Szeretnék készíteni egy webalkalamzást Java nyelven Spring Boot-tal készülne a backend az adatokat mariaDB-ben tartanám nyilván és Angularban a frontend. Terveim szerint minden docker-ben futna. Az alkalmazás egy készlet nyilvántartó és számlázó alkalmazás lenne, esetleg egy webshophoz is tudna adatot szolgáltatni, de a webshop funkció fejlesztése nem része a mostani feladatnak. A készleten lévő termékeket két fő csoportra sorolnám, bár később lehet. hogy több csoport is lesz. 1-es csoport árú jellegű termék, ennek a fő jellemzője, hogy készletet kezelünk vele. Tehát értelmezhető a bevételezés és az eladás. Bevételezésen nő a raktáron lévő termékek száma, eladáskor pedig csökken. Negatív készlet nem engedélyezett. 2-es csoport a szolgáltatások. Ebben az esetben nem kezelünk készletet. Ilyen szolgáltatás lehet pl.: munkadíj. Munkadíjat nem vételezünk be, viszont számlázásnál eladhatjuk. Mindezekből következik, hogy kell törzskarbantartás menü, bevételezés menü és eladás menü. Már most gondolkodok lekérdezés menüben is, de ez jelenleg még nem konkretizálódott bennem. Az eladás végén valamilyen nyomtatvány keletkezik pdf formátumban. Az alábbi eladás típusokat különböztetünk meg.

* Készpénzes számla készül,
* Bankkártyás számla készül,
* Átutalásos számla készül
* Készpénzes pénztárbizonylat készül (nyugta)
* Bankkártyás pénztárbizonylat készül (nyugta)

A termékekkel kapcsolatban az alábbi adatokat kezelném:

* Megnevezés
* Gyártói cikkszám (egyedi)
* Gyártó
* EAN kód (egyedi)
* Mennyiség egység
* Készleten lévő mennyiség (szolgáltatás esetében nem kezelendő)
* Nettó beszerzési ár
* Bruttó beszerzési ár(Bármelyik árat adom meg az Áfa értékét figyelembe véve a másikat kiszámolja
* Nettó eladási ár
* Bruttó eladási ár (Bármelyik árat adom meg az Áfa értékét figyelembe véve a másikat kiszámolja)
* Áfa kulcs (értékei lehetnek 0%,5%,27%, stb.)
* Termék leírás
* Termék kép
* Garancia időtartama (sajnos előfordulhat, hogy két különböző időpontban bevételezett termének eltérő a garancia időpontja)
* Termék kategória (Alaplap, Egér, stb.)

A partnerekkel kapcsolatban az alábbi adatokat kezelném

* Név
* Irányítószám
* Település
* Közterület jellege
* Házszám
* Épület
* Lépcsőház
* Emelet
* Ajtó
* Helyrajzi szám
* Magánszemély/Nem magánszemély (Adószámmal rendelkező)
* Adószám
* E-mail
* Telefonszám
* Preferált fizetési mód (Készpénz/Bankkártya, Átutalás, Előre utalás, Utánvétel, stb.)
* Eltárolnám azt is, hogy mikor rögzítettem a táblába
* Egyenleg (Itt jó volna tudnom, hogy mennyit költött és mennyit fizetett ki eddig)
* alapértelmezett fizetési határidő

Saját adatok, hasonlóan a partnerhez, de ezek az adatok kibővülnek továbbiakkal:

* Székhely címe
* Aktuális telephelyek címe (Automatikusan a Székhely, de módosítható legyen)
* Cégjegyzék száma

A Számla esetében az alábbi adatokat kezelném, illetve nagyon fontos, hogyha egy számla kiállításra került, akkor a későbbiekben a kiállításkori állapotot kel tudnom produkálni akkor is, ha közben a termék megváltozott, vagy az Áfa kódja változott a terméknek, megváltozott a vevő címe stb.

