

Rita Castro

Building the Right Product and Building It Right

A glimpse into XP, Atomic Design and Micro Frontends

Hello World We Are Developers!



Building the Right Product and Building It Right



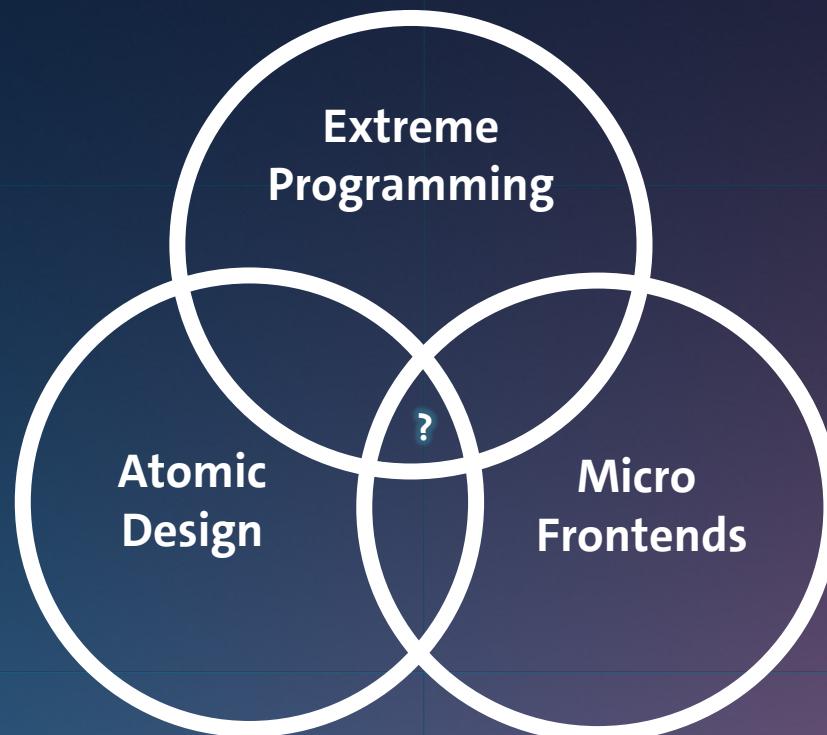
Building the Right Product and Building It Right

**Extreme
Programming**

Atomic Design

Micro Frontends

Building the Right Product and Building It Right





SOFTWARE
DEVELOPMENT
CENTER:LISBON

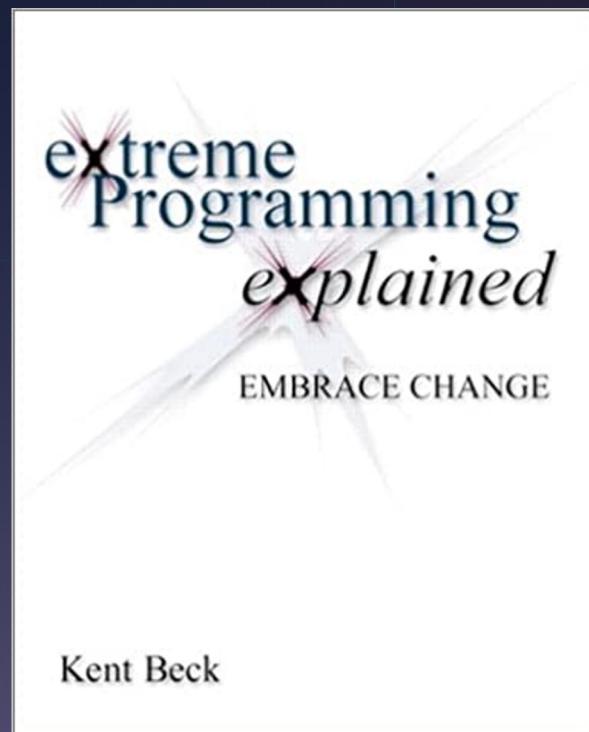


Balanced Teams



Extreme Programming

XP is an agile software development framework that aims to produce higher quality software, and increase the quality of life for the development team.



