# Baza podataka

## UserType

|  |  |  |
| --- | --- | --- |
| Atribut/Stupac | Tip | Opis |
| Id | Integer | Identifikator tipa korisnika |
| Name | String | Naziv tipa korisnika, jedinstven u aplikaciji (npr. Admin, Waiter) |

## UserData

|  |  |  |
| --- | --- | --- |
| Atribut/Stupac | Tip | Opis |
| Id | Integer | Identifikator korisnika |
| UserName | String | Korisničko ime, jedinstveno u aplikaciji |
| Password | String | Lozinka |
| FullName | String | Puno ime korisnika |
| UserTypeId | Integer | Identifikator tipa korisnika kojem korisnik pripada (strani ključ) |

## Category

|  |  |  |
| --- | --- | --- |
| Atribut/Stupac | Tip | Opis |
| Id | Integer | Identifikator kategorije |
| Name | String | Naziv kategorije, jedinstven u aplikaciji |

## Product

|  |  |  |
| --- | --- | --- |
| Atribut/Stupac | Tip | Opis |
| Id | Integer | Identifikator proizvoda |
| Name | String | Naziv proizvoda, jedinstven u sustavu |
| AvailableQuantity | Integer | Raspoloživa količina |
| Price | Float | Cijena proizvoda |
| CategoryId | Integer | Identifikator kategorije kojoj proizvod pripada (strani ključ) |

## Bill

|  |  |  |
| --- | --- | --- |
| Atribut/Stupac | Tip | Opis |
| Id | Integer | Identifikator računa |
| Number | String | Naziv računa u formatu yyyy-MM-dd-xxxx, gdje je xxxx redni broj računa u tom danu, jedinstven u sustavu |
| TotalPrice | Float | Ukupan iznos računa, može se izračunati iz tablice BillContent, ali zgodno ga je imati i ovdje zbog preglednosti |

## BillContent

|  |  |  |
| --- | --- | --- |
| Atribut/Stupac | Tip | Opis |
| Id | Integer | Identifikator retka |
| ProductPrice | Float | Cijena proizvoda – iako imamo BillId, navodimo je ovdje jer se cijena može mijenjati kroz vrijeme pa ćemo u danom retku imati cijenu koja je tada bila aktualna |
| ProductQuantity | Integer | Količina proizvoda |
| ProductId | Integer | Identifikator proizvoda (strani ključ) |
| BillId | Integer | Identifikator računa (strani ključ) |

## BillStatus

|  |  |  |
| --- | --- | --- |
| Atribut/Stupac | Tip | Opis |
| Id | Integer | Identifikator statusa |
| Name | String | Naziv statusa |
| StatusTimestamp | DateTime | Vrijeme statusa |
| BillId | Integer | Identifikator računa (strani ključ) |

## FileData

|  |  |  |
| --- | --- | --- |
| Atribut/Stupac | Tip | Opis |
| Id | Integer | Identifikator datoteke |
| Name | String | Naziv datoteke, jedinstven u sustavu |
| FileTimestamp | DateTime | Vrijeme kreiranja datoteke |
| ContentType | String | Format datoteke |
| Data | String | Sadržaj datoteke |

## BillReport

|  |  |  |
| --- | --- | --- |
| Atribut/Stupac | Tip | Opis |
| Id | Integer | Identifikator izvještaja |
| Year | Integer | Godina izvještaja |
| Month | Integer | Mjesec izvještaja |
| Day | Integer | Dan izvještaja |
| FileDataId | Integer | Identifikator datoteke u kojoj se nalazi tekst izvještaja (strani ključ) |

## SupplyReport

|  |  |  |
| --- | --- | --- |
| Atribut/Stupac | Tip | Opis |
| Id | Integer | Identifikator izvještaja |
| Year | Integer | Godina izvještaja |
| Month | Integer | Mjesec izvještaja |
| Day | Integer | Dan izvještaja |
| FileDataId | Integer | Identifikator datoteke u kojoj se nalazi tekst izvještaja (strani ključ) |

# Preslikavanje tablica u klase (entitete)

Ako nije drugačije navedeno, sve tablice iz baze podataka preslikavaju se u klase točno onako kako su definirane i u bazi.

