# Sprint 3
# Products and Pricing

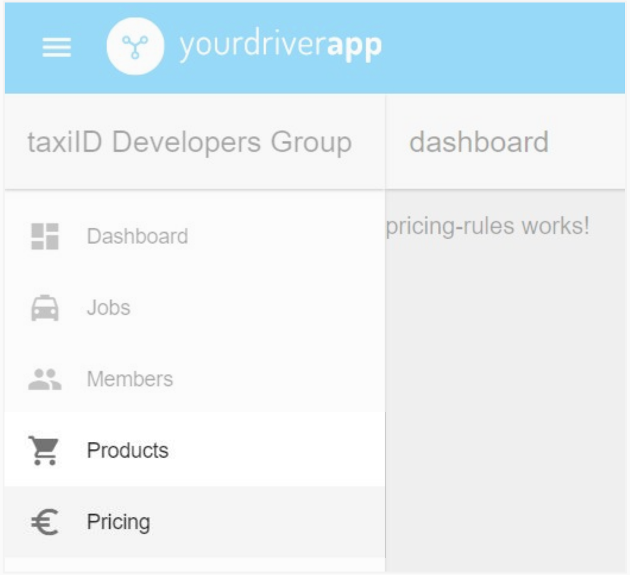
Products and pricing rules, automatically managing the many-to-many relations

# 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



pricing-rules works!