XP Values



Communication



Simplicity



Feedback



Courage



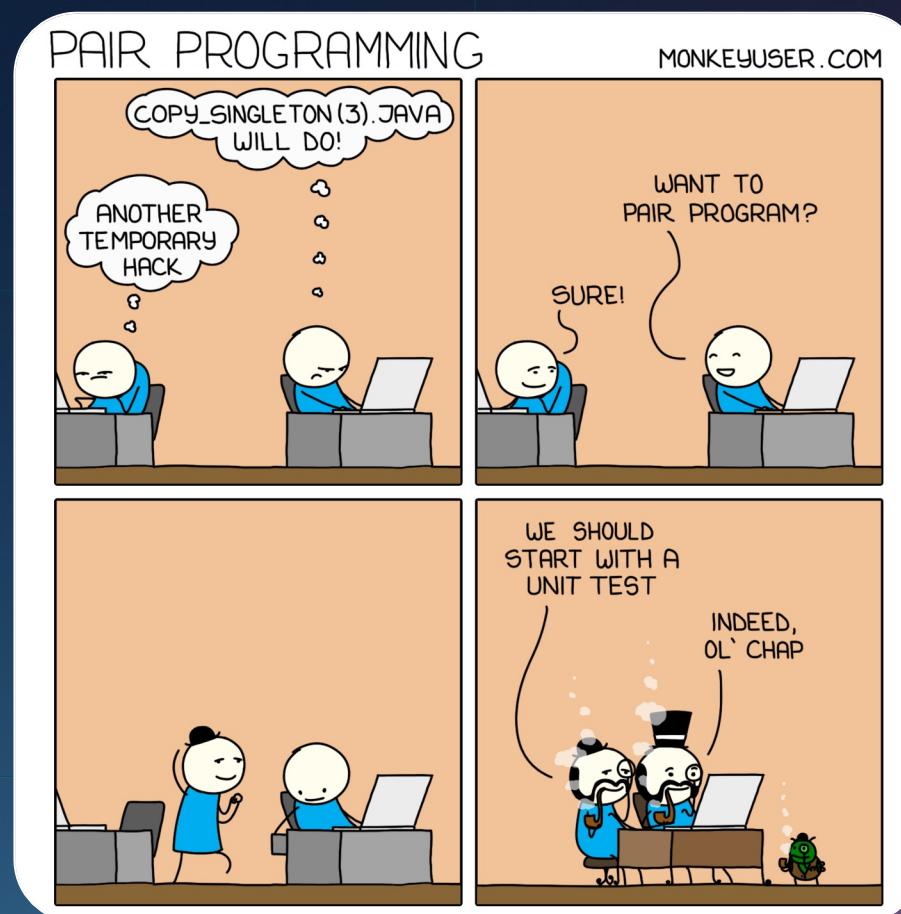
Respect

VWAG Group Essentials



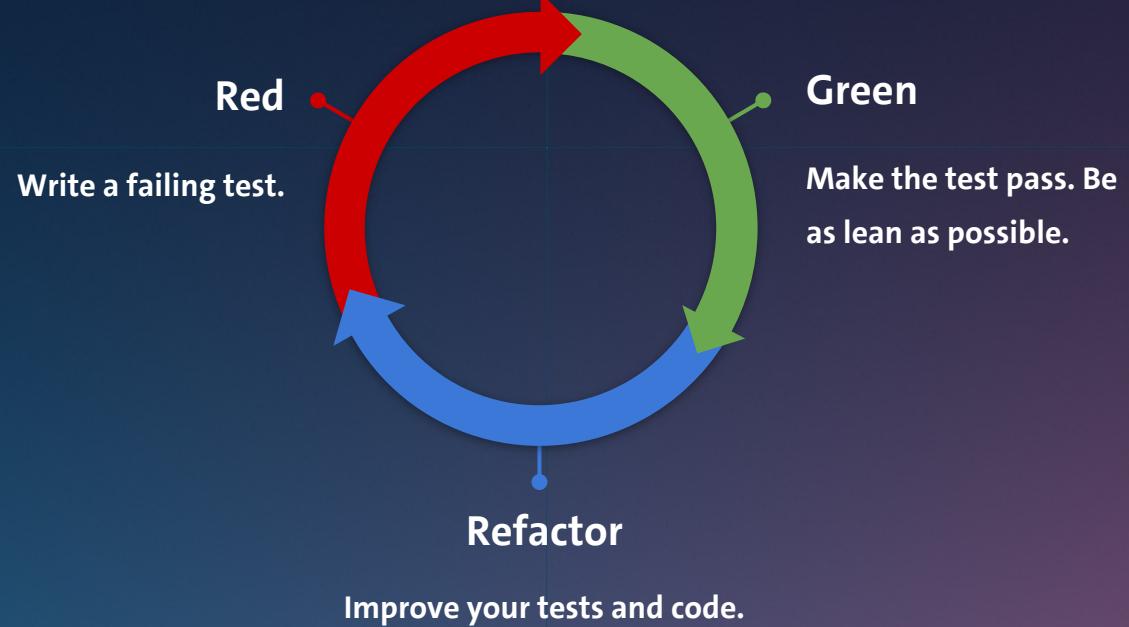
XP Practices

Pair Programming



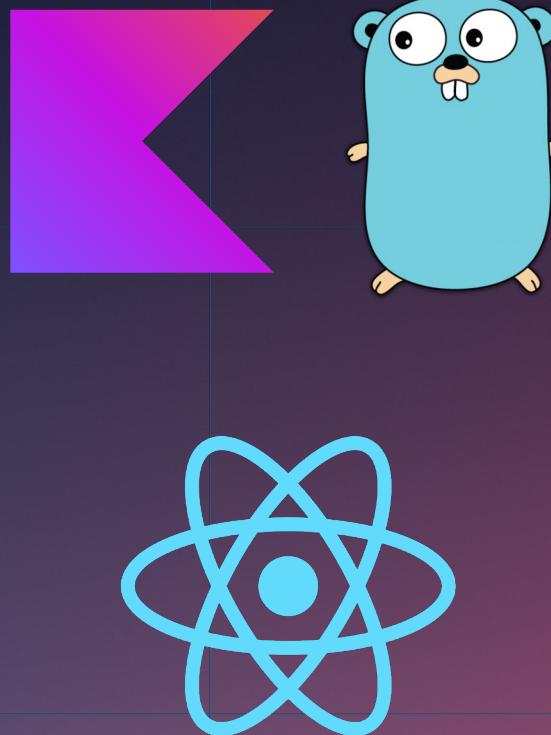
XP Practices

Test Driven Development

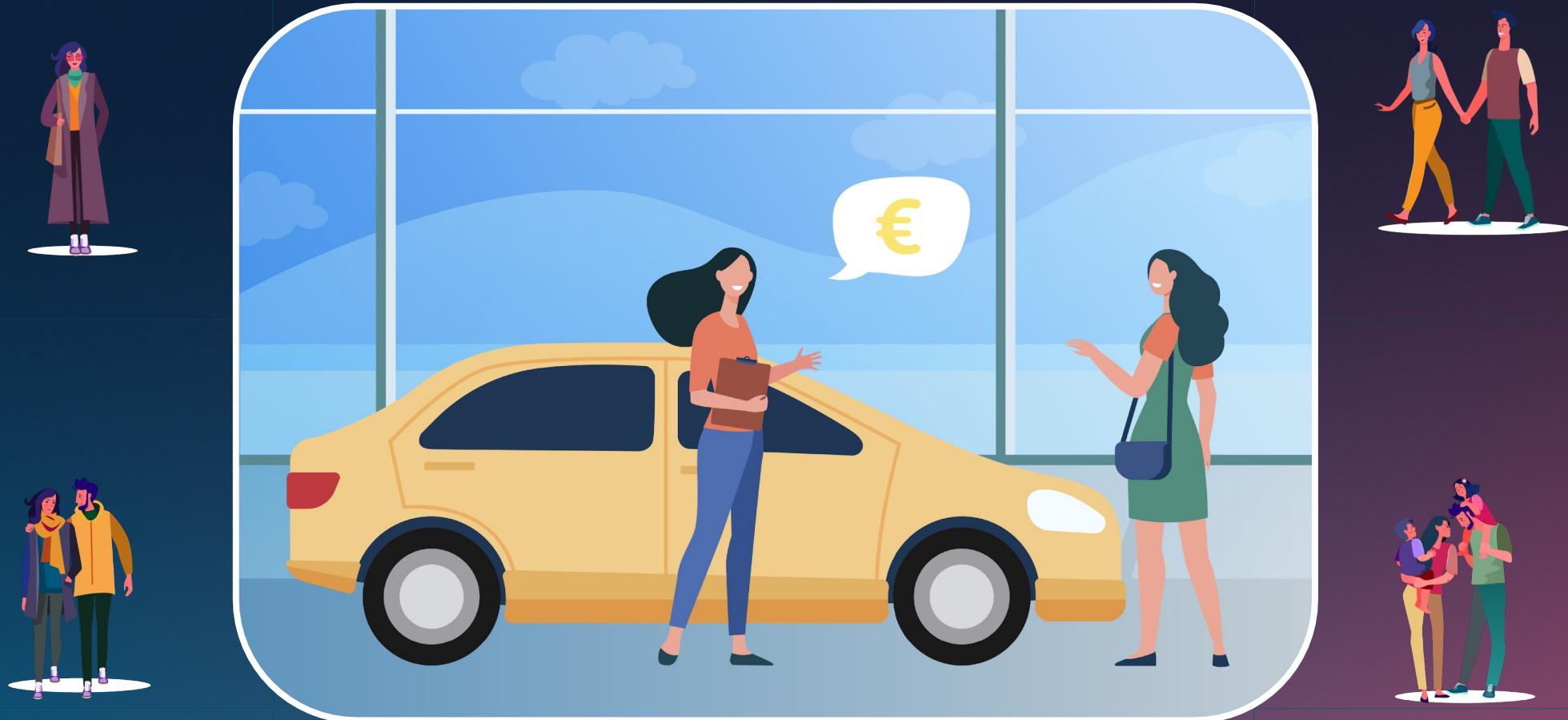


Pre-Delivery Enrolment

Drives adoption for multiple digital products in VW's Group





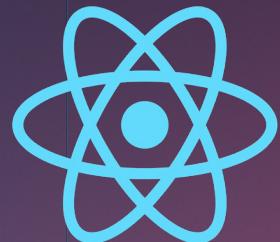




Car Tracker



**Enable the dealer to create stronger
relationships with customers by
removing the annoyances of finding
information about their orders**



- ➔ Clear mission
- ➔ Narrow scope
- ➔ Autonomous full stack teams
- ➔ Independent deployments
- ➔ Customer focus
- ➔ Scaling gets easier



Teamwork people vector created by pch.vector - www.freepik.com [Adapted]

