AppointmentBooking Backend Dokumentáció

Göncző Kristóf

2021. augusztus 10.

Tartalomjegyzék

1. Bevezetés

A huszonegyedik században, ahol mindent gyorsan és hatékonyan akarunk végezni, akár a bevásárlásról akár a buszjegyünk megvételéről van szó, úgy a fodrászt, a banki ügyintézést, és az orvost is minél hamarabb magunk mögött szeretnénk tudni. Ez a projekt az előbb említett szolgáltatások időpontfoglalásának digitalizálásához és automatizálásához nyújt segítséget.

A cél, hogy minél rugalmasabb szoftvert hozzunk létre, hogy az a legtöbb időpontfoglalást igénylő szervezetnél használható legyen.

2. Hatáskör

A projekt hatásköre csak a szoftver backend részére terjed ki, amit két nagyobb modulra bontunk:

- *Admin*
  Ennek a modulnak a feladata, hogy a rendszer üzemeltetői könnyedén kezeljék a projekt azon részeit amiket a végfelhasználóknak nem szabad elérniük.
- *ClientService*
  Ez a modul tartalmazza az összes olyan részt ami eljut a végfelhasználók felé.

Az Admin modulnak tudnia kell:

- a szolgáltatások és szolgáltatás kategóriák létrehozását,
- az üzlet és/vagy kategóriák nyitvatartásának tárolását,
- bironyos kategóriák foglaltságának részletes lekérdezését[1].

A Client Service modulnak tudnia kell:

- a szolgáltatások és szolgáltatás kategóriák lekérdezését,
- az üzlet és/vagy kategóriák nyitvatartásának lekérdezését,
- a már foglalt/elérhető idősávok lekérdezését,
- az időpontok lefoglalását.

A projekt hatásköre nem terjed ki:

- az adatok megjelenítésére,
- a felhasználók kezelésére és autentikálására,
- az értesítések kiküldésére (SMS/e-mail).

[1] A kategóriák jelenthetnek alkalmazottakat és különálló szolgáltatásokat is.

3. Adatbázis

3.1. Category

A *category* tábla tartalmazza azokat az adatokat amik kellenek a kategória, illetve a szolgáltatás kiválasztásához. Az esetünkben ez csak egy név lesz, viszont természetesen ez a frontend igényeihez szabható, tartalmazhat képet, leírást, egyéb információkat.

A táblában szerepel egy olyan kategória amit nem kell megjeleníteni, ez a *root*. Erre hivatkozva lehet jelölni, hogy főkategóriáról van szó, azaz a felületen először a *root* parentel rendelkező kategóriák jelennek meg.

Név	Típus	Megszorítások	Leírás
id	int	PRIMARY KEY	Azonosító
name	nvarchar		A kategória neve
length	int		A kategória (szolgáltatás) időbeli hossza percben.
parent	int	FOREIGN KEY(category.id)	A kategória apa kategóriája

3.2. Opening_hours

Az *opening_hours* táblában az üzlet (*root*), illetve a kategóriák nyitvatartását adhatjuk meg. Amennyiben egy kategóriának nincs megadva nyitvatartás, a legközelebbi olyan apa kategória nyitvatartását fogja használni aminek van beállítva nyitvatartás.

Név	Típus	Megszorítások	Leírás
category_id	int	PRIMARY KEY, FOREIGN KEY(category.id)	Azonosító
day	enum	PRIMARY KEY	A hét napja amelyre vonatkozik a nyitvatartás
from_time	timestamp		Az óra és perc amitől az adott kategória "nyitva van"
to_time	timestamp		Az óra és perc amitől az adott kategória "zárva van"

3.3. Opening_hours_exception

Az *opening_hours_exception* táblában lehetőség van olyan időtartamok tárolására ahol egy kategória az *opening_hours* tábla szerint nyitva tart, azonban a valóságban nem ez a helyzet. Jó példa erre a munkaszüneti napok.

Név	Típus	Megszorítások	Leírás
id	int	PRIMARY KEY	Azonosító
category_id	int	FOREIGN KEY(category.id)	A kategória azonosítója
from_date	datetime		A dátum és idő amettől érvényes
to_date	datetime		A dátum és idő ameddig érvényes

3.4. Reservation

A *reservation* táblában a foglalásokat tároljuk. Mivel a foglalást is nagyon sok féle képpen lehet csinálni, ezért itt is a legegyszerűbb módszert használom. Alap esetben csak az *id*, a *category*, a *from*, és a *to* mezőkre lenne szükség, illetve a vendég azonosítójára ami egy olyan tábla idegen kulcsa amit a dokumentáció nem tartalmaz, abban a táblában vannak a felhasználók nevei, e-mail címe, stb..

