réservations ne peut pas dépasser le nombre de places.

Il faut commencer par faire des tests qui échouent. Là je supprime tes modifications à chrono/api/views.py et ton test continue pourtant à fonctionner.

#12 - 07 septembre 2017 00:04 - Josué Kouka

Frédéric Péters a écrit :

> J'ai du mal à piger, peut-être parce que la même variable se trouve utilisée avec des sens différents entre deux moments.
>
> [...]
>
> Cette valeur est passée time_period.get_time_slots, mais cette méthode produira aussi un résultat pour demain 10h, s'il est aujourd'hui 17h et que le délai est à 1 jour ?

oui http://git.entrouvert.org/chrono.git/tree/chrono/agendas/models.py#n176 et
http://git.entrouvert.org/chrono.git/tree/chrono/agendas/models.py#n182

> C'est-à-dire, à 2017-09-07 17:00, min_datetime vaudra 2017-09-08 17:00, et pourtant on attend que le créneau 2017-09-08 10:00 soit ouvert; du nom du filtre je dirais qu'en toute logique ça ne le serait pas alors que ça devrait l'être.

Oui.

> Sur la même idée, si min_datetime = min_datetime.replace(hour=0, minute=0) a lieu dès le début, ça change quoi ?

Pour le filtre on prend on compte toutes les résa du jour correspondant au minimal_booing_delay et non juste les résa de ce jour à partir de 17:00 (2017-09-08 17:00)

> max_datetime = min_datetime.replace(hour=23, minute=59); ça devrait plutôt marcher avec "lendemain minuit", plutôt que "aujourd'hui 23h59". (s'il y a jeu comme ça sur des minutes je t'invite à faire des tests où les créneaux durent une minute, pour pouvoir comparer).

Ok

> def test_agenda_meeting_api_multiple_desk_datetime_range(app, user):; cela n'exécute pas les tests dans les conditions de timezone et horaire différents. (ajouter meetings_agenda en paramètre doit être suffisant).

Oui ça devrait. Je ne l'ai pas fait parce que je n'ai pas pu le reproduire avec le ow() "mocké".

> > Le créneau est disponible et réservable n fois alors que , dans ce cas de 2 guichet, devrait ne plus apparaître après avoir été réservé 2 fois.
> > Mais donc du coup le problème est plus, il me semble : ces créneaux sont affichés alors qu'ils ne le devraient pas (sont hors des limites réservables)
>
> Ça sonne pour moi comme deux problèmes différents; d'une part un horaire trop proche ne devrait pas être réservable; mais d'autre part, si

jamais il s'est trouvé réservable, le nombre de réservations ne peut pas dépasser le nombre de places.

Il faut commencer par faire des tests qui échouent. Là je supprime tes modifications à chrono/api/views.py et ton test continue pourtant à fonctionner.

J'ai commencé par un test qui échoue. Résultat chez moi

```
    def test_agenda_meeting_api_multiple_desk_datetime_range(app, user):
        app.authorization = ('Basic', ('john.doe', 'password'))
        agenda = Agenda(label='Foo', kind='meetings')
        agenda.minimal_booking_delay = 2
        agenda.maximal_booking_delay = 15
        agenda.save()
        meeting_type = MeetingType.objects.create(agenda=agenda, label='Blah', duration=30)
        datetime_url = '/api/agenda/meetings/%s/datetimes/' % meeting_type.id
        desk = Desk.objects.create(label='ying', agenda=agenda)
        today = datetime.datetime.today()
        weekday = today.weekday() + 2
        time_period = TimePeriod.objects.create(
            weekday=weekday, start_time=datetime.time(10, 0), end_time=datetime.time(12, 30),
            desk=desk)
        resp = app.get(datetime_url)
        # book the first slot
        event_data = resp.json['data'][0]
        booking_url = event_data['api']['fillslot_url']
        app.post(booking_url)
        assert Booking.objects.count() == 1
        # make sure first slot is unavailable
        resp2 = app.get(datetime_url)
>       assert len(resp.json['data']) == len([x for x in resp2.json['data'] if not x['disabled']]) + 1
E       AssertionError: assert 10 == (10 + 1)
E        +  where 10 = len([{'api': {'fillslot_url': 'http://testserver/api/agenda/foo/fillslot/1:2017-09-08-1
000/'}, 'datetime': '2017-09-08 10:...lslot/1:2017-09-15-1000/'}, 'datetime': '2017-09-15 10:00:00', 'disabled
': False, 'id': '1:2017-09-15-1000', ...}, ...])
E        +  and   10 = len([{'api': {'fillslot_url': 'http://testserver/api/agenda/foo/fillslot/1:2017-09-08-1
000/'}, 'datetime': '2017-09-08 10:...lslot/1:2017-09-15-1000/'}, 'datetime': '2017-09-15 10:00:00', 'disabled
': False, 'id': '1:2017-09-15-1000', ...}, ...])
```