Micro Frontends

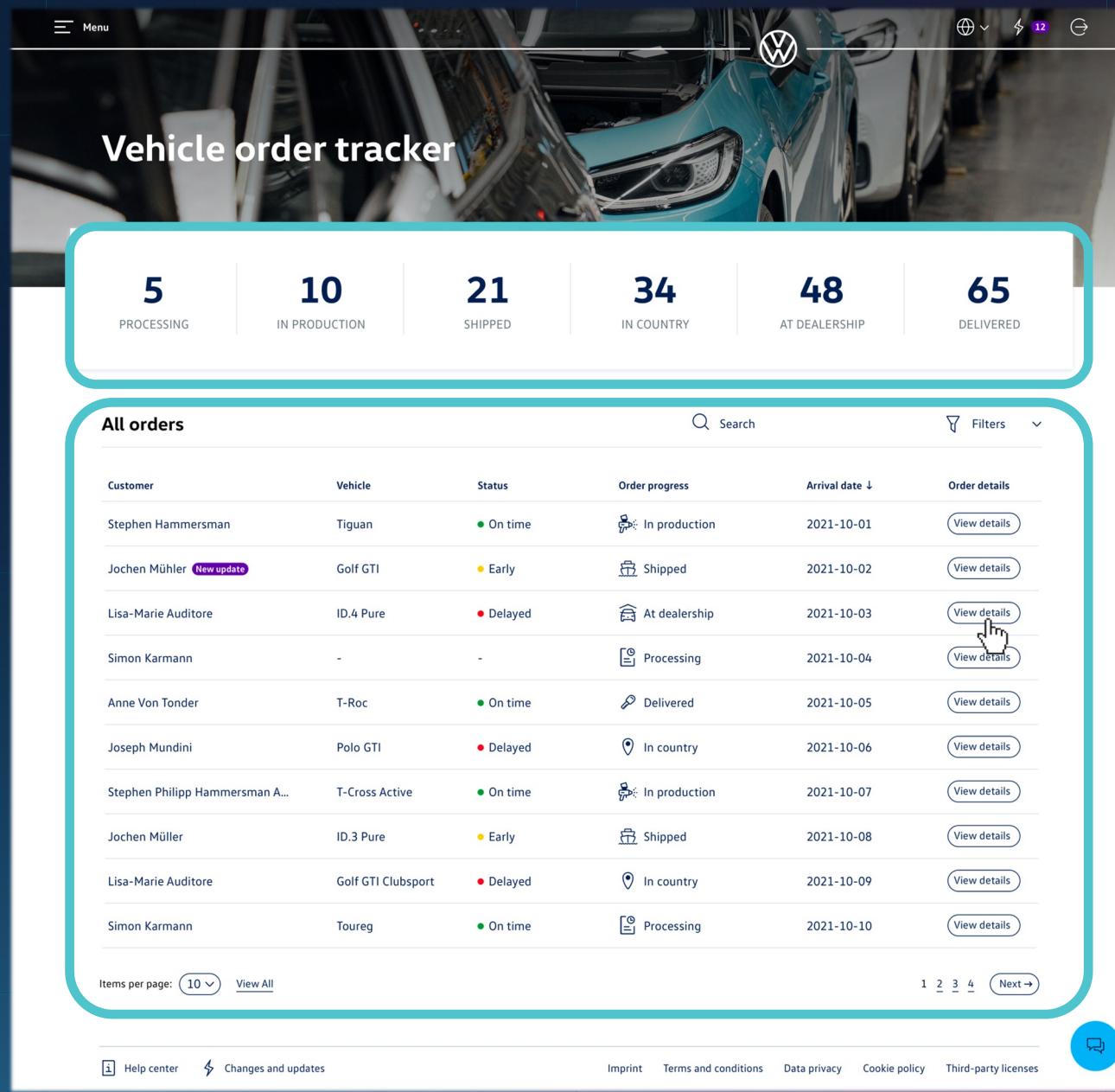
“An architectural style where independently deliverable frontend applications are composed into a greater whole.” [1]

“... not a concrete technology. They’re an alternative organizational and architectural approach.” [2]



[1] Cam Jackson, <https://martinfowler.com/articles/micro-frontends.html>

[2] Michael Geers, *Micro Frontends in Action*, Manning Publications, Nov 2020



The image shows a screenshot of a vehicle order tracker application. At the top, there's a banner with the text "Vehicle order tracker" and a background image of a teal-colored Volkswagen car. Below the banner is a summary section with six status indicators:

PROCESSING	IN PRODUCTION	SHIPPED	IN COUNTRY	AT DEALERSHIP	DELIVERED
5	10	21	34	48	65

The "DELIVERED" section is highlighted with a blue rounded rectangle. Below this is a table titled "All orders" showing ten rows of order details. The columns are: Customer, Vehicle, Status, Order progress, Arrival date, and Order details. The "Order details" column for the fourth row (Simon Karmann) has a cursor hovering over it. At the bottom, there are pagination controls and links for Help center, Changes and updates, Imprint, Terms and conditions, Data privacy, Cookie policy, and Third-party licenses.

All orders

Customer	Vehicle	Status	Order progress	Arrival date	Order details
Stephen Hammersman	Tiguan	● On time	🕒 In production	2021-10-01	View details
Jochen Mühler New update	Golf GTI	● Early	🕒 Shipped	2021-10-02	View details
Lisa-Marie Auditore	ID.4 Pure	● Delayed	🕒 At dealership	2021-10-03	View details
Simon Karmann	-	-	🕒 Processing	2021-10-04	View details
Anne Von Tonder	T-Roc	● On time	🕒 Delivered	2021-10-05	View details
Joseph Mundini	Polo GTI	● Delayed	📍 In country	2021-10-06	View details
Stephen Philipp Hammersman A...	T-Cross Active	● On time	🕒 In production	2021-10-07	View details
Jochen Müller	ID.3 Pure	● Early	🕒 Shipped	2021-10-08	View details
Lisa-Marie Auditore	Golf GTI Clubsport	● Delayed	📍 In country	2021-10-09	View details
Simon Karmann	Toureg	● On time	🕒 Processing	2021-10-10	View details

Items per page: [10](#) [View All](#) 1 2 3 4 [Next →](#)

[Help center](#) [Changes and updates](#) [Imprint](#) [Terms and conditions](#) [Data privacy](#) [Cookie policy](#) [Third-party licenses](#)

☰ Menu

 ⊕ ↘ 12 ⌂

← Back

Golf GTI for Jochen Mühler

Share

Order details

Data fetched from commission number

Status

● Early

Note

You cannot make changes to this order

Customer

Jochen Mühler

Model

Golf GTI 2.0 l TSI 180 kW (245 hp)
7-speed dual-clutch transmission DSG

Colour

Deep Black Pearlescent (2T) Exterior
Deep Black Pearlescent (2T) Roof
Titanium Black (T0) Interior

Enrollment status

 Volkswagen ID

Commission number (CN)

182 NVCNXX 2020

Vehicle identification number (VIN)

1 HG BO 41 JX MN 50 90 98

Order progress

Data might have a delay of up to 24 hours

Processing

Order date 2021-10-21

In production

Completed on 2021-10-21

Shipped

Started on 2021-10-21

In country

At dealership

Estimated for 2021-10-21

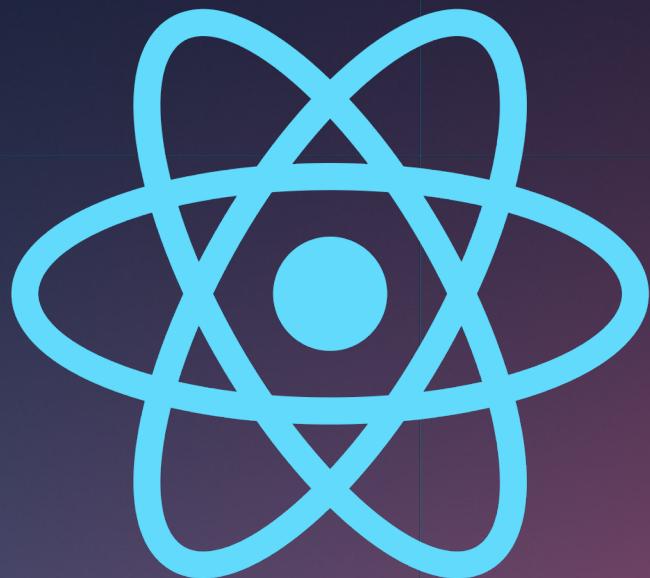
Delivered

A note about the Frontend tech-stack...