* Eladó (Saját magam, azaz bővített partner)
* Vevő (Partner)
* Számla kelte
* Teljesítés kelte
* Fizetési határidő
* Számla tételek-> Termékek
* A számla tételei tetszőlegesen bővíthetőek kellenek, hogy legyenek, azaz bármennyi termék hozzá adható legyen
* A számla tételeinél fontos, hogy hány darabot adok el
* Mennyi az egységára a tételeknek, Termék törzsből átvett eladási ár, viszont ezen lehessen módosítani, ha szükséges
* Kedvezmény értéke (%-ban megadva, alapértelmezés szerint 0, de lehessen változtatni rajta, ilyenkor az eladási árat számolja újra)
* Számla nettó összesen
* Számla bruttó összesen
* Áfánkénti bontásban a számla Áfa tartalma
* Számla típusa (Számla/eladási jegy/Garanciajegy)
* Fizetési mód (Készpénz, bankkártya, átutalás, stb.)

A termékeket vonalkód olvasóval EAN kód alapján szeretném hozzáadni a számla tételeihez, de legyen lehetőség gyártói cikkszámra, vagy megnevezésre is kereseni/szűrni és úgy kiválasztani a tételt. Ha esetleg tudom fejből a gyártói cikkszámot, akkor azt beírva is ki tudjam választani a termékeket.

Arra gondoltam, hogy minden táblára legyen igaz, hogyha már volt felhasználva az adott rekord valamilyen célra, akkor ne lehessen véglegesen törölni, csak egy státusz legyen beállítva, hogy törölt és ilyenkor a keresésnél ne is látszódjon, de a törzs karbantartásnál viszont legyen látható és akár vissza is tudjam vonni a törölt státuszt.

A számlázásnál a partnert ki tudjam keresni adószám alapján, egyedi azonosító alapján, vagy név töredék alapján is.

Egyenlőre ezek jutottak eszembe, de kíváncsi vagyok, hogy mivel egészítenéd ki, gondolkodjunk együtt, esetleg tegyél fel kérdéseket, hogy van-e valamire még szükségem, amire hirtelen nem gondoltam.

Elsősorban mindent angolul szeretnék a forrásban és a táblákban elnevezni, viszont a frontenden, vagy a swaggerben magyarul jelenítenék meg minden információt a felhasználó számára.  
Kérlek javasolj egy nevet is az applikációnak angolul, ami utal a funkciójára.

Ha minden összeállt, akkor az entitásokat össze is kellene egyúttal állítani Java class formájában. Természetesen JPA-t használnék.

Javasolt végpontok:

|  |  |  |
| --- | --- | --- |
| **Módszer** | **URL** | **Leírás** |
| GET | /api/products | Aktív termékek listázása |
| GET | /api/products/{id} | Adott termék lekérdezése |
| POST | /api/products | Új termék rögzítése |
| PUT | /api/products/{id} | Termék módosítása |
| DELETE | /api/products/{id} | Termék "törlése" (státusz átállítás) |
| GET | /api/partners | Partnerek listázása |
| GET | /api/partners/{id} | Partner adatainak lekérdezése |
| POST | /api/partners | Új partner rögzítése |
| PUT | /api/partners/{id} | Partner adatainak módosítása |
| GET | /api/partners/search | Keresés: adószám, név töredék, stb. |
| GET | /api/invoices | Számlák listázása |
| GET | /api/invoices/{id} | Számla lekérdezése |
| POST | /api/invoices | Számla kiállítása |
| GET | /api/invoices/{id}/pdf | PDF letöltése |
| POST | /api/stock/in | Bevételezés |
| POST | /api/stock/out | Eladás |
| GET | /api/stock/history | Készletmozgások lekérdezése |
| GET | /api/company | Cégadatok lekérdezése |
| PUT | /api/company | Cégadatok módosítása |
| GET | /api/manufacturers | Gyártók listázása |
| POST | /api/manufacturers | Új gyártó rögzítése |
| GET | /api/zip-codes | Irányítószám lista |
| GET | /api/products/{id}/barcodes | Termék vonalkódjainak lekérdezése |
| POST | /api/products/{id}/barcodes | Vonalkód hozzáadása termékhez |
| GET | /api/products/{id}/serial-numbers | Termék gyári számainak listája |
| POST | /api/products/{id}/serial-numbers | Új gyári szám(ok) hozzáadása |
| POST | /api/barcodes/validate | Vonalkód validálása |
| POST | /api/barcodes/generate | Vonalkód generálása |
| POST | /api/stock-in | Bevételezés létrehozása |
| POST | /api/stock-in/{id}/serial-numbers | Gyári számok rögzítése bevételezéshez |
| POST | /api/invoices/{id}/cancel | Számla sztornózása (gyáriszám + készlet visszaáll) |