#13 - 07 septembre 2017 07:38 - Frédéric Péters

Chez moi, ton patch + son annulation + l'exécution du test sur différentes zones :

```
diff --git a/chrono/api/views.py b/chrono/api/views.py
index 53df293..26cb326 100644
--- a/chrono/api/views.py
+++ b/chrono/api/views.py
@@ -50,8 +50,6 @@ def get_open_slots(agenda, meeting_type):
             open_slots_by_desk[time_period.desk_id].addi(slot.start_datetime, slot.end_datetime, slot.desk)

     # set min_datetime to the beginning of the day and max_datetime to the end of the day
-    min_datetime = min_datetime.replace(hour=0, minute=0)
-    max_datetime = min_datetime.replace(hour=23, minute=59)
     for event in agenda.event_set.filter(
             agenda=agenda, start_datetime__gte=min_datetime,
             start_datetime__lte=max_datetime + datetime.timedelta(meeting_type.duration)).select_related(
diff --git a/tests/test_api.py b/tests/test_api.py
index 9f6e16d..641c07f 100644
--- a/tests/test_api.py
+++ b/tests/test_api.py
@@ -780,7 +780,7 @@ def test_agenda_meeting_api_multiple_desk(app, meetings_agenda, user):
         assert queries_count_datetime1 == len(ctx.captured_queries)

-def test_agenda_meeting_api_multiple_desk_datetime_range(app, user):
+def test_agenda_meeting_api_multiple_desk_datetime_range(app, meetings_agenda, user):
     app.authorization = ('Basic', ('john.doe', 'password'))
     agenda = Agenda(label='Foo', kind='meetings')
     agenda.minimal_booking_delay = 2

tests/test_api.py::test_agenda_meeting_api_multiple_desk_datetime_range[Europe/Brussels-mock_now0] PASSED
tests/test_api.py::test_agenda_meeting_api_multiple_desk_datetime_range[Europe/Brussels-mock_now1] PASSED
tests/test_api.py::test_agenda_meeting_api_multiple_desk_datetime_range[Europe/Brussels-mock_now2] PASSED
tests/test_api.py::test_agenda_meeting_api_multiple_desk_datetime_range[Asia/Kolkata-mock_now0] PASSED
```

```
tests/test_api.py::test_agenda_meeting_api_multiple_desk_datetime_range[Asia/Kolkata-mock_now1] PASSED
tests/test_api.py::test_agenda_meeting_api_multiple_desk_datetime_range[Asia/Kolkata-mock_now2] PASSED
tests/test_api.py::test_agenda_meeting_api_multiple_desk_datetime_range[Brazil/East-mock_now0] PASSED
tests/test_api.py::test_agenda_meeting_api_multiple_desk_datetime_range[Brazil/East-mock_now1] PASSED
tests/test_api.py::test_agenda_meeting_api_multiple_desk_datetime_range[Brazil/East-mock_now2] PASSED
```

Et comme pour la moi la description du bug n'est toujours pas claire; plus haut j'écris : « ça sonne pour moi comme deux problèmes différents; d'une part un horaire trop proche ne devrait pas être réservable; mais d'autre part, si jamais il s'est trouvé réservable, le nombre de réservations ne peut pas dépasser le nombre de places. », j'ai vraiment besoin qu'il y ait d'abord du temps passé à décrire clairement le ou les bugs. (et la description ça peut être un ou deux tests qui jouent une situation spécifique).

> Sur la même idée, si min_datetime = min_datetime.replace(hour=0, minute=0) a lieu dès le début, ça change quoi ?

> Pour le filtre on prend on compte toutes les résa du jour correspondant au minimal_booing_delay et non juste les résa de ce jour à partir de 17:00 (2017-09-08 17:00)

Ça ne m'éclaire pas du tout. Si tu dis qu'on peut passer "2017-09-08 17:00" ou "2017-09-08 22:00" et que peu importe, ça va retourner tous les créneaux du "2017-09-08", alors pourquoi ne peut-on pas aussi lui passer "2017-09-08 00:00" pour les obtenir ?