“making it painless to create interactive UIs”

“encapsulated components ... then compose them to make complex UIs”

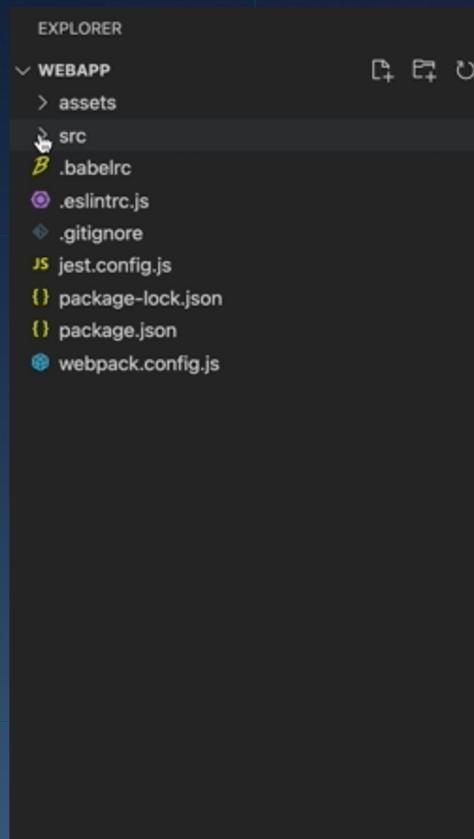
Design 6 is provided in React 😊



When you just want to tidy up a bit...



Feature driven development



Dashboard

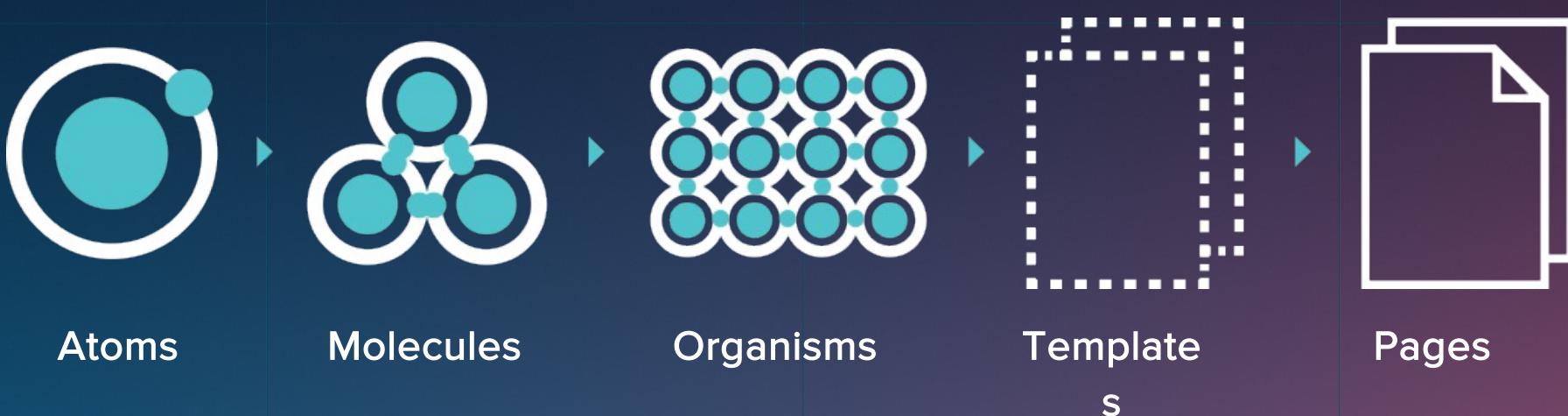
The screenshot shows a dashboard titled "Vehicle order tracker". At the top, there are six status indicators with counts: Processing (5), In production (10), Shipped (21), In country (34), At dealership (48), and Delivered (65). Below this is a table titled "All orders" listing 10 rows of order details. The columns include Customer, Vehicle, Status, Order progress, Arrival date, and Order details. The status column uses color-coded dots to indicate the status of each order. The arrival date column shows dates ranging from 2021-10-01 to 2021-10-10. Each row has a "View details" button. At the bottom of the table, there are pagination controls for items per page (10) and a "View All" link.

Customer	Vehicle	Status	Order progress	Arrival date	Order details
Stephen Hammersman	Tiguan	● On time	In production	2021-10-01	<button>View details</button>
Jochen Müller	Golf GTI	● Early	Shipped	2021-10-02	<button>View details</button>
Lisa-Marie Auditore	ID.4 Pure	● Delayed	At dealership	2021-10-03	<button>View details</button>
Simon Karmann	-	-	Processing	2021-10-04	<button>View details</button>
Anne Von Tonder	T-Roc	● On time	Delivered	2021-10-05	<button>View details</button>
Joseph Mundini	Polo GTI	● Delayed	In country	2021-10-06	<button>View details</button>
Stephen Philipp Hammersman A...	T-Cross Active	● On time	In production	2021-10-07	<button>View details</button>
Jochen Müller	ID.3 Pure	● Early	Shipped	2021-10-08	<button>View details</button>
Lisa-Marie Auditore	Golf GTI Clubsport	● Delayed	In country	2021-10-09	<button>View details</button>
Simon Karmann	Toureg	● On time	Processing	2021-10-10	<button>View details</button>



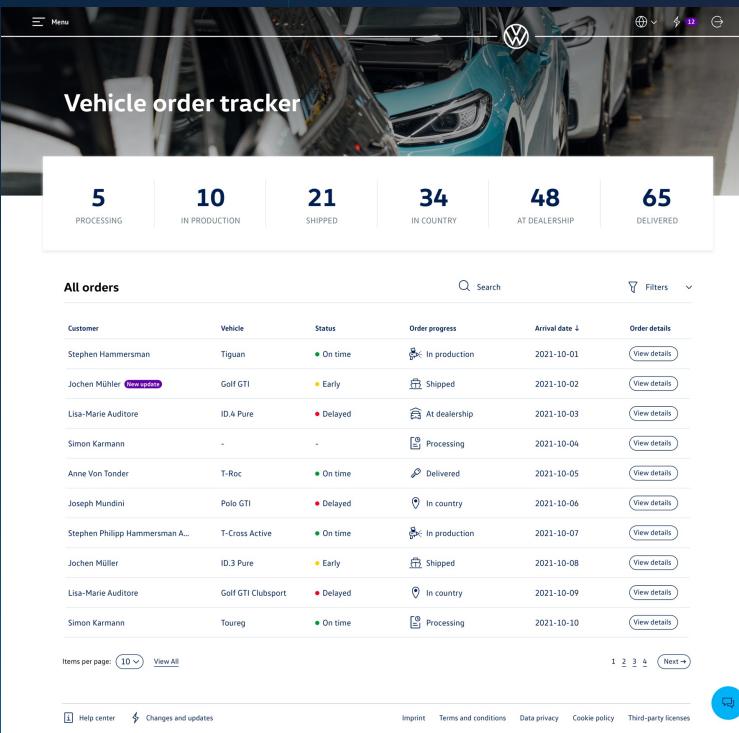
Atomic Design

It “details all that goes into creating and maintaining robust design systems, allowing you to roll out higher quality, more consistent UIs faster than ever before”.