// Teljes kódállomány az InvotraxApp entitásaihoz, DTO-ihoz és kontrollereihez

// A fájl több részletben fog felépülni az átláthatóság érdekében

// 1. ENTITÁSOK

// Manufacturer

@AllArgsConstructor

@NoArgsConstructor

@Getter

@Setter

@Entity

@Table(name = "manufacturer")

public class Manufacturer {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "code", unique = true, nullable = false)

private String code;

@Column(name = "name", nullable = false)

private String name;

@Column(name = "website")

private String website;

}

// ZipCode

@AllArgsConstructor

@NoArgsConstructor

@Getter

@Setter

@Entity

@Table(name = "zip\_code")

public class ZipCode {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "zip\_code", nullable = false)

private String zipCode;

@Column(name = "city", nullable = false)

private String city;

}

// SerialNumber

@AllArgsConstructor

@NoArgsConstructor

@Getter

@Setter

@Entity

@Table(name = "serial\_number")

public class SerialNumber {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "serial", nullable = false)

private String serial;

@Column(name = "used", nullable = false)

private boolean used = false;

@ManyToOne

@JoinColumn(name = "product\_id")

private Product product;

@ManyToOne

@JoinColumn(name = "inventory\_item\_id")

private InventoryItem inventoryItem;

}

// Barcode

@AllArgsConstructor

@NoArgsConstructor

@Getter

@Setter

@Entity

@Table(name = "barcode")

public class Barcode {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "code", unique = true, nullable = false)

private String code;

@ManyToOne

@JoinColumn(name = "product\_id")

private Product product;

}

// Inventory

@AllArgsConstructor

@NoArgsConstructor

@Getter

@Setter

@Entity

@Table(name = "inventory")

public class Inventory {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToOne

@JoinColumn(name = "supplier\_id")

private Partner supplier;

@Column(name = "received\_at")

private LocalDateTime receivedAt;

@OneToMany(mappedBy = "inventory", cascade = CascadeType.ALL)

private List<InventoryItem> items;

}

// InventoryItem

@AllArgsConstructor

@NoArgsConstructor

@Getter

@Setter

@Entity

@Table(name = "inventory\_item")

public class InventoryItem {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToOne

@JoinColumn(name = "inventory\_id")

private Inventory inventory;

@ManyToOne

@JoinColumn(name = "product\_id")

private Product product;

@Column(name = "quantity")

private int quantity;

@Column(name = "net\_price")

private BigDecimal netPrice;

@Column(name = "gross\_price")

private BigDecimal grossPrice;

@Column(name = "warranty\_period\_months")

private Integer warrantyPeriodMonths;

}

// Invoice

@AllArgsConstructor

@NoArgsConstructor

@Getter

@Setter

@Entity

@Table(name = "invoice")

public class Invoice {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToOne

@JoinColumn(name = "seller\_id")

private Partner seller;

@ManyToOne

@JoinColumn(name = "buyer\_id")

private Partner buyer;

@Column(name = "issued\_at")

private LocalDate issuedAt;

@Column(name = "fulfillment\_at")

private LocalDate fulfillmentAt;

@Column(name = "due\_date")

private LocalDate dueDate;

@Column(name = "invoice\_type")

private String invoiceType;

@Column(name = "payment\_method")

private String paymentMethod;

@Column(name = "net\_total")

private BigDecimal netTotal;

@Column(name = "gross\_total")

private BigDecimal grossTotal;

@OneToMany(mappedBy = "invoice", cascade = CascadeType.ALL)

private List<InvoiceItem> items;

}

// InvoiceItem

@AllArgsConstructor

@NoArgsConstructor

@Getter

@Setter

@Entity

@Table(name = "invoice\_item")

public class InvoiceItem {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToOne

@JoinColumn(name = "invoice\_id")

private Invoice invoice;

