

# Sprint 3

## Products and Pricing

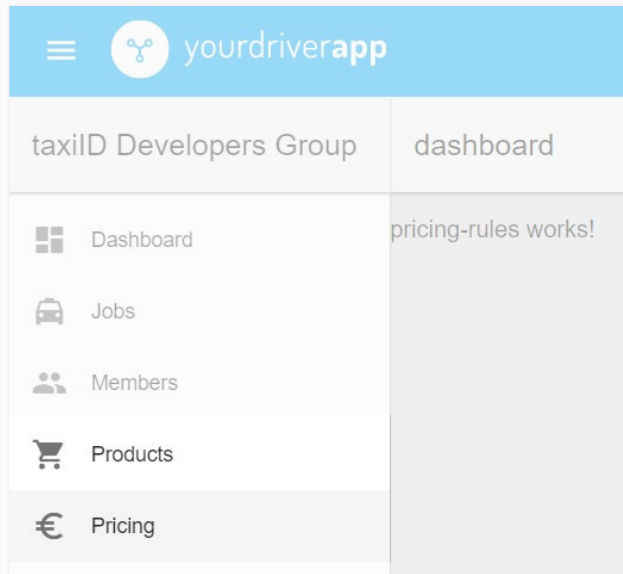
Products and pricing rules, automatically managing the many-to-many relations

# Views

During this sprint, the portal project was set up for development, and two categories were added to the group sidebar:

1. Products
2. Pricing

The following slides show the views that have been implemented



# Products Overview

The products overview shows products per vehicle type

Products can be filtered and sorted

Clicking on a products will bring the user to the detail page

The button in the bottom right corner creates a new vehicle that can be edited and saved in the detail page

The screenshot displays the 'yourdriverapp' interface. The top navigation bar is blue with the app name and a menu icon. The sidebar on the left lists navigation options: Dashboard, Jobs, Members, Products (highlighted), and Pricing. The main content area is titled 'Products' and features a search bar and filter tabs for vehicle types: all (28), saloon (4), estate (9), minivan (6) (selected), bus (5), and limo (4). Below the filters is a table with columns for 'name', 'maxPassengers', and 'type'. The table contains six rows of product data. At the bottom right of the table area is a blue button with a white plus sign. The footer shows copyright information for 2016-2018.

name	maxPassengers	type
asdfsdf	342	minivan
Cotton salmon car	8	minivan
Minivan	6	minivan
Minivan	6	minivan
aasdfsdf	2	minivan
Fresh Ivory car	1	minivan

# Products Detail Page

The products detail page shows the properties that belong to the product

The product may be deleted, upon which a confirmation screen protects the user against accidentals

The screenshot displays the 'yourdriverapp' interface. The top navigation bar is blue with the app logo and name. Below it, a grey bar shows the user 'taxiID Developers Group' and the current section 'Products'. A left sidebar contains navigation links: Dashboard, Jobs, Members, Products (active), and Pricing. The main content area is titled 'Expensive Product' and contains two form fields: 'Name' with the value 'Expensive Product' and 'Max Passengers' with the value '30'. To the right of the form is a dropdown menu with options: saloon, estate, minivan (highlighted in blue), bus, and limo. At the bottom of the form area are two buttons: 'save\_product\_button' (dark grey) and 'delete\_product\_button' (red). The footer contains the copyright notice 'Copyright © 2016 - 2018 YourDriverPortal. All rights reserved'.

yourdriverapp

taxiID Developers Group Products

Dashboard  
Jobs  
Members  
Products  
Pricing

Expensive Product

Name  
Expensive Product

Max Passengers  
30

saloon  
estate  
minivan  
bus  
limo

save\_product\_button delete\_product\_button

Copyright © 2016 - 2018 YourDriverPortal. All rights reserved

# Pricing Overview

The pricing overview has similar characteristics to that of the products overview

☰

yourdriverapp

taxiID Developers Group

Pricing/rules

Dashboard

Jobs

Members

Products

Pricing

Search here

€ all (3)   € dynamic (3)   € fixed (0)

name	type	enabled	precedence
Empty RULE	dynamic	<input checked="" type="checkbox"/>	
Empty rule	dynamic	<input checked="" type="checkbox"/>	
Empty rule	dynamic	<input type="checkbox"/>	