[Brad Frost, <https://bradfrost.com/blog/post/atomic-web-design/>]

From Designs to Atomic components



The screenshot shows a "Vehicle order tracker" application interface. At the top, there's a banner with a blue car image and the text "Vehicle order tracker". Below the banner, a summary bar displays six counts: 5 PROCESSING, 10 IN PRODUCTION, 21 SHIPPED, 34 IN COUNTRY, 48 AT DEALERSHIP, and 65 DELIVERED. The main area is titled "All orders" and contains a table with columns for Customer, Vehicle, Status, Order progress, Arrival date, and Order details. The table lists ten orders with various statuses like "On time", "Early", "Delayed", and "In production". Each row includes a "View details" button. At the bottom, there are pagination controls for items per page (10) and links for Help center, Changes and updates, Imprint, Terms and conditions, Data privacy, Cookie policy, and Third-party licenses.

Customer	Vehicle	Status	Order progress	Arrival date	Order details
Stephen Hammersman	Tiguan	On time	In production	2021-10-01	View details
Jochen Müller	Golf GTI	Early	Shipped	2021-10-02	View details
Lisa-Marie Auditore	ID.4 Pure	Delayed	At dealership	2021-10-03	View details
Simon Karmann	-	-	Processing	2021-10-04	View details
Anne Von Tonder	T-Roc	On time	Delivered	2021-10-05	View details
Joseph Mundini	Polo GTI	Delayed	In country	2021-10-06	View details
Stephen Philipp Hammersman A...	T-Cross Active	On time	In production	2021-10-07	View details
Jochen Müller	ID.3 Pure	Early	Shipped	2021-10-08	View details
Lisa-Marie Auditore	Golf GTI Clubsport	Delayed	In country	2021-10-09	View details
Simon Karmann	Toureg	On time	Processing	2021-10-10	View details



- **Unbreakable components**
- **Base styles from the Design System**

[View details](#)

15/04/2021

ID.3 Style





```
1 import React from "react"
2 import { render, screen } from "@testing-library/react"
3 import ViewDetailsButton from "./view-details-button"
4 import { Link } from "react-router-dom"
5
6 jest.mock("react-router-dom", () => ({
7     ...jest.requireActual("react-router-dom"),
8     Link: jest.fn().mockImplementation(({ children }) => children)
9 }))
10
11 describe("ViewDetails button", () => {
12     it("shows the 'See Details' that will takes us somewhere", () => {
13
14         render(<ViewDetailsButton to="/take-me-to-wad" />)
15
16         expect(screen.getByText("View details")).toBeInTheDocument()
17
18         expect(Link).toHaveBeenCalledWith(expect.objectContaining(
19             { to: "/take-me-to-wad" }
20         ), {})
21     })
22 })
```

```
src > ui > atoms > view-details-button >  view-details-button.jsx > [e] default
1  const ViewDetailsButton = () => null
2
3  export default ViewDetailsButton
4
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
FAIL src/ui/atoms/view-details-button/view-details-button-mock-react-router-dom.test.jsx
ViewDetails button
  ✕ shows the 'See Details' that will takes us somewhere (26 ms)

● ViewDetails button > shows the 'See Details' that will takes us somewhere

  TestingLibraryElementError: Unable to find an element with the text: View details. This could be because the text is broken up by multiple
  elements. In this case, you can provide a function for your text matcher to make your matcher more flexible.

  <body>
    <div />
  </body>

  14 |       render(<ViewDetailsButton to="/take-me-to-wad" />)
  15 |
  > 16 |       expect(screen.getByText("View details")).toBeInTheDocument()
        ^
  17 |
  18 |       expect(Link).toHaveBeenCalledWith(expect.objectContaining(
  19 |         { to: "/take-me-to-wad" }

  at Object.getElementError (node_modules/@testing-library/dom/dist/config.js:37:19)
  at node_modules/@testing-library/dom/dist/query-helpers.js:90:38
  at node_modules/@testing-library/dom/dist/query-helpers.js:62:17
  at getByText (node_modules/@testing-library/dom/dist/query-helpers.js:111:19)
  at Object.<anonymous> (src/ui/atoms/view-details-button/view-details-button-mock-react-router-dom.test.jsx:16:23)

Test Suites: 1 failed, 1 total
Tests:       1 failed, 1 total
Snapshots:   0 total
Time:        5.137 s
Ran all test suites matching /src/ui/atoms/view-details-button/view-details-button-mock-react-router-dom.test.jsx/i.

Watch Usage: Press w to show more.
```

```
src > ui > atoms > view-details-button >  view-details-button.jsx > ...
1 import React from "react"
2
3 const ViewDetailsButton = () => <p>View details</p>
4
5 export default ViewDetailsButton
6
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

FAIL src/ui/atoms/view-details-button/**view-details-button-mock-react-router-dom.test.jsx**

ViewDetails button

 x shows the 'See Details' that will takes us somewhere (30 ms)

● **ViewDetails button > shows the 'See Details' that will takes us somewhere**

```
expect(jest.fn()).toHaveBeenCalled(...expected)
```

```
Expected: ObjectContaining {"to": "/take-me-to-wad"}, {}
```

Number of calls: 0

```
16 |       expect(screen.getByText("View details")).toBeInTheDocument() ✓
17 |
18 |     > expect(Link).toHaveBeenCalled(expect.objectContaining(
19 |       ^
20 |       { to: "/take-me-to-wad" }
21 |     ), {})
22 |   }
```

```
at Object.<anonymous> (src/ui/atoms/view-details-button/view-details-button-mock-react-router-dom.test.jsx:18:22)
```

Test Suites: 1 failed, 1 total

Tests: 1 failed, 1 total

Snapshots: 0 total

Time: 4.153 s

Ran all test suites matching /src/ui/atoms/view-details-button/view-details-button-mock-react-router-dom.test.jsx/i.

Watch Usage: Press w to show more. []

```
src > ui > atoms > view-details-button >  view-details-button.jsx >  ViewDetailsButton  
1 import React from "react"  
2 import { Link } from "react-router-dom"  
3 import "./view-details-button.scss"  
4  
5 const ViewDetailsButton = ({ to }) => [  
6  
7     return <div className="view-details-button">  
8         <Link to={to}>  
9             | View details  
10            </Link>  
11        </div>  
12    [  
13  
14    export default ViewDetailsButton  
15
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PASS src/ui/atoms/view-details-button/**view-details-button-mock-react-router-dom.test.jsx**
ViewDetails button
✓ shows the 'See Details' that will takes us somewhere (29 ms)

Test Suites: **1 passed**, 1 total
Tests: **1 passed**, 1 total
Snapshots: 0 total
Time: 3.347 s
Ran all test suites matching /src/ui/atoms/view-details-button/view-details-button-mock-react-router-dom/.test/.jsx/i.

Watch Usage: Press w to show more.

From Designs to Atomic components

The dashboard features a header with the title "Vehicle order tracker" and a background image of a blue Volkswagen car. Below the header, there are six large numbers representing different stages of the order process: 5 (PROCESSING), 10 (IN PRODUCTION), 21 (SHIPPED), 34 (IN COUNTRY), 48 (AT DEALERSHIP), and 65 (DELIVERED). A "All orders" table follows, displaying columns for Customer, Vehicle, Status, Order progress, Arrival date, and Order details. The table lists ten orders with various statuses like "On time", "Early", "Delayed", and "In production". At the bottom, there are pagination controls for items per page (10) and links for Help center, Changes and updates, Imprint, Terms and conditions, Data privacy, Cookie policy, and Third-party licenses.