Entitet UserData umjesto atributa userTypeId ima atribut koji predstavlja instancu klase UserType jer preko toga možemo znati je li korisnik administrator, konobar ili nešto treće. Bez toga bismo imali samo identifikator tipa korisnika, što nam ne bi ništa govorilo o tome. To znači da taj entitet u C#-u izgleda npr. ovako:

class UserData {

public int id;

public string userName;

public string password;

public string fullName;

public UserType userType;

};

Dodatno, radi preglednosti i jednostavnosti rada s podatcima, neke je stvari zgodno grupirati u cjelinu kako bismo minimalizirali broj upita prema sustavu, npr. možemo imati klasu MainData koja se sastoji od sljedećih elemenata:

* List<Category> categories;
* List<Product> products;

Na taj način kad dođemo na naslovnu stranicu možemo uputiti 1 request kojim ćemo pokupiti kategorije i proizvode u komadu, tj. ne moramo imati 1 request za kategorije i 1 za proizvode.

# Specifikacija REST servisa

Servis će se nalaziti na nekom serveru u obliku neke web aplikacije. U sljedećim potpoglavljima navodimo i detaljno opisujemo metode koje će biti implementirane u sklopu servisa, a koristit će ih svaka frontend aplikacija po potrebi. Metode navodimo u obliku „TIP\_METODE /<IME\_METODE>“, što znači da je endpoint (URL) metode dobiven konkatenacijom lokacije servera, imena projekta i imena metode.

## Login

### POST /login

* Opis: prijava korisnika
* Parametri i tip: objekt klase userData, application/json
* Povratni entitet i tip: objekt klase UserData (preko toga ćemo dobiti pune podatke o korisniku – njegovo ime, je li administrator ili ne i sl.), application/json
* Status odgovora (ovo je standarni scenarij pa ćemo ga ubuduće samo kratko navesti):
  + 200 (OK) – ako je sve bilo OK
  + 500 (Internal Server Error) – ako je došlo do greške na strani servisa
* Primjer poziva: curl -X POST localhost:8080/backend-app/api/login -H „Content-Type: application/json“ -d „<JSON\_REPREZENTACIJA\_KLASE\_USERDATA>“

## MainData

### GET /main-data

* Opis: vraća sve kategorije i dostupne proizvode
* Parametri i tip: nema
* Povratni entitet i tip: objekt klase MainData, application/json
* Status odgovora: standardno
* Primjer poziva: curl -X GET localhost:8080/backend-app/api/main-data

## Bill

### POST /bills

* Opis: zahtjev za kreiranjem računa
* Parametri i tip: objekt klase Bill, application/json
* Povratni entitet i tip: nema
* Status odgovora: standardno
* Primjer poziva: curl -X POST localhost:8080/backend-app/api/bills -H „Content-Type: application/json“ -d „<JSON\_REPREZENTACIJA\_KLASE\_BILL>“

Klasa Bill izgledat će ovako:

class Bill {

public int id;

public string number;

public string totalPrice;

public List<BillContent> contents;

public List<BillStatus> statuses;

};

Drugim riječima, sadržaj računa bit će naveden u listi objekata klase BillContent, a povijest statusa u listi objekata klase BillStatus – prije kreiranja nije moguće vezati preko stranih ključeva jer entitet Bill nema ID dok se ne spremi u bazu. Klasa BillContent bit će ista kao i odgovarajuća tablica u bazi.