Row per page: 1-3 of 3   |◀ < > ▶|

+

Copyright © 2016 - 2018 YourDriverPortal. All rights reserved

5

# Pricing Detail Page

The detail page displays complex properties for each company product

Threshold prices have not been implemented fully, as can be seen in the lowest row of the table

When the table overflows, it becomes scrollable in the horizontal direction

**yourdriverapp**

taxiID Developers Group

**Empty RULE**

Name: Empty RULE    Type: dynamic    Precedence: 0    ☒ Enabled

	Stefan	test	Expensive Prod...	Cheap product
<b>Product</b>				
Enabled	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Cascading	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Waiting price	70	70	70	70
Minimum price	1	0	0	0
Start price	1	123	0	0
Kilometer price	1	2	2	0
Minute price +	1	0	0	0
after 100 duration —	1	0	0	0
after 100 dynamic —	100			

save\_rule\_button    delete\_rule\_button

Copyright © 2016 - 2018 YourDriverPortal. All rights reserved

# Price Calculator Classes

Improvements have been made to the DynamicCalculator and FixedCalculator classes that are responsible for:

1. The reading of pricing information
2. Define which metrics are calculated
3. Contain definition of a calculation
4. Generate high order calculator functions with default params
5. Call the calculate total function that calculates the total price using the generated calculators
6. Return the appropriate final amount

# Dynamic Calculator

```
/**
 * Main calculation for a dynamic calculation.
 */
public static calculate(
    pricing: pricing,
    metrics: metrics & indexed,
): number {

    const thresholds: threshold[] = pricing.prices.dynamicThresholds;
    const startAmount = pricing.prices.dynamicStartPrice;
    const minAmount = pricing.prices.dynamicMinimumPrice;
    const cascaded = pricing.prices.cascadingThresholdCalculation;

    // Price for each metric
    const prices: indexed = {
        distance: pricing.prices.dynamicDistancePrice,
        duration: pricing.prices.dynamicMinutePrice,
    };
}
```

...

```
...
// Explain how the price should be calculated
const func = (price: number, unit: number) => price * unit;

// Calculator for each metric using prices as default params
const calculators = Calculator.generateCalculators(
    prices, metrics, func);

// If thresholds have been provided, check if some metrics have
// surpassed these thresholds per metric type, and calculate
// the price using a provided function for highest or all
// thresholds.
const total = Price.total(
    metrics, thresholds, calculators, cascaded);

// Return biggest of the two
return Math.max(total + startAmount, minAmount);
}
```



# Fixed Calculator

```
/**
 * Main calculation for a fixed calculation.
 */
public static calculate(
    pricing: pricing,
    metrics: metrics & indexed,
): number {

    const thresholds: threshold[] = pricing.prices.fixedThresholds;
    const cascaded = false;

    // Price for distance metric only
    const prices: indexed = {
        distance: pricing.prices.fixedPrice,
        duration: 0,
    }

    ...
```

```
...
// Explain how the price should be calculated
const func = (price: number, unit?: number) => price;

// Calculator for each metric using prices as default params
const calculators = Calculator.generateCalculators(
    prices, metrics, func);

// If thresholds have been provided, check if some metrics have
// surpassed these thresholds per metric type, and calculate
// the price using a provided function for highest or all
// thresholds.
const total = Price.total(
    metrics, thresholds, calculators, cascaded);

// Return biggest of the two
return Math.max(total, 0);
}
```

# Meter Calculator

```
/**  
 * Main calculation for a on meter calculation.  
 */  
public static calculate(  
    pricing: pricing,  
    metrics: metrics & indexed,  
): number {  
    return 0;  
}
```

# Data Management

1. When a product is created, pricings (pricing, dynamicPricing, fixedPricing) are automatically attached to each rule.
2. When a rule is created, pricings are created for each product.

# Reflection - Must Haves

1. Thresholds that are incrementally bigger should be added to each pricing of a rule if the add button is pressed.
2. A threshold of a rule should be deleted for each pricing with the click of a button.
3. An error is sometimes shown when a particular combination of fields is mutated and saved.
4. Display prices in € instead of cents.
5. Test pricing calculation with data inserted by the user.

# Reflection - Could Haves

1. Before leave warning, when a user has modified a form.
2. Order rules by precedence automatically.
3. Set precedence by dragging rules up and down.