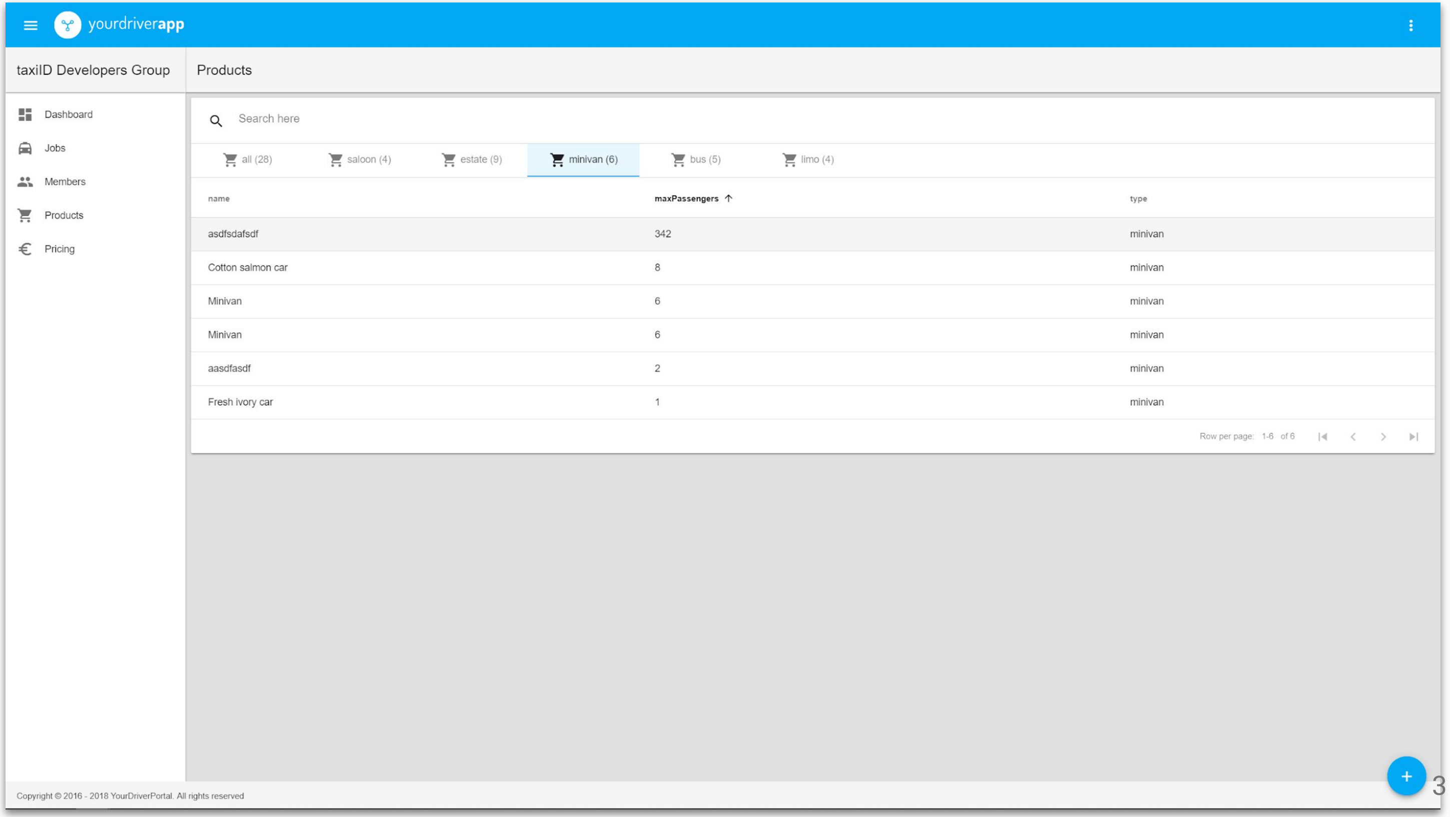
# Products Overview

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



| name | maxPassengers ↑ | type |
|---|---|---|
| asdfsdafsdf | 342 | minivan |
| Cotton salmon car | 8 | minivan |
| Minivan | 6 | minivan |
| Minivan | 6 | minivan |
| aasdfasdf | 2 | minivan |
| Fresh ivory car | 1 | minivan |