- Atoms bonded together
- Bring functionality to the UI

Search

In transit

```
src > ui > molecules > progress > progress.test.jsx > ...
1 import React from "react"
2 import { render, screen } from "@testing-library/react"
3 import Progress from "./progress"
4 import { InTransitIcon } from "../../atoms/icons/progress-icons"
5
6 jest.mock("../../atoms/icons/progress-icons", () => {
7   return {
8     __esModule: true,
9     default: jest.fn().mockReturnValue(null)
10  }
11})
12
13 describe("Progress molecule", () => {
14   it("renders the visual elements when status is 'In transit'", () => {
15     render(<Progress />)
16
17     expect(screen.getByText("In transit")).toBeInTheDocument()
18
19     expect(InTransitIcon).toHaveBeenCalled()
20   })
21 })
22 }
```

PROBLEMS 21 OUTPUT DEBUG CONSOLE TERMINAL

FAIL src/ui/molecules/progress/**progress.test.jsx**

Progress molecule

✗ renders the visual elements when status is 'In transit' (26 ms)

● Progress molecule > renders the visual elements when status is 'In transit'

TestingLibraryElementError: Unable to find an element with the text: In transit. This could be because the text is broken up by multiple elements. In this case, you can provide a function for your text matcher to make your matcher more flexible.

```
<body>
  <div />
</body>

16 |       render(<Progress />)
17 |
18 |     expect(screen.getByText("In transit")).toBeInTheDocument()
19 |
20 |     expect(InTransitIcon).toHaveBeenCalled()
21 |   })
```

render(<Progress />)

expect(screen.getByText("In transit")).toBeInTheDocument()

expect(InTransitIcon).toHaveBeenCalled()

```
src > ui > molecules > progress > ⚡ progress.jsx > ...
```

```
1  import React from "react"
2  import InTransitIcon from " ../../atoms/icons/in-transit"
3  import Text from " ../../atoms/text/text"
4
5  const Progress = () => {
6
7      return (
8          <div>
9              <InTransitIcon />
10             <Text>In transit</Text>
11         </div>
12     )
13 }
14
15 export default Progress
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

1: node



PASS src/ui/molecules/progress/progress.test.jsx

Progress molecule

✓ renders the visual elements when status is 'In transit' (24 ms)

Test Suites: 1 passed, 1 total

Tests: 1 passed, 1 total

Snapshots: 0 total

Time: 1.402 s

Ran all test suites matching /src/ui/molecules/progress/progress.test.jsx/i.

Watch Usage: Press w to show more. []

From Designs to Atomic components

The screenshot shows a web-based vehicle order tracking system. At the top, there's a banner with the text "Vehicle order tracker" and a background image of a blue Volkswagen car. Below the banner, a summary bar displays six counts: 5 (PROCESSING), 10 (IN PRODUCTION), 21 (SHIPPED), 34 (IN COUNTRY), 48 (AT DEALERSHIP), and 65 (DELIVERED). The main area is titled "All orders" and contains a table with columns for Customer, Vehicle, Status, Order progress, Arrival date, and Order details. The table lists ten customer entries with their vehicle models and current status (e.g., On time, Early, Delayed). At the bottom, there are pagination controls for items per page (10) and links for Help center, Changes and updates, Imprint, Terms and conditions, Data privacy, Cookie policy, and Third-party licenses.



- A flock of atoms, molecules and even organisms!
- *User Value*

```

1 import React from "react"
2 import { render, screen } from "@testing-library/react"
3 import OrdersList from "./orders-list"
4 import FiltersButton from "../../molecules/filters-button/filters-button"
5 import SearchBar from "../../molecules/search-bar/search-bar"
6 import OrdersTable from "../../molecules/orders-table/orders-table"
7
8 > jest.mock("../../molecules/filters-button/filters-button", () => { ...
13   })
14
15 > jest.mock("../../molecules/search-bar/search-bar", () => { ...
20   })
21
22 > jest.mock("../../molecules/orders-table/orders-table", () => { ...
27   })
28
29 describe('OrdersList organism', () => {
30   it('should have a title, the search bar, the filters button and the table with the orders', () => {
31     render(<OrdersList />)
32
33     expect(screen.getByText("All orders")).toBeInTheDocument()
34
35     expect(SearchBar).toHaveBeenCalled()
36     expect(FiltersButton).toHaveBeenCalled()
37     expect(OrdersTable).toHaveBeenCalled()
38   })
39 })
40 }
41

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: node

OrdersList organism
 ✘ should have a title, the search bar, the filters button and the table with the orders (6 ms)

- OrdersList organism > should have a title, the search bar, the filters button and the table with the orders

TestingLibraryElementError: Unable to find an element with the text: All orders. This could be because the text is broken up by multiple elements. In this case, you can provide a function for your text matcher to make your matcher more flexible.

```

<body>
  <div />
</body>

      render(<OrdersList />)
33
> 34   expect(screen.getByText("All orders")).toBeInTheDocument()

35
36   expect(SearchBar).toHaveBeenCalled()
37   expect(FiltersButton).toHaveBeenCalled()

```

```

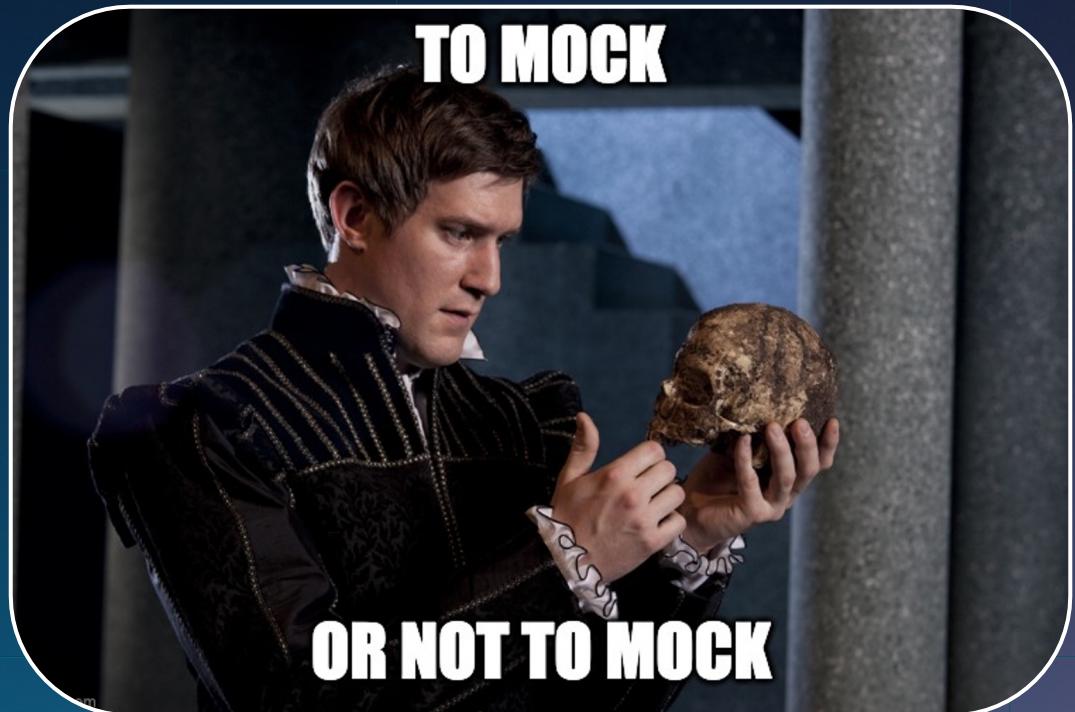
render(<OrdersList />)

expect(screen.getByText("All orders")).toBeInTheDocument()

expect(SearchBar).toHaveBeenCalled()
expect(FiltersButton).toHaveBeenCalled()
expect(OrdersTable).toHaveBeenCalled()

