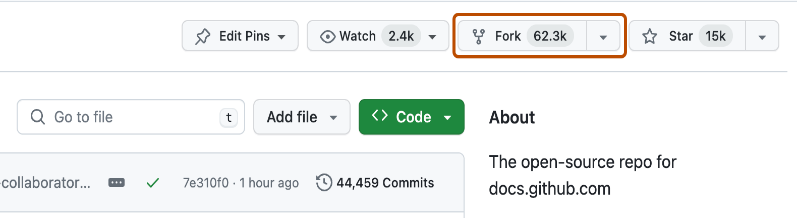
Candy Shop

Project 3

**მოთხოვნა:**

git**-**ის გამოყენებით ატვირთლი [პროექტის შაბლონს](https://github.com/sabazed/MZIURI-Project-2) გაუკეთეთ fork, რომელშიც შემდგომში იმუშავებთ.

Fork ინსტრუქცია - დააჭირეთ მონიშნულ ღილაკს და შემდეგ გვერძე დააწექით Create Fork  
  
  
**აღწერა:**

პროექტის ფარგლებში თქვენ უნდა გააკეთოთ პატარა საიტი, სადაც შეძლებენ მომხმარებლები პროდუქტების ნახვას და შეძენას, მფლობელი კი ახალი პროდუქტების დამატებას

1. **შექმენით მონაცემთა ბაზა:**

**სახელი:** CandyShop  
**ცხრილი**:

* product:  
  ველები:
  + prod\_id (int) PK
  + prod\_name (varchar(50)) UNIQUE
  + prod\_price(float)
  + prod\_amount(int)

1. **შექმენით კლასები:**

* **Product** :

product ბაზის ცხრილის შესაბამისი @Entity კლასი

* **GetProductsResponse**:
  + String[] productNames
* **StorageConfig**
  + Product[] products
  + String password
* **GetProductInfoRequest**:
  + String name
* **GetProductInfoResponse**:
  + String name
  + Float price
  + Integer amount
* **PurchaseRequest**:
  + String name
  + Integer amount
* **PurchaseResponse**:
  + String name
  + Integer remainingAmount
* **AddProductRequest**:
  + String password
  + String name
  + Integer amount
* **AddProductResponse**:
  + String name
  + Integer remainingAmount

1. **შექმენით სერვისები:**

* **StorageReader (Singleton):**

გამოიყენეთ პროექტში მოცემული storage.json ფაილის წასაკითხად, წაკითხული პროდუქტი კი დაამატეთ თქვენს ბაზაში. წასაკითხად გამოიყენეთ Jackson

* **DatabaseManager (Singleton):**

გამოიყენეთ თქვენს ბაზასთან სამუშაოდ. მხოლოდ ამ კლასს უნდა ჰქონდეს EntityManager-ის გამოყენების უფლება (და არა პირდაპირ სერვლეტებს); სამუშაოდ გამოიყენეთ Criteria API

1. **storage.json**:

ეს არის ფაილი, რომელიც მდებარეობს src/main/resources ფოლდერში, სადაც მოცემულია ყველა ის პროდუქტი, რაც გაიყიდება მაღაზიაში. ასევე აქვე იქნება მოცემული სპეციალური პაროლი, რომელიც საჭიროა მაღაზიაში პროდუქტების დასამატებლად. ეს ფაილი წაიკითხეთ როგორც StorageConfig კლასის ობიექტი.

1. **შექმენით ორი servlet**:

* **StoreServlet** (/store):  
  მეთოდები:
  + [**GET**]  
    **აღწერა:** პროდუქტების სიის ნახვა.  
    **პარამეტრები:** არაფერი   
    **აბრუნებს:** GetProductsResponse ობიექტს (response body-ში)
* **ProductServlet** (/store/product):  
  მეთოდები:
  + [**GET**]

**აღწერა:** პროდუქტის შესახებ ინფორმაციის ნახვა.

**პარამეტრები:** GetProductInfoRequest ობიექტი (request body-ში)  
**აბრუნებს:** GetProductInfoResponse ობიექტს (response body-ში), იმ შემთხვევაში თუ მოცემული სახელის პროდუქტი მოიძებნა, წინააღმდეგ შემთხვევაში კი response-ს გაუწერეთ სტატუს კოდი *405*

* + [**POST**]

**აღწერა:** გამოიყენება პროდუქტის შესაძენად.

**პარამეტრები:** PurchaseRequest ობიექტი (request body-ში)  
**აბრუნებს:** PurchaseResponse ობიექტს (response body-ში), იმ შემთხვევაში თუ გვაქვს პროდუქტის ხელმისაწვდომი რაოდენობა და პროდუქტი მოიძებნა, წინააღმდეგ შემთხვევაში არ შეიძინოთ არცერთი პროდუქტი და response-ს გაუწერეთ სტატუს კოდი *405*

* + [**PUT**]

**აღწერა:** პროდუქტის რაოდენობის განახლება. მხოლოდ იმ შემთხვევაშია შესაძლებელი, თუ request-ის პარამეტრად იქნება სპეციალური პაროლი ჩაწერილი

**პარამეტრები:** AddProductRequest ობიექტი (request body-ში)  
**აბრუნებს:** AddProductResponse ობიექტს (response body-ში), იმ შემთხვევაში თუ მოთხოვნაში გადაცემული პაროლი შეესაბამება storage.json-ში მოცემულ პაროლს, წინააღმდეგ შემთხვევაში არ დაამატოთ პროდუქტი და response-ს გაუწერეთ სტატუს კოდი *403.* თუ პროდუქტი ვერ მოიძებნა, გაუწერეთ კოდი 405.