Név	Típus	Megszorítások	Leírás
id	int	PRIMARY KEY	Azonosító
category_id	int	FOREIGN KEY(category.id)	A kategória azonosítója
from_date	datetime		A foglalás időpontjának kezdete
to_date	datetime		A foglalás időpontjának vége
customer_id	nvarchar		A vendég neve

4. Végpontok

4.1. Admin

4.1.1. /category/put

Beszúr vagy frissít egy kategóriát.
HTTP Method: **PUT**
Kéréstörzs mezői:

Név	Típus	Mandatory	Example
id	Integer	0	2
name	String	1	String
length	Integer	0	30
parent	Integer	0	1

Ha az *id* 0, az azonosító automatikusan generálva lesz.
Ha a *parent* 0, a kategória főkategóriaként értelmezendő.
Ha a *parent* nem található, a kategória főkategóriaként értelmezendő.
Válasz kódok:

Kód	Leírás
201	Létrejött az új kategória.
200	Frissült a kategória.
500	Kiszolgáló hiba.

4.1.2. /category/delete

Töröl egy kategóriát.
HTTP Method: **DELETE**
A kérés paraméterei:

Név	Típus	Mandatory	Example
id	Integer	1	1

Válasz kódok:

Kód	Leírás
200	A kategória törlése sikeres vagy nem is létezett.
406	A törledő kategóriának vannak gyerekei.
500	Kiszolgáló hiba.

4.1.3. /openingHours/put

Beszúr vagy frissít egy nyitvatartást.
HTTP Method: **PUT**
Kéréstörzs mezői:

Név	Típus	Mandatory	Example
category	Integer	1	1
day	DayOfWeek	1	MONDAY
from	LocalTime	1	08:00
to	LocalTime	1	16:00

Válasz kódok:

Kód	Leírás
201	Létrejött az új nyitvatartás.
200	Frissült a nyitvatartás.
500	Kiszolgáló hiba.

4.1.4. /openingHours/delete

Töröl egy nyitvatartást.
HTTP Method: **DELETE**
Kérés paraméterei:

Név	Típus	Mandatory	Example
categoryId	Integer	1	1
day	DayOfWeek	1	MONDAY

Válasz kódok:

Kód	Leírás
200	A nyitvatartás törlése sikeres vagy nem is létezett.
500	Kiszolgáló hiba.

4.1.5. /openingHoursException/post

Beszúr egy nyitvatartás kivételt.
HTTP Method: **POST**
Kéréstörzs mezői:

Név	Típus	Mandatory	Example
categoryId	Integer	1	1
from	LocalDateTime	1	1970-01-01T00:00:00.000Z
to	LocalDateTime	1	1970-01-01T00:00:00.000Z

Válasz kódok:

Kód	Leírás
201	A kivétel beszúrása sikeresen megtörtént.
500	Kiszolgáló hiba.

4.1.6. /openingHoursException/delete

Töröl egy nyitvatartás kivételt.
HTTP Method: **DELETE**
Kérés paraméterei:

Név	Típus	Mandatory	Example
id	Integer	1	1

Válasz kódok:

Kód	Leírás
200	A követel törlése sikeres vagy nem is létezett.
500	Kiszolgáló hiba.

4.1.7. /reservation/get

Lekérdezi a lefoglalásokat.
HTTP Method: **GET**
Kérés paraméterei:

Név	Típus	Mandatory	Example
categoryId	Integer	1	1

Ha a *categoryId* üres, akkor az összes foglalást lekérdezi. Abban az esetben ha a kategóriának vannak gyerek kategóriái, azok foglalásait is lekérdezi.
Minta válasz:

```
[
  {
```

```
    "category": 0,
    "customerId": "string",
    "from": "1970-01-01T08:00:00.000Z",
    "id": 0,
    "to": "1970-01-01T08:30:00.000Z"
  }
]
```


4.1.8. /reservation/delete

Töröl egy foglalást.
HTTP Method: **DELETE**
Kérés paraméterei:

Név	Típus	Mandatory	Example
reservationId	Integer	1	1

Válasz kódok:

Kód	Leírás
200	A követel törlése sikeres vagy nem is létezett.
500	Kiszolgáló hiba.

4.2. ClientService

4.2.1. /category/getAll

Lekérdezi az összes kategóriát. Csak akkor ajánlott ha kevés kategória van kevés adattal.
HTTP Method: **GET**
A kérést paraméterek nélkül lehet vérehajtani.
Minta válasz:


```
[
  {
    "id": 0,
    "lenght": 0,
    "name": "string",
    "parent": 0
  }
]
```

Válasz kódok:

Kód	Leírás
200	A válaszüzenet törzse értelmezhető.
500	Kiszolgáló hiba.