#14 - 07 septembre 2017 12:49 - Josué Kouka

Ce que j'ai compris du bug:
Prenons un agenda dont le nombre de jour minimale de réservation est de 1, un type de rendez-vous d'une durée de 30 minutes avec une période horaire allant de 10h à 16h tous les vendredi et supposons que nous sommes à la date d'aujourd'hui.
Dans ce paramétrage on aura les créneaux suivants pour demain

2017-09-08
10:00
10:30
11:00
11:30
12:00
..
15:30
16:30

Supposons qu'il est midi et que l'on réserve pour demain à 10h30. Dans le cas ou du mono guichet, après une réservation ce créneau ne devrait plus etre disponnible par conséquent ne devrait plus apparaitre; ce qui n'est pas le cas. Idem pour du miltiguichet, 2 résa sur le meme créneau dans le cas de 2 guichet n'empeche pas que ce créneau soit affiché comme dispinnible.

IMHO, la cause vient du faite que lorsque l'on filtre les évenement réservés de la journée de demain on commence à partir de l'heure à laquelle on est (12h dans notre cas) et non à partir du début du premier créneau de demain (10h).

> J'ai du mal à piger, peut-être parce que la même variable se trouve utilisée avec des sens différents entre deux moments.

Oui, min_datetime est utilisée lors de la génération des créneaux (TimePeriod.get_time_slots) et pour filtrer les réservations (évenements avec résa non annulé).
min_datetime = now() + datetime.timedelta(days=agenda.minimal_booking_delay) => 2017-09-08 12:00
Dans le cas de get_time_slot on s'assure de commencer au où la période horaire commence

```
event_datetime = real_min_datetime.replace(hour=self.start_time.hour,
    minute=self.start_time.minute, second=0, microsecond=0)
```

Ce qui fait que @min_datetime => 2017-09-08 10:00

Ce n'est pas le cas lorsque que l'on filtre les résa et on a min_datetime = 2017-09-08 12:00. Du coup toutes les réservations avant cette date ne sont pas prises en compte.

> Ça ne m'éclaire pas du tout. Si tu dis qu'on peut passer "2017-09-08 17:00" ou "2017-09-08 22:00" et que peu importe, ça va retourner tous les créneaux du "2017-09-08", alors pourquoi ne peut-on pas aussi lui passer "2017-09-08 00:00" pour les obtenir ?

Oui je suis d'arccord, j'aurai du le faire différement que de définir de manière un peu brute que l'on filtrait à partir de 2017-09-08 00:00. L'idéal serait de commencer à l'heure du créneau le plus petit.

#15 - 07 septembre 2017 13:28 - Frédéric Péters

> Ça ne m'éclaire pas du tout. Si tu dis qu'on peut passer "2017-09-08 17:00" ou "2017-09-08 22:00" et que peu importe, ça va retourner tous les créneaux du "2017-09-08", alors pourquoi ne peut-on pas aussi lui passer "2017-09-08 00:00" pour les obtenir ?

> Oui je suis d'arccord, j'aurai du le faire différement que de définir de manière un peu brute que l'on filtrait à partir de 2017-09-08 00:00. L'idéal serait de commencer à l'heure du créneau le plus petit.

Je ne suis pas sûr de comprendre, ce que j'exprimais, dans la continuité d'un message précédent, c'est que les deux lignes que tu ajoutes, si elles n'ont pas d'incidence sur ce que retourne get_time_slots, elles pourraient être en haut de la fonction. (et on s'épargne ainsi des questions de min_datetime qui change en milieu de fonction).

> Prenons un agenda dont le nombre de jour minimale de réservation est de 1, un type de rendez-vous d'une durée de 30 minutes avec une période horaire allant de 10h à 16h tous les vendredi et supposons que nous sommes à la date d'aujourd'hui.

Est-ce possible d'en faire un test, qui couvrirait les neufs cas automatiques.