@ManyToOne

@JoinColumn(name = "product\_id")

private Product product;

@Column(name = "quantity")

private int quantity;

@Column(name = "unit")

private String unit;

@Column(name = "unit\_price")

private BigDecimal unitPrice;

@Column(name = "discount\_percent")

private BigDecimal discountPercent;

@Column(name = "net\_amount")

private BigDecimal netAmount;

@Column(name = "gross\_amount")

private BigDecimal grossAmount;

}

// Product

package com.invotraxapp.model;

import jakarta.persistence.\*;

import lombok.\*;

import java.math.BigDecimal;

import java.util.List;

@Entity

@Table(name = "product")

@Data

@NoArgsConstructor

@AllArgsConstructor

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String sku;

private String description;

@ManyToOne

@JoinColumn(name = "category\_id")

private ProductCategory category;

@ManyToOne

@JoinColumn(name = "manufacturer\_id")

private Manufacturer manufacturer;

@Column(name = "net\_price")

private BigDecimal netPrice;

@Column(name = "gross\_price")

private BigDecimal grossPrice;

@Column(name = "warranty\_period\_months")

private Integer warrantyPeriodMonths;

@Column(name = "serial\_number\_required")

private boolean serialNumberRequired;

@OneToMany(mappedBy = "product", cascade = CascadeType.ALL, orphanRemoval = true)

private List<Barcode> barcodes;

@OneToMany(mappedBy = "product", cascade = CascadeType.ALL, orphanRemoval = true)

private List<SerialNumber> serialNumbers;

}

DTO-k

// DTO-k minden entitáshoz

// DTO-k minden entitáshoz és Controller + Repository osztályok

package com.invotraxapp.dto;

import lombok.\*;

import java.math.BigDecimal;

import java.time.LocalDate;

import java.time.LocalDateTime;

import java.util.List;

@Data

@AllArgsConstructor

@NoArgsConstructor

public class ProductCategoryDto {

private Long id;

private String name;

}

@Data

@AllArgsConstructor

@NoArgsConstructor

public class ManufacturerDto {

private Long id;

private String code;

private String name;

private String website;

}

@Data

@AllArgsConstructor

@NoArgsConstructor

public class ZipCodeDto {

private Long id;

private String zipCode;

private String city;

}

@Data

@AllArgsConstructor

@NoArgsConstructor

public class SerialNumberDto {

private Long id;

private String serial;

private boolean used;

private Long productId;

private Long inventoryItemId;

}

@Data

@AllArgsConstructor

@NoArgsConstructor

public class BarcodeDto {

private Long id;

private String code;

private Long productId;

}

@Data

@AllArgsConstructor

@NoArgsConstructor

public class InventoryItemDto {

private Long id;

private Long inventoryId;

private Long productId;

private int quantity;

private BigDecimal netPrice;

private BigDecimal grossPrice;

private Integer warrantyPeriodMonths;

}

@Data

@AllArgsConstructor

@NoArgsConstructor

public class InventoryDto {

private Long id;

private Long supplierId;

private LocalDateTime receivedAt;

private List<InventoryItemDto> items;

}

@Data

@AllArgsConstructor

@NoArgsConstructor

public class InvoiceItemDto {

private Long id;

private Long invoiceId;

private Long productId;

private int quantity;

private String unit;

private BigDecimal unitPrice;

private BigDecimal discountPercent;

private BigDecimal netAmount;

private BigDecimal grossAmount;

}

@Data

@AllArgsConstructor

@NoArgsConstructor

public class InvoiceDto {

private Long id;

private Long sellerId;

private Long buyerId;

private LocalDate issuedAt;

private LocalDate fulfillmentAt;

private LocalDate dueDate;

private String invoiceType;

private String paymentMethod;

private BigDecimal netTotal;

private BigDecimal grossTotal;

private List<InvoiceItemDto> items;

}

//ProductDto

package dev.lkeleti.invotraxapp.dto;  
  
import java.math.BigDecimal;  
import java.util.List;  
  
