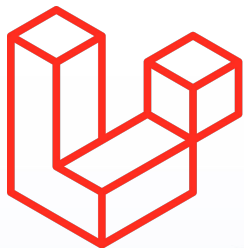


# **EKT projekts – Datora komplektēšanas sistēma**

**Kristaps Kārkliņš**

Ventspils Tehnikums, 2025

# Izstrādes līdzekļu izvēle



# Izstrādes līdzekļu izvēle



Kāpēc php kopā ar Laravel?

1. Tīmekļa risinājums, nevis darbvirsmas lietojumprogramma
2. Praktiska pieredze un izpratne
3. Laravel strukturēts ietvars

# Izstrādes līdzekļu izvēle



## Kāpēc MySQL?

1. Atbalsts vairākiem vienlaicīgiem pieprasījumiem
2. Plaša izmantošana un saderība ar Laravel

# Izstrādes modeļu izvēle



## Izmantots iteratīvais izstrādes modelis

Katrā iterācijā tiek izveidota jauna, funkcionāli pabeigta sistēmas daļa, kas tiek pārbaudīta un testēta. Pēc veiksmīgas testēšanas tai tiek pievienotas jaunas darbības un uzlabojumi. Palīdz pakāpeniski paplašināt darbu un savlaicīgi identificēt kļūdas

# Datu struktūras

Centrālā tabula products ar komponentes tipu

Komponentšu specifikācija tabulas:

- Processoru specifikācija

cpu_specs
123 product_id
AZ manufacturer
AZ series
AZ socket
123 cpu_speed_ghz
123 wattage_w
AZ created_at
AZ updated_at

motherboard_specs
123 product_id
AZ manufacturer
AZ series
AZ socket
AZ chipset
AZ memory_technology
AZ form_factor
AZ created_at
AZ updated_at

ram_specs
123 product_id
AZ manufacturer
AZ series
AZ memory_type
123 capacity_gb
123 modules
123 speed_mhz
123 base_speed_mhz
123 cas_latency
123 voltage_v
AZ created_at
AZ updated_at

products
123 id
AZ name
AZ type
AZ created_at
AZ updated_at



# Algoritmu un metožu realizācija



```
class ProductTypeRegistry
{
    protected static array $typeMap = [
        'motherboard' => MotherboardSpec::class,
        'cpu' => CpuSpec::class,
        'ram' => RamSpec::class,
    ];
    public static function getModel(string $type): ?string
    {
        return self::$typeMap[$type];
    }
    public static function exists(string $type): bool
    {
        return isset(self::$typeMap[$type]);
    }
    public static function returnTypes(): array
    {
        return array_keys(self::$typeMap);
    }
}
```

ProductTypeRegistry			
- typeMap : array<string, Class>			
+ getModel(type : string) : Class			
+ exists(type : string) : bool			
+ returnTypes() : array			

Tiek realizēta dinamiska komponentu tipu sasaistīšana ar atbilstošajiem specifikāciju datu modeļiem. Sistēma spēj noteikt, kurš datu modelis jāizmanto katram komponentes tipam.

# Algoritmu un metožu realizācija



```
class Build
{
    private $items = null;

    public function __construct($oldcart)
    {
        $this->items = $oldcart->items ?? [];
    }
}
```

Konfigurācijas loģika ir ieviesta neatkarīgi no Laravel sesiju mehānisma, izmantojot atsevišķu Build klasi. Šī pieeja ļauj viegli testēt sistēmas darbību un atkārtoti izmantot loģiku dažādos kontekstos.

Build	
- items: array	
+ __construct(oldCart: Build  null)	
+ hasItem(type : string) : bool	
getSpec(type : string) : Spec [0..*]	
+ getField(type : string, field : string): string	
+ getProduct(type : string): Product	
+ getItems() : array	
+ addItem(type : string, id)	
+ loadModel(type : string) : Spec [0..*]	

# Saskarnes



Navigācijas josla nodrošina piekļuvi galvenajām sistēmas sadaļām, piemēram, produktu sarakstam un komplektēšanas rīkam (Builder).