#16 - 08 septembre 2017 01:22 - Frédéric Péters

```
diff --git a/chrono/api/views.py b/chrono/api/views.py
index 53df293..4956055 100644
--- a/chrono/api/views.py
+++ b/chrono/api/views.py
@@ -38,6 +38,10 @@ from ..agendas.models import (Agenda, Event, Booking, MeetingType,
 def get_open_slots(agenda, meeting_type):
     min_datetime = now() + datetime.timedelta(days=agenda.minimal_booking_delay)
     max_datetime = now() + datetime.timedelta(days=agenda.maximal_booking_delay)
+    min_datetime = min_datetime.replace(hour=0, minute=0, second=0, microsecond=0)
+    max_datetime = max_datetime.replace(hour=0, minute=0, second=0, microsecond=0)
+    max_datetime = max_datetime + datetime.timedelta(days=1)
+
     time_period_filters = {
         'min_datetime': min_datetime,
         'max_datetime': max_datetime,
@@ -50,8 +54,6 @@ def get_open_slots(agenda, meeting_type):
             open_slots_by_desk[time_period.desk_id].addi(slot.start_datetime, slot.end_datetime, slot.desk)

     # set min_datetime to the beginning of the day and max_datetime to the end of the day
-    min_datetime = min_datetime.replace(hour=0, minute=0)
-    max_datetime = min_datetime.replace(hour=23, minute=59)
     for event in agenda.event_set.filter(
             agenda=agenda, start_datetime__gte=min_datetime,
             start_datetime__lte=max_datetime + datetime.timedelta(meeting_type.duration)).select_related(
```

D'une part, éviter ces redéfinitions qui amènent confusion, et donc mettre directement min/max datetime à leurs valeurs correctes, plutôt qu'en change confusément les valeurs en cours de route :

```
+++ b/tests/test_api.py
@@ -780,7 +780,8 @@ def test_agenda_meeting_api_multiple_desk(app, meetings_agenda, user):
         assert queries_count_datetime1 == len(ctx.captured_queries)

-def test_agenda_meeting_api_multiple_desk_datetime_range(app, user):
+def test_agenda_meeting_api_multiple_desk_datetime_range(app, meetings_agenda,
+        mock_now, user):
     app.authorization = ('Basic', ('john.doe', 'password'))
     agenda = Agenda(label='Foo', kind='meetings')
     agenda.minimal_booking_delay = 2
@@ -790,7 +791,9 @@ def test_agenda_meeting_api_multiple_desk_datetime_range(app, user):
     datetime_url = '/api/agenda/meetings/%s/datetimes/' % meeting_type.id
     desk = Desk.objects.create(label='ying', agenda=agenda)
     today = datetime.datetime.today()
-    weekday = today.weekday() + 2
+    weekday = mock_now.weekday() + 2
+    if weekday >= 7:
+        weekday -= 7
     time_period = TimePeriod.objects.create(
         weekday=weekday, start_time=datetime.time(10, 0), end_time=datetime.time(12, 30),
         desk=desk)
@@ -815,6 +818,9 @@ def test_agenda_meeting_api_multiple_desk_datetime_range(app, user):
     resp = app.get(datetime_url)
     assert Booking.objects.count() == 2
     assert len(resp3.json['data']) == len([x for x in resp.json['data'] if not x['disabled']]) + 1
```

```
+
+    # wtf, prétend remplir le dernier fuseau mais après la comparaison sur le
+    # nombre d'événements "disabled" donne le même résultat.
     last_event_data = resp3.json['data'][-1]
     last_event_booking_url = last_event_data['api']['fillslot_url']
     resp = app.post(last_event_booking_url)
@@ -824,3 +830,28 @@ def test_agenda_meeting_api_multiple_desk_datetime_range(app, user):

     resp = app.get(datetime_url)
     assert len(resp3.json['data']) == len([x for x in resp.json['data'] if not x['disabled']]) + 1
+
+
+def test_agenda_meeting_same_day(app, meetings_agenda, mock_now, user):
+    app.authorization = ('Basic', ('john.doe', 'password'))
+    agenda = Agenda(label='Foo', kind='meetings')
+    agenda.minimal_booking_delay = 0
+    agenda.maximal_booking_delay = 15
+    agenda.save()
+    meeting_type = MeetingType.objects.create(agenda=agenda, label='Blah', duration=30)
+    datetime_url = '/api/agenda/meetings/%s/datetimes/' % meeting_type.id
+    desk = Desk.objects.create(label='ying', agenda=agenda)
+    today = datetime.datetime.today()
+    weekday = mock_now.weekday()
+    time_period = TimePeriod.objects.create(
+        weekday=weekday, start_time=datetime.time(11, 0), end_time=datetime.time(12, 30),
+        desk=desk)
+    resp = app.get(datetime_url)
+    event_data = resp.json['data'][0]
+    # check first proposed event is on the same day unless we're past the last
+    # open timeperiod.
+    event_datetime = datetime.datetime.strptime(event_data['datetime'], '%Y-%m-%d %H:%M:%S').timetuple()
+    assert (event_datetime[:3] == mock_now.timetuple()[:3] and
+            event_datetime[3:5] >= mock_now.timetuple()[3:5]) or (
+            event_datetime[:3] > mock_now.timetuple()[:3] and
+            event_datetime[3:5] < mock_now.timetuple()[3:5])

diff --git a/chrono/api/views.py b/chrono/api/views.py
diff --git a/chrono/api/views.py b/chrono/api/views.py
index 53df293..4956055 100644
--- a/chrono/api/views.py
+++ b/chrono/api/views.py
@@ -38,6 +38,10 @@ from ..agendas.models import (Agenda, Event, Booking, MeetingType,
 def get_open_slots(agenda, meeting_type):
     min_datetime = now() + datetime.timedelta(days=agenda.minimal_booking_delay)
     max_datetime = now() + datetime.timedelta(days=agenda.maximal_booking_delay)
+    min_datetime = min_datetime.replace(hour=0, minute=0, second=0, microsecond=0)
+    max_datetime = max_datetime.replace(hour=0, minute=0, second=0, microsecond=0)
+    max_datetime = max_datetime + datetime.timedelta(days=1)
+
     time_period_filters = {
         'min_datetime': min_datetime,
         'max_datetime': max_datetime,
```