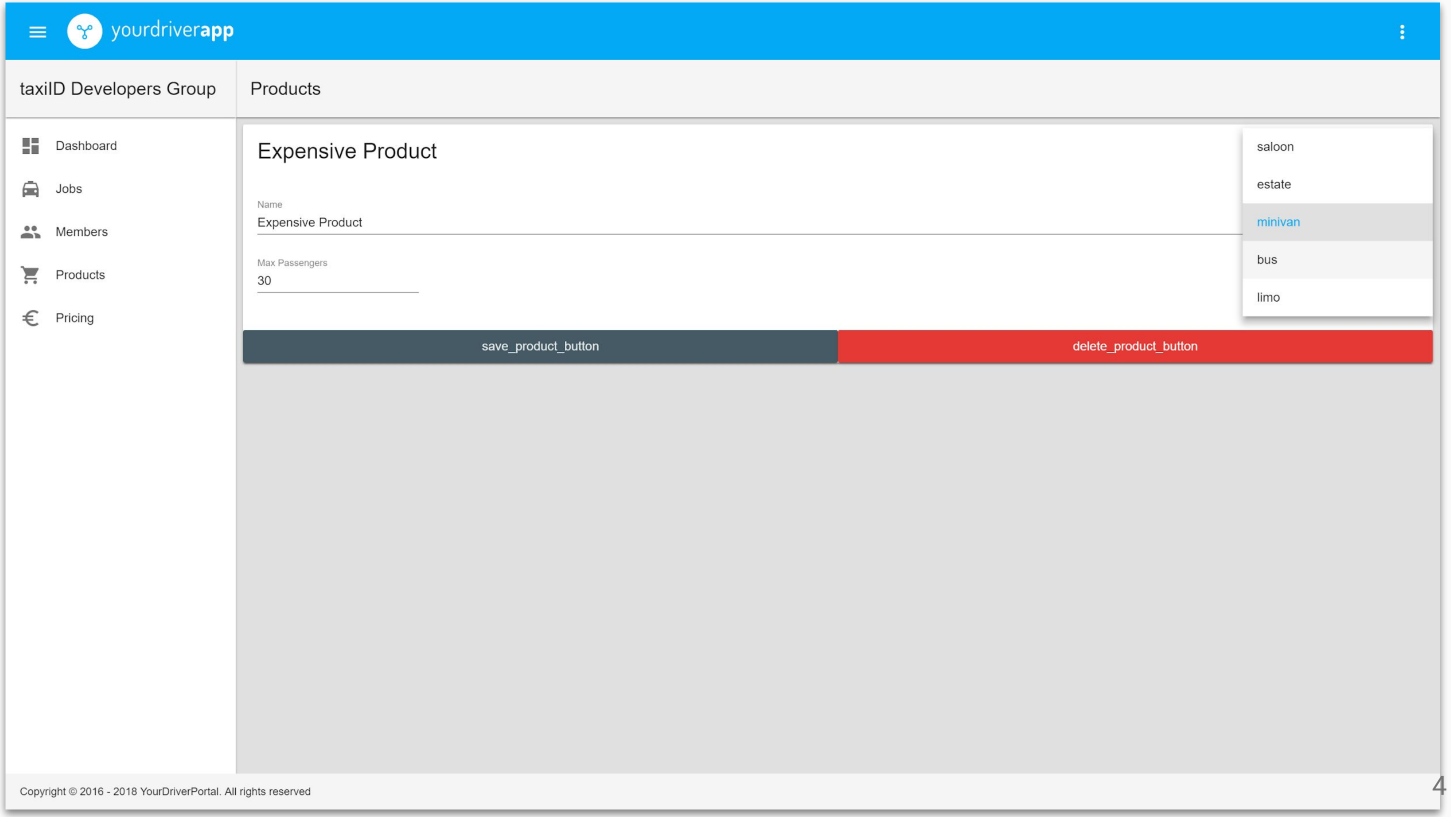
Row per page: 1-6 of 6

Copyright © 2016 - 2018 YourDriverPortal. All rights reserved

# 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



Expensive Product

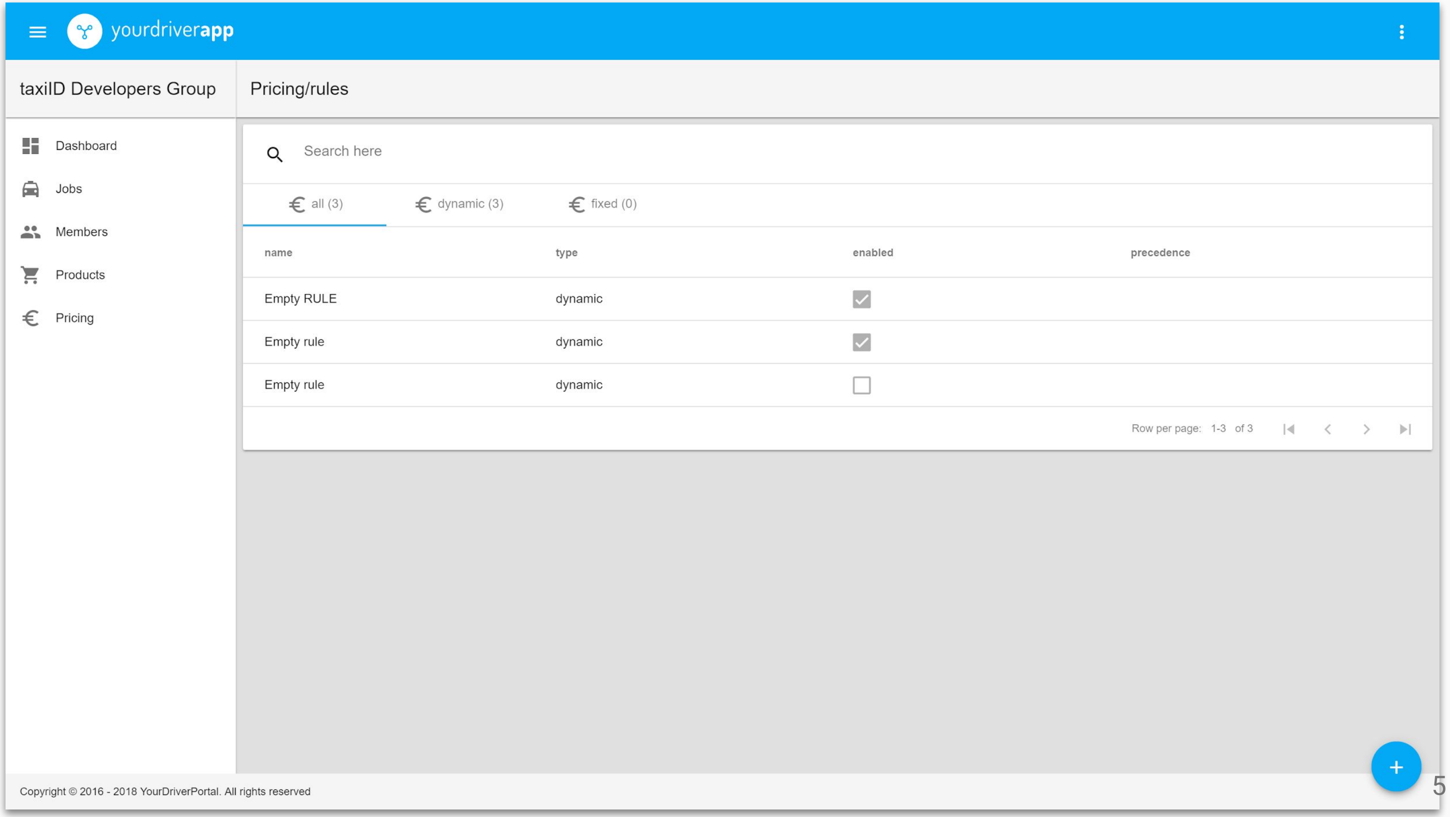
Name
Expensive Product

Max Passengers
30

Copyright © 2016 - 2018 YourDriverPortal. All rights reserved

# Pricing Overview

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



| name | type | enabled | precedence |
|---|---|---|---|
| Empty RULE | dynamic | ✓ | |
| Empty rule | dynamic | ✓ | |
| Empty rule | dynamic | ☐ | |