### PUT /bills

* Opis: ažuriranje računa – dodavanje/brisanje sadržaja, izmjena statusa
* Parametri i tip: objekt klase Bill, application/json
* Povratni entitet i tip: nema
* Status odgovora: standardno
* Primjer poziva: curl -X PUT localhost:8080/backend-app/api/bills -H „Content-Type: application/json“ -d „<JSON\_REPREZENTACIJA\_KLASE\_BILL>“

### DELETE /bills/{id}

* Opis: brisanje računa prema ID-u
* Parametri i tip: ID računa, dio patha
* Povratni entitet i tip: nema
* Status odgovora: standardno
* Primjer poziva: curl -X DELETE localhost:8080/backend-app/api/bills/1

### GET /bills

* Opis: dohvat svih računa
* Parametri i tip: nema
* Povratni entitet i tip: lista objekata klase Bill, application/json
* Status odgovora: standardno
* Primjer poziva: curl -X GET localhost:8080/backend-app/api/bills

## Category

### POST /categories

* Opis: kreiranje nove kategorije
* Parametri i tip: objekt klase Category
* Povratni entitet i tip: nema
* Status odgovora: standardno
* Primjer poziva: curl -X POST localhost:8080/backend-app/api/categories -H „Content-Type: application/json“ -d „<JSON\_REPREZENTACIJA\_KLASE\_CATEGORY>“

### PUT /categories

* Opis: ažuiranje kategorije – mijenjanje imena, premještanje proizvoda u drugu kategoriju, dodavanje proizvoda iz druge kategorije
* Parametri i tip: objekt klase Category, application/json
* Povratni entitet i tip: nema
* Status odgovora: standardno
* Primjer poziva: curl -X POST localhost:8080/backend-app/api/categories -H „Content-Type: application/json“ -d „<JSON\_REPREZENTACIJA\_KLASE\_CATEGORY>“

### DELETE /categories/{id}

* Opis: brisanje kategorije prema ID-u
* Parametri i tip: ID kategorije, dio patha
* Povratni entitet i tip: nema
* Status odgovora: standardno
* Primjer poziva: curl -X DELETE localhost:8080/backend-app/api/categories/1

## Product

### POST /products

* Opis: kreiranje novog proizvoda
* Parametri i tip: objekt klase Product, application/json
* Povratni entitet i tip: nema
* Status odgovora: standardno
* Primjer poziva: curl -X POST localhost:8080/backend-app/api/products -H „Content-Type: application/json“ -d „<JSON\_REPREZENTACIJA\_KLASE\_PRODUCT>“

### PUT /products

* Opis: ažuriranje proizvoda – izmjena cijene, količine, kategorije (ime se ne mijenja jer će to zapravo biti novi proizvod)
* Parametri i tip: objekt klase Product, application/json
* Povratni entitet i tip: nema
* Status odgovora: standardno
* Primjer poziva: curl -X PUT localhost:8080/backend-app/api/products -H „Content-Type: application/json“ -d „<JSON\_REPREZENTACIJA\_KLASE\_PRODUCT>“

### DELETE /product/{id}

* Opis: brisanje proizvoda prema ID-u
* Parametri i tip: ID proizvoda, dio patha
* Povratni entitet i tip: nema
* Status odgovora: standardno
* Primjer poziva: curl -X DELETE localhost:8080/backend-app/api/products/1

## BillReport

### GET /bill-reports

* Opis: dohvat svih izvještaja za račune
* Parametri i tip: nema
* Povratni entitet i tip: lista objekata klase BillReport, application/json
* Status odgovora: standardno
* Primjer poziva: curl -X GET localhost:8080/backend-app/api/bill-reports

## SupplyReport

### GET /supply-reports

* Opis: dohvat svih izvještaja za nabavu
* Parametri i tip: nema
* Povratni entitet i tip: lista objekata klase SupplyReport, application/json
* Status odgovora: standardno
* Primjer poziva: curl -X GET localhost:8080/backend-app/api/supply-reports

## FileData

### GET /file-data/{id}

* Opis: dohvat datoteke po ID-u
* Parametri i tip: ID datoteke, dio patha
* Povratni entitet i tip: objekt klase FileData, application/json
* Status odgovora: standardno
* Primjer poziva: curl -X GET localhost:8080/backend-app/api/file-data/1