```

Let's get technical on Mocking

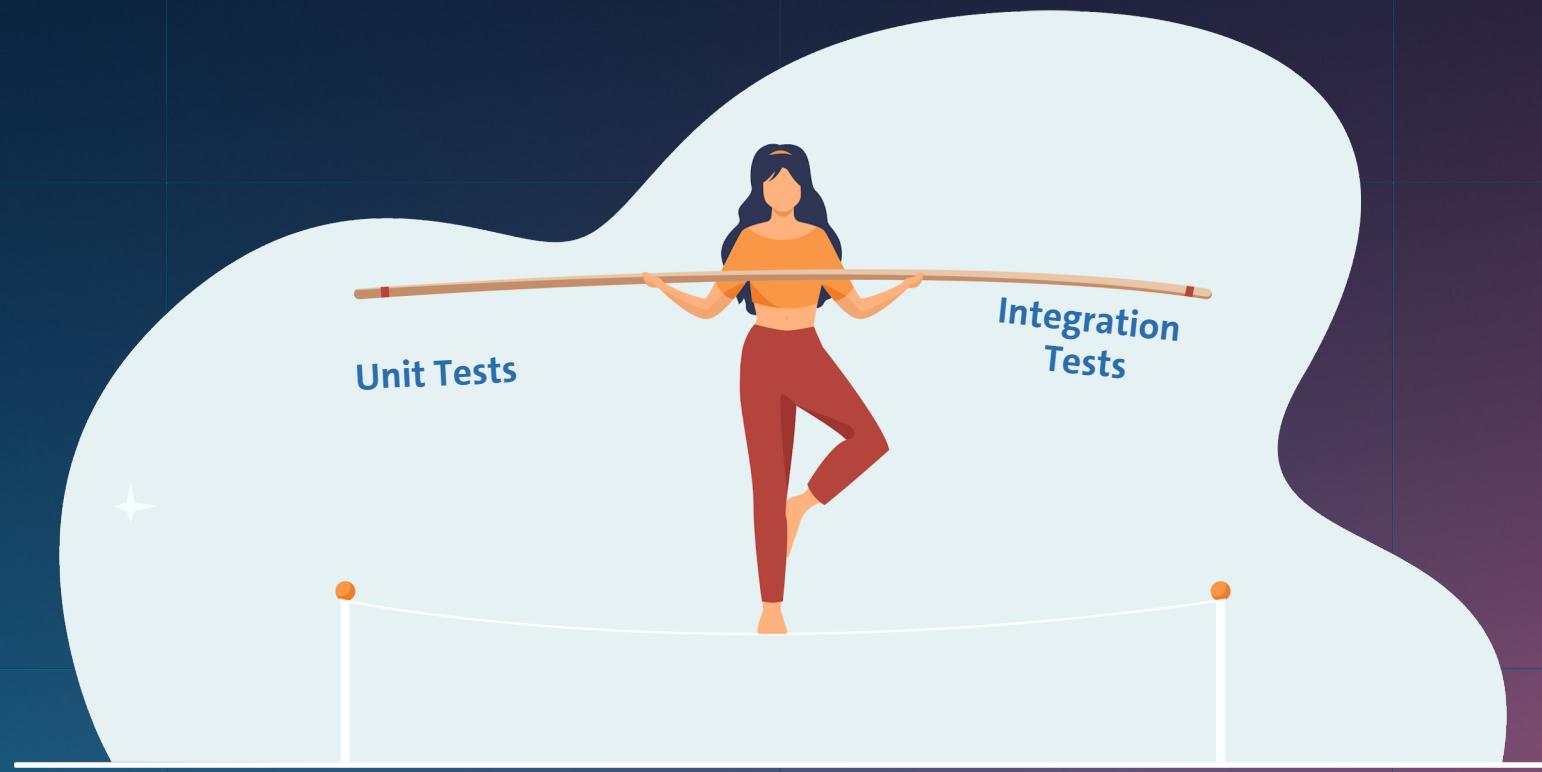


👍 Encapsulation
No need to know details

👎 Further away from User experience



Find your balance



The more your tests resemble
the way your software is used,
the more confidence they can
give you.

[Kent C. Dodds, <https://twitter.com/kentcdodds/status/977018512689455106?lang=en>]



```
import React from "react"
import { MemoryRouter, Route, Switch } from "react-router"
import { render, screen } from "@testing-library/react"
import userEvent from "@testing-library/user-event"
import OrdersList from "./orders-list"
import { getOrders } from "../../services/get-orders"

jest.mock("../../services/get-orders", () => ({
    getOrders: jest.fn()
}))

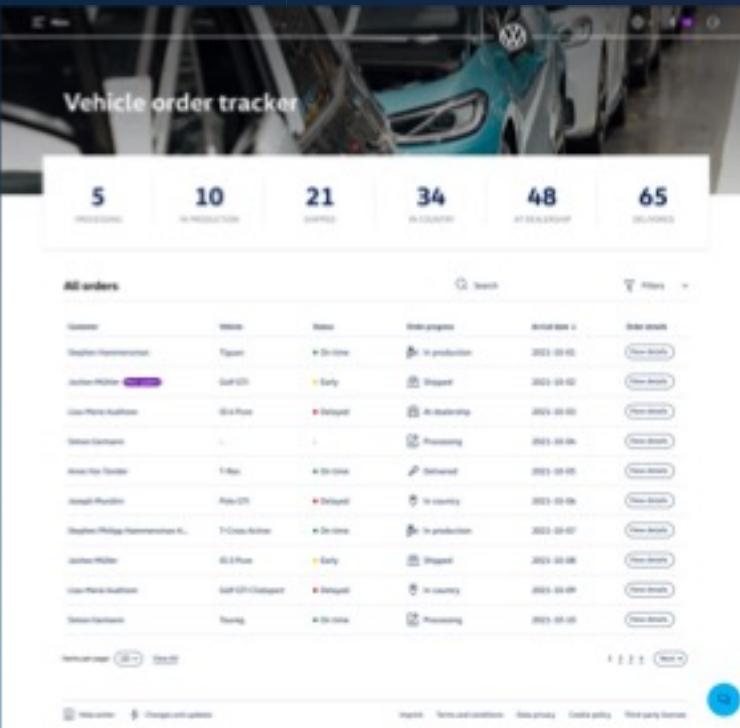
describe("OrdersList organism", () => {
    it("shows the order list and we can click to 'View Details'", () => {
        getOrders.mockReturnValue([
            {
                id: 1024,
                name: "Rita Castro",
                vehicle: "T-Roc",
                status: "delayed",
                progress: "order received",
                arrivalDate: "2021-07-24",
            }
        ])
        render(
            <MemoryRouter initialEntries={["/"]}>
                <Switch>
                    <Route exact path="/" render={() => <OrdersList />} />
                    <Route path="/orders/:orderId" render={() => <div>Stub Details Page</div>} />
                </Switch>
            </MemoryRouter>
        )
        expect(screen.getByRole("heading", { name: "All orders" })).toBeInTheDocument()

        expect(screen.getByText("Rita Castro")).toBeInTheDocument()
        // same goes for the other elements in the table

        const viewDetailsButton = screen.getByRole("link", { name: "View details" })
        userEvent.click(viewDetailsButton)

        expect(screen.getByText("Stub Details Page")).toBeInTheDocument()
    })
})
```