4.2.2. /category/getMain

Lekérdezi az összes főkategóriát.
HTTP Method: **GET**
A kérést paraméterek nélkül lehet vérehajtani.
Minta válasz:

```
[
  {
    "id": 0,
    "lenght": 0,
    "name": "string",
    "parent": 0
  }
]
```

Válasz kódok:

Kód	Leírás
200	A válaszüzenet törzse értelmezhető.
500	Kiszolgáló hiba.

4.2.3. /category/getChildren

Lekérdezi egy kategória közvetlen gyerek kategóriáit.
HTTP Method: **GET**
A kérés paraméterei:

Név	Típus	Mandatory	Example
parentId	Integer	1	1

Minta válasz:

```
[
  {
    "id": 0,
    "lenght": 0,
```

```
    "name": "string",
    "parent": 0
  }
]
```

Válasz kódok:

Kód	Leírás
200	A válaszüzenet törzse értelmezhető.
404	Az apa kategória nem található.
500	Kiszolgáló hiba.

4.2.4. /openingHours/get

Lekérdezi a nyitvatartást.

HTTP Method: **GET**

Kérés paraméterei:

Név	Típus	Mandatory	Example
categoryId	Integer	1	1
day	DayOfWeek	0	MONDAY

Ha a *day* üres, akkor megkeresi a nyitvatartást a hét minden napjára.

Ha a kategóriához nem tartozik nyitvatartás, az apa kategória nyitvatartása értelmezendő.

A válaszban egy *Map* érkezik vissza aminek a kulcsai a hét napjai, az értékei a nyitvatartások.

Minta válasz:

```
[
    {
      "MONDAY": {
        "category": 1,
        "day": "MONDAY",
        "from": "08:00",
        "to": "16:00"
      }
    }
]
```

4.2.5. /openingHoursException/get

Lekérdezi a nyitvatartás kivételeket.
HTTP Method: **GET**
Kérés paraméterei:

Név	Típus	Mandatory	Example
categoryId	Integer	1	1

A válaszban egy lista jön, aminek az elemei tartalmazzák a kivételek a nyitó és záró időpontjait.
Minta válasz:

```
[
  {
    "id": 1,
    "categoryId": 1,
    "from": "1970-01-01T00:00:00.000Z",
    "to": "1970-01-08T00:00:00.000Z"
  }
]
```

4.2.6. /reservation/post

Beszúr egy foglalást.
HTTP Method: **POST**
Kéréstörzs mezői:

Név	Típus	Mandatory	Example
category	Integer	1	1
from	LocalDateTime	1	1970-01-01T00:00:00.000Z
to	LocalDateTime	1	1970-01-01T00:00:00.000Z
customerId	String	1	String

Válasz kódok:

Kód	Leírás
201	A kivétel beszúrása sikeresen megtörtént.
500	Kiszolgáló hiba.

5. Tesztelés

A szoftver jelenlegi fázisában bonyolult műveleteket nem hajt végre, így automatikus egység tesztelést (még) nem igényel. Egység tesztelés kézzel lehetséges, hiszen a modulokhoz tartoznak *Swagger UI* felületek a /admin/swagger-ui/index.html#/ /client-service/swagger-ui/index.html#/ és URL-en.

Komponens tesztelést ellenben tudunk automatizálni, ezt SoapUI-ban fogjuk megtenni. A projekt tartalmaz egy *AppointmentBooking-soapui-project.xml* nevű fájlt, ez a SoapUI projekt, amiben a tesztek találhatók.

Mindkét modul minden hívását legalább egyszer tesztelem úgy, hogy a lehető legtágabbabban lefedjem eshetőségeket.

Komponens tesztelést a 6.3. fejezetben komplettebb kategória hierarchiával is elvégezzük.

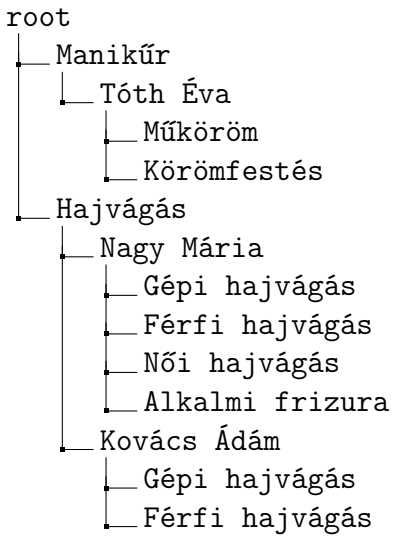
6. Példa a felhasználásra

6.1. Bevezetés

A példa felhasználásban egy fodrászat/szépségszalon időpontfoglalási rendszerének a felépítését fogom bemutatni, 2021 márciusi hónapra konfigurálva.