PRODUCTS

BUILDER

# Saskarnes



Skatā lietotājs izvēlas komponentu tipu (piemēram, pamatplate, procesors vai operatīvā atmiņa). Izvēle nosaka, kādi komponenti tiks ielādēti nākamajā solī, nodrošinot strukturētu un loģisku darba plūsmu.

## Choose Product Type

GET ALL MOTHERBOARD

GET ALL CPU

GET ALL RAM

# Saskarnes



Sistēma attēlo izvēlētā komponentu tipa produktu sarakstu tabulas veidā. Lietotājs var apskatīt katra produkta detaļas vai pievienot to konfigurācijai, tādējādi veidojot savu datora komplektu.

rem	<a href="#">View Details</a>	Add
aut	<a href="#">View Details</a>	Add
ullam	<a href="#">View Details</a>	Add

nulla	<a href="#">View Details</a>	Add
rem	<a href="#">View Details</a>	Add
non	<a href="#">View Details</a>	Add
aut	<a href="#">View Details</a>	Add

Atkarībā no lietotā konfigurācijas saraksts tiek pielāgots, attiecīgi lietotāja izvēlem



Šajā skatā tiek parādīta konkrētā produkta detalizēta informācija un tehniskās specifikācijas. Tas ļauj lietotājam salīdzināt parametrus un pieņemt lēmumu par komponentes izvēli.

iure

MOTHERBOARD

iure

Specifications

SOCKET	LGA1151
CHIPSET	B550
FORM FACTOR	ATX
MEMORY TYPE	DDR5



Komplektācijas skatā tiek attēloti visi pievienotie komponenti pa tiem. Lietotājs var pievienot vai aizstāt komponentes.

COMPONENT	SELECTION
Motherboard	<a href="#">+ Add Motherboard</a>
Cpu	<a href="#">+ Add Cpu</a>
Ram	<a href="#">+ Add Ram</a>

COMPONENT	SELECTION
Motherboard	iure
Cpu	<a href="#">+ Add Cpu</a>
Ram	<a href="#">+ Add Ram</a>



Prasība Nr. 1: Sistēmai jāspēj ielādēt procesora komponenti un tās specifikāciju, izmantojot komponentes tipu.

Testi:

- Procesora ielāde, ja konfigurācijā ir pievienots CPU ar korektu ID.

Rezultāts: sistēma atgriež pamatplates specifikācijas modeli ar piesaistītu produktu.

- Procesora ielāde, ja konfigurācijā nav pievienots CPU.

Rezultāts: sistēma atgriež null vērtību.

```
it('returns null when trying to load a non-existing product', function () {  
    $build = new Build(null);  
    $build->addItem('cpu', 999); // ID does not exist  
  
    $result = $build->loadModel('cpu');  
  
    expect($result)->toBeNull();  
});
```



Prasība Nr. 6: Sistēmai jāspēj pārrakstīt komponenti, ja viens un tas pats tips tiek pievienots atkārtoti.

Testi:

- Konfigurācijai tiek pievienoti divi CPU pēc kārtas ar dažādiem ID.  
Rezultāts: konfigurācijā tiek saglabāts tikai pēdējais pievienotais CPU.

```
it('overwrites an item when adding same type twice', function () {  
  $build = new Build(null);  
  
  $build->addItem('cpu', 1);  
  $build->addItem('cpu', 2);  
  
  $items = $build->getItems();  
  
  expect($items['cpu'])->toBe(['product_id' => 2]);  
});
```



Darba gaitā tiek veidota sistēmas loģika, izmantojot dinamisku programmēšanas pieeju. 7.slaid.

# **EKT projekts – Datora komplektēšanas sistēma**

**Kristaps Kārkliņš**

Ventspils Tehnikums, 2025