@AllArgsConstructor  
@NoArgsConstructor  
@Getter  
@Setter

public class ProductDto {  
 private Long id;  
 private String name;  
 private String sku;  
 private String description;  
 private ProductCategoryDto category;  
 private ManufacturerDto manufacturer;  
 private BigDecimal netPrice;  
 private BigDecimal grossPrice;  
 private Integer warrantyPeriodMonths;  
 private boolean serialNumberRequired;  
 private List<BarcodeDto> barcodes;  
 private List<SerialNumberDto> serialNumbers;  
}

//PartnerDto

package dev.lkeleti.invotraxapp.dto;  
  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
import java.math.BigDecimal;  
import java.time.LocalDateTime;  
  
@AllArgsConstructor  
@NoArgsConstructor  
@Getter  
@Setter  
public class PartnerDto {  
 private Long id;  
 private String name;  
 private ZipCodeDto zipCode;  
 private String streetType;  
 private String houseNumber;  
 private String building;  
 private String staircase;  
 private String floor;  
 private String door;  
 private String landRegistryNumber;  
 private boolean isPrivate;  
 private String taxNumber;  
 private String email;  
 private String phoneNumber;  
 private String preferredPaymentMethod;  
 private LocalDateTime createdAt;  
 private BigDecimal balance;  
 private Integer defaultPaymentDeadline;  
 private boolean deleted;  
}

// =============================

// REPOSITORY INTERFÉSZEK

// =============================

package com.invotraxapp.repository;

import com.invotraxapp.model.\*;

import org.springframework.data.jpa.repository.JpaRepository;

public interface ProductRepository extends JpaRepository<Product, Long> {}

public interface ManufacturerRepository extends JpaRepository<Manufacturer, Long> {}

public interface ZipCodeRepository extends JpaRepository<ZipCode, Long> {}

public interface PartnerRepository extends JpaRepository<Partner, Long> {}

public interface InventoryRepository extends JpaRepository<Inventory, Long> {}

public interface InventoryItemRepository extends JpaRepository<InventoryItem, Long> {}

public interface SerialNumberRepository extends JpaRepository<SerialNumber, Long> {}

public interface BarcodeRepository extends JpaRepository<Barcode, Long> {}

public interface InvoiceRepository extends JpaRepository<Invoice, Long> {}

public interface InvoiceItemRepository extends JpaRepository<InvoiceItem, Long> {}

public interface ProductCategoryRepository extends JpaRepository<ProductCategory, Long> {}

// =============================

// CONTROLLEREK

// =============================

package com.invotraxapp.controller;

import com.invotraxapp.dto.\*;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/api/products")

public class ProductController {

@GetMapping

public List<ProductDto> getAllProducts() {

// implementáció

return List.of();

}

@GetMapping("/{id}")

public ProductDto getProductById(@PathVariable Long id) {

return new ProductDto();

}

@PostMapping

public ProductDto createProduct(@RequestBody ProductDto dto) {

return dto;

}

@PutMapping("/{id}")

public ProductDto updateProduct(@PathVariable Long id, @RequestBody ProductDto dto) {

return dto;

}

@DeleteMapping("/{id}")

public void deleteProduct(@PathVariable Long id) {

// törlés

}

}

@RestController

@RequestMapping("/api/inventory")

public class InventoryController {

@GetMapping

public List<InventoryDto> getAllInventories() {

return List.of();

}

@PostMapping

public InventoryDto createInventory(@RequestBody InventoryDto dto) {

return dto;

}

}

@RestController

@RequestMapping("/api/invoice")

public class InvoiceController {

@GetMapping

public List<InvoiceDto> getAllInvoices() {

return List.of();

}

@PostMapping

public InvoiceDto createInvoice(@RequestBody InvoiceDto dto) {

return dto;

}

@GetMapping("/{id}")

public InvoiceDto getInvoiceById(@PathVariable Long id) {

return new InvoiceDto();

}

}

@RestController

@RequestMapping("/api/partners")

public class PartnerController {

@GetMapping

public List<PartnerDto> getAllPartners() {

return List.of();

}

}

@RestController

@RequestMapping("/api/barcodes")

public class BarcodeController {

@GetMapping("/generate")

public BarcodeDto generateBarcode() {

return new BarcodeDto();

}

}