Cette partie modifiée, tu t'inquiétais du mock du now mais c'est parce que tu utilisais today.weekday(), qui est en lien avec l'exécution des tests, pas l'heure "réelle".

```
+def test_agenda_meeting_same_day(app, meetings_agenda, mock_now, user):
+    app.authorization = ('Basic', ('john.doe', 'password'))
+    agenda = Agenda(label='Foo', kind='meetings')
+    agenda.minimal_booking_delay = 0
+    agenda.maximal_booking_delay = 15
+    agenda.save()
+    meeting_type = MeetingType.objects.create(agenda=agenda, label='Blah', duration=30)
+    datetime_url = '/api/agenda/meetings/%s/datetimes/' % meeting_type.id
+    desk = Desk.objects.create(label='ying', agenda=agenda)
+    today = datetime.datetime.today()
+    weekday = mock_now.weekday()
+    time_period = TimePeriod.objects.create(
+        weekday=weekday, start_time=datetime.time(11, 0), end_time=datetime.time(12, 30),
+        desk=desk)
+    resp = app.get(datetime_url)
+    event_data = resp.json['data'][0]
+    # check first proposed event is on the same day unless we're past the last
+    # open timeperiod.
+    event_datetime = datetime.datetime.strptime(event_data['datetime'], '%Y-%m-%d %H:%M:%S').timetuple()
```

```
+    assert (event_datetime[:3] == mock_now.timetuple()[:3] and
+            event_datetime[3:5] >= mock_now.timetuple()[3:5]) or (
+            event_datetime[:3] > mock_now.timetuple()[:3] and
+            event_datetime[3:5] < mock_now.timetuple()[3:5])
```

Une des questions posées dans le commentaire c'est assurer ce qui est présenté, un test comme celui-ci fait ça.

#17 - 08 septembre 2017 12:34 - Frédéric Péters

- Fichier 0001-fix-next-day-but-less-than-24h-delta-booking-18495.patch ajouté

Ma compréhension (une réservation prise pour le lendemain et dans moins de 24h n'est pas prise en compte dans le calcul des disponibilités), mon patch, mes tests.

#18 - 08 septembre 2017 17:13 - Thomas Noël

Ca me parait complètement logique. Ack.

#19 - 08 septembre 2017 17:27 - Frédéric Péters

- Statut changé de En cours à Résolu (à déployer)

```
commit b72116012861b9e1febd131f6808b323d6c89001
Author: Frédéric Péters <fpeters@entrouvert.com>
Date:   Wed Sep 6 19:24:15 2017 +0200

    fix next day but less than 24h delta booking (#18495)
```

#20 - 04 décembre 2018 20:08 - Frédéric Péters

- Statut changé de Résolu (à déployer) à Fermé

Fichiers

0001-fix-multiple-desk-booking-fetching-range-18495.patch	3,6 ko 06 septembre 2017	Josué Kouka
0001-fix-multiple-desk-booking-fetching-range-18495.patch	3,6 ko 06 septembre 2017	Josué Kouka
0001-fix-next-day-but-less-than-24h-delta-booking-18495.patch	5,43 ko 08 septembre 2017	Frédéric Péters