Fontos tudnivalók:

- Az üzletben 2 fodrász (Nagy Mária, Kovács Ádám) és 1 manikűrös (Tóth Éva) dolgozik.
- Az üzlet hétfőtől péntekig reggel 8 órától délután 6 óráig van nyitva.
- Nagy Mária pénteken nem dolgozik.
- Kovács Ádám hétfőn csak déltől dolgozik.
- A nemzeti ünnepen (2021 március 15. - hétfő) az üzlet nincs nyitva.
- Tóth Éva március 15. és 21. között szabadságon lesz.



1. ábra. Szolgáltatások fagráfban

6.2. Konfiguráció

1. ábra alapján létre kell hoznunk a kategóriákat, ezeket az admin modul */category/put* hívásával tudjuk megtenni. Mivel az üzletnek külön is van nyitvatartása

ezért létre kell hoznunk egy *root* kategóriát, ami a felületen nem lesz megjelenítve, viszont a többi kategóriára egy alapértelmezett nyitvatartásként fog hatni. A hívások paraméterei a következők:

id	name	length	parent
1	root	0	0
2	Manikűr	0	1
3	Tóth Éva	0	2
4	Műköröm	60	3
5	Körömfestés	30	3
6	Hajvágás	0	1
7	Nagy Mária	0	6
8	Gépi hajvágás	20	7
9	Férfi hajvágás	30	7
10	Női hajvágás	45	7
11	Alkalmi frizura	60	7
12	Kovács Ádám	0	6
13	Gépi hajvágás	20	12
14	Férfi hajvágás	30	12

A nyitvatartások konfigurálásához az */openingHours/put* hívásra lesz szükségünk. Először a *root* kategóriát fogjuk konfigurálni (hétfőtől péntekig reggel 8-tól délután 6 óráig), ezt követően pedig az alkalmazottak egyéni munkaóráit. Az utóbbinál csak annyira lesz szükség, hogy Nagy Máriának pénteken egy 0 perces intervallumot adunk meg (0 óra 0 perctől 0 óra 0 percig), ezzel felülírjuk a *root* pénteki nyitvatartását a *Nagy Mária* kategóriára, illetve Kovács Ádámnak hétfőre beállítjuk a 12 órai kezdést, így felülírva a *root* kategória reggel 8 órai kezdését. Az */openingHours/put* metódust a következő paraméterekkel hívjuk:

category	day	from	to
1	MONDAY	08:00	18:00
1	TUESDAY	08:00	18:00
1	WEDNESDAY	08:00	18:00
1	THURSDAY	08:00	18:00
1	FRIDAY	08:00	18:00
7	FRIDAY	00:00	00:00
12	MONDAY	12:00	18:00

Az üzlet kivételes zárvatartását (nemzeti ünnepen) és Tóth Éva szabadságát a

/openingHoursException/post hívással tudjuk konfigurálni. A hívások paraméterei:

categoryId	from	to
1	2021-03-15T00:00:00.000Z	2021-03-16T00:00:00.000Z
3	2021-03-15T00:00:00.000Z	2021-03-22T00:00:00.000Z

Ezek után a hívások után a példában leírt követelményeknek megfeleltünk.

6.3. Tesztelés

A példa tesztelését SoapUI segítségével tudjuk tesztelni. A projekthez tartozik egy *AppointmentBooking-soapui-project.xml* nevű fájl, az a SoapUI projekt, ami tartalmaz egy *Example Values* nevű tesztet. Azt lefuttatva létrehozza a 6.2. fejezetben leírt konfigurációt és leellenőrzi azt, ezzel szimulálva a front-end felületen történő konfigurációt.

6.4. Összefoglalás

A konfiguráció után a vendégeknek a nyitvatartást a *root* kategória nyitvatartáson keresztül kell kommunikálni (kivonva belőle a *root*hoz tartozó kivételeket), az alkalmazottak pedig a 3 (Tóth Éva), 7 (Nagy Mária) és 12 (Kovács Ádám) id-vel ellátott kategóriákat figyelik, az azokhoz tartozó foglalásokat és nyitvatartást.

7. Összefoglalás

Ezzel a két modullal egy egyszerűbb időpontfoglalási rendszer konfigurációja elvégezhető, viszont az tisztán látszik, hogy ez egy prototípus. A különböző kategóriákról általában több információra van szüksége a vendégnek (például: részletes leírás, ár), illetve érdekes kérdés az is, hogy mi történik az adatbázis inserten kívül amikor egy foglalás megtörténik. Küldön visszaigazoló e-mailt? Esetleg mobil értesítést? A projekt hatásköre (egyelőre) ezekre a kérdésekre nem ad. A cél egy minél általánosabb rendszer fejlesztése volt amit elvégeztünk.