1. **შექმენით ორი გვერდი:**

* Index.html:

ეს იქნება ძირითადი მომხმარებლების გვერდი, სადაც ისინი შესვლისთანავე ნახავენ იმ პროდუქტების ჩამონათვალს, რომელიც მაღაზიას აქვს (გამოიყენეთ **StoreServlet**). ყველა ამ პროდუქტს უნდა ახლდეს ღილაკი „Purchase“, რომელზე დაჭერის შედეგადაც მომხმარებელის ეკრანზე უნდა გამოვიდეს პატარა ფანჯარა, სადაც ნაჩვენები იქნება პროდუქტის შესახებ ინფორმაცია: სახელი, ფასი, ხალმისაწვდომი რაოდეობა. ამ ინფორმაციის ქვემოთ კი უნდა იყოს ველი, სადაც მომხმარებელი შეიყვანს სასურველ რაოდენობას, რამდენის შეძენაც სურს, და ღილაკი „Buy Now“, რომელიც გზავნის POST request-ს **ProductServlet**-თან. მიღებული repsonse თუ წარმატებულია (სტატუსი 200), მაგ შემთხვევაშ დახურავს ფანჯარას, ხოლო თუ მოთხოვნა წარუმატებელი იყო (სტატუსი არ არის 200), მაგ შემთხვევაში აჩვენებს alert()-ის მეშვეობით ნოტიფიკაციას მესიჯით: „Your purchase has failed!“ და აღარ ხურავს პროდუქტის ფანჯარას.

* storage.html:  
  ეს იქნება გვერდი, საიდანაც შესაძლებელი იქნება პროდუქტების დამატება. გვერდზე იქნება მხოლოდ 3 input ველი და 1 ღილაკი წარწერით “Add Products”. ამ ველებს თან ახლავთ თავიანთი label (ან ნებისმიერი სხვა) ტექსტური ელემენტები:
  + Product Name
  + Amount
  + Password

ველების შეყავნის და ღილაკზე დაჭერის შემდგომ, **ProductServlet**-თან იგზავნება PUT request და პასუხის სტატუსის კოდი გამოდის ეკრანზე alert()-ის მეშვეობით

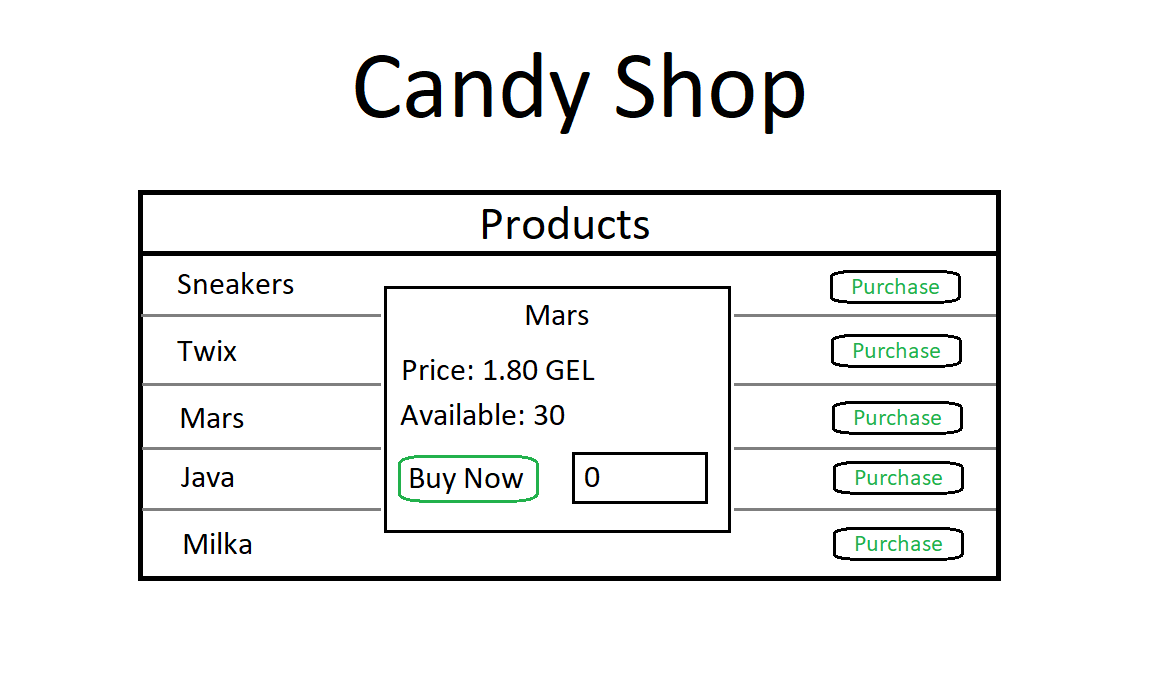
**ინსტრუქცია:**

გადმოწერილი პროექტის გაშვების შემდგომ, შეგიძლიათ შეხვიდეთ <http://localhost:8989/candy-shop/index.html> და <http://localhost:8989/candy-shop/storage.html> რათა იხილოთ თქვენი საიტი. ხოლო საიტის კოდის შესაცვლელად შეგიძლიათ შეცვალოთ src/main/webapp/index.html და storage.html ფაილი და დაამატოთ თქვენი სასურველი ნებისმიერი ფაილი/რესურსი ამავე webapp ფოლდერში

**ვებსაიტის მაგალითი (დამაკმაყოფილებელი):**

**Index.html:**

****

**`**

**storage.html:**

****