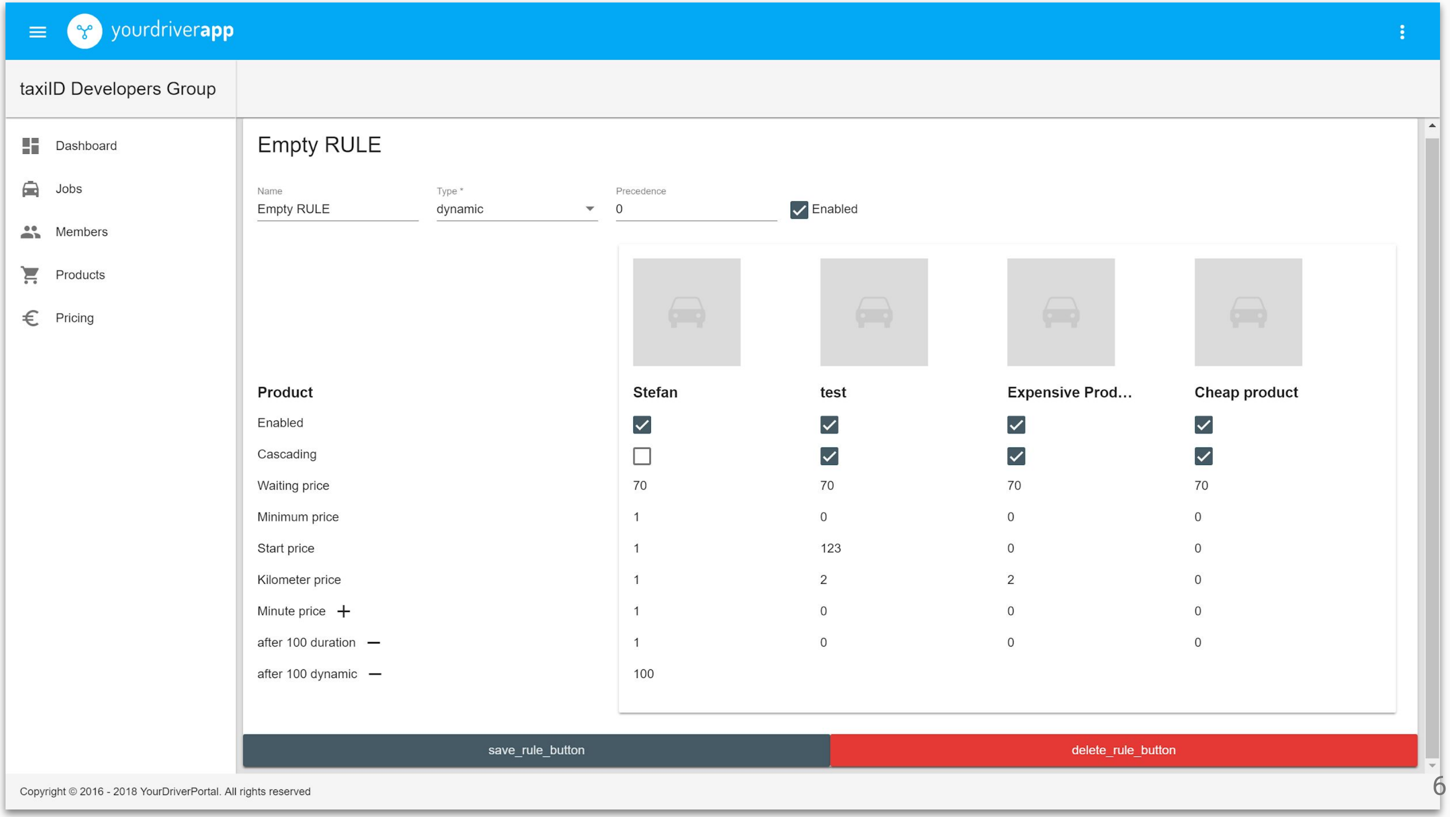
Copyright © 2016 - 2018 YourDriverPortal. All rights reserved

# 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



Empty RULE

Name: Empty RULE
Type *: dynamic
Precedence: 0
✓ Enabled

| Product | Stefan | test | Expensive Prod... | Cheap product |
|---|---|---|---|---|
| Enabled | ✓ | ✓ | ✓ | ✓ |
| Cascading | ☐ | ✓ | ✓ | ✓ |
| 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 | | | |

## 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 defintion 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.