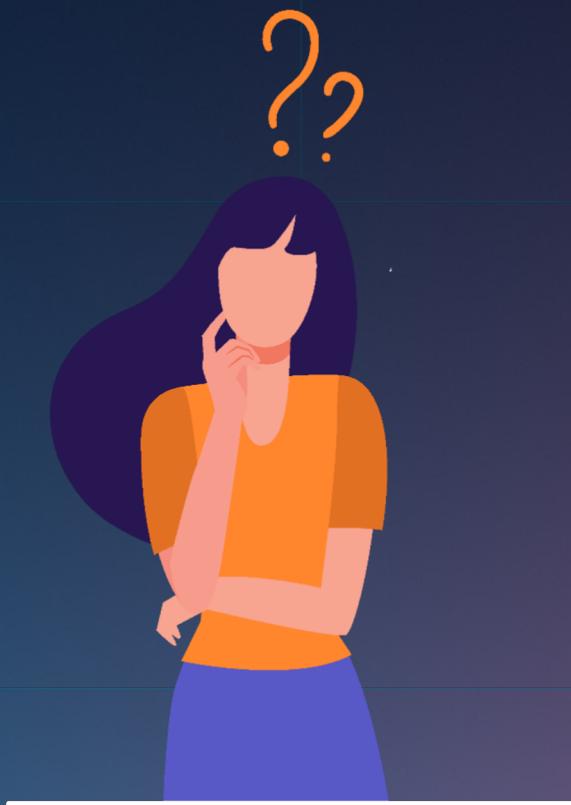
From Designs to Atomic components



- Define a layout
- Place organisms there



The Dealer's journey



```

import React from "react"
import { render, screen, waitFor, within } from "@testing-library/react"
import userEvent from "@testing-library/user-event"
import VehicleOrderTracker from "./vehicle-order-tracker"

describe("Vehicle order tracker for Remote React Summit 2021", () => {
  it("renders the Dashboard and we can 'View details' of an Order and come 'Back' to the Dashboard", () => {
    render(<VehicleOrderTracker />)

    const dashboard = screen.getByRole("region", { name: "Dashboard" })
    expect(dashboard).toBeInTheDocument()

    const overviewTabs = within(dashboard).getByRole("tablist")
    expect(overviewTabs).toBeInTheDocument()

    expect(within(dashboard).getByText("All orders")).toBeInTheDocument()
    expect(within(dashboard).getByLabelText("Search for an order")).toBeInTheDocument()
    expect(within(dashboard).getByLabelText("Filter orders")).toBeInTheDocument()

    const scullysOrder = within(dashboard).getByLabelText("Dana Scully's order")
    expect(scullysOrder).toBeInTheDocument()

    const viewDetailsButton = within(scullysOrder).getByRole("link", { name: "View details" })
    userEvent.click(viewDetailsButton)

    expect(screen.queryByRole("region", { name: "Dashboard" })).not.toBeInTheDocument()
    expect(screen.getByRole("region", { name: "Order details" })).toBeInTheDocument()

    const backButton = screen.getByRole("button", { name: "Back" })
    expect(backButton).toBeInTheDocument()

    userEvent.click(backButton)

    await waitFor(() => {
      expect(screen.getByText("All orders")).toBeInTheDocument()
    })
  })
}
)

```



```

import React from "react"
import OrdersList from "../../organisms/orders-list/orders-list"
import OverviewTabs from "../../organisms/overview-tabs/overview-tabs"

const Dashboard = () => {
  return (
    <div role="region" aria-label="Dashboard">
      <OverviewTabs />
      <OrdersList />
    </div>
  )
}

export default Dashboard

```

```

import React from "react"
import { render, screen, waitFor, within } from "@testing-library/react"
import userEvent from "@testing-library/user-event"
import VehicleOrderTracker from "./vehicle-order-tracker"

describe("Vehicle order tracker for Remote React Summit 2021", () => {
  it("renders the Dashboard and we can 'View details' of an Order and come 'Back' to the Dashboard", async () => {
    render(<VehicleOrderTracker />)

    const dashboard = screen.getByRole("region", { name: "Dashboard" })
    expect(dashboard).toBeInTheDocument()

    const overviewTabs = within(dashboard).getByRole("tablist")
    expect(overviewTabs).toBeInTheDocument()

    expect(within(dashboard).getByText("All orders")).toBeInTheDocument()
    expect(within(dashboard).getByLabelText("Search for an order")).toBeInTheDocument()
    expect(within(dashboard).getByLabelText("Filter orders")).toBeInTheDocument()

    const scullysOrder = within(dashboard).getByLabelText("Dana Scully's order")
    expect(scullysOrder).toBeInTheDocument()

    const viewDetailsButton = within(scullysOrder).getByRole("link", { name: "View details" })
    userEvent.click(viewDetailsButton)

    expect(screen.queryByRole("region", { name: "Dashboard" })).not.toBeInTheDocument()
    expect(screen.getRole("region", { name: "Order details" })).toBeInTheDocument()

    const backButton = screen.getByRole("button", { name: "Back" })
    expect(backButton).toBeInTheDocument()

    userEvent.click(backButton)

    await waitFor(() => {
      expect(screen.getByText("All orders")).toBeInTheDocument()
    })
  })
}
)

```

```

import React, { useEffect, useState } from "react"
import { getOrders } from "../../services/get-orders"
import FormattedDateNumbers from "../../atoms/formatted-date/numbers/formatted-date-numbers"
import ViewDetailsButton from "../../atoms/view-details-button/view-details-button"

const OrdersTable = () => {
  const [orders, setOrders] = useState([])

  useEffect(() => setOrders(getOrders), [])

  return (
    <table aria-label="Orders list">
      <thead>
        <tr>
          <th>Customer</th>
          <th>Vehicle</th>
          <th>Status</th>
          <th>Order progress</th>
          <th>Arrival date</th>
          <th>Order details</th>
        </tr>
      </thead>
      <tbody>
        {
          orders.map(order =>
            <tr key={order.id} aria-label={`${order.name}'s order`}>
              <td>{order.name}</td>
              <td>{order.vehicle}</td>
              <td>{order.status}</td>
              <td>{order.progress}</td>
              <td><FormattedDateNumbers value={order.arrivalDate} /></td>
              <td><ViewDetailsButton to={`/orders/${order.id}`}></ViewDetailsButton></td>
            </tr>
          )
        }
      </tbody>
    </table>
  )
}

```

```

import React from "react"
import { render, screen, waitFor, within } from "@testing-library/react"
import userEvent from "@testing-library/user-event"
import VehicleOrderTracker from "./vehicle-order-tracker"

describe("Vehicle order tracker for Remote React Summit 2021", () => {
  it("renders the Dashboard and we can 'View details' of an Order and come 'Back' to the Dashboard", async () => {
    render(<VehicleOrderTracker />)

    const dashboard = screen.getByRole("region", { name: "Dashboard" })
    expect(dashboard).toBeInTheDocument()

    const overviewTabs = within(dashboard).getByRole("tablist")
    expect(overviewTabs).toBeInTheDocument()

    expect(within(dashboard).getByText("All orders")).toBeInTheDocument()
    expect(within(dashboard).getByLabelText("Search for an order")).toBeInTheDocument()
    expect(within(dashboard).getByLabelText("Filter orders")).toBeInTheDocument()

    const scullysOrder = within(dashboard).getByLabelText("Dana Scully's order")
    expect(scullysOrder).toBeInTheDocument()

    const viewDetailsButton = within(scullysOrder).getByRole("link", { name: "View details" })
    userEvent.click(viewDetailsButton)

    expect(screen.queryByRole("region", { name: "Dashboard" })).not.toBeInTheDocument()
    expect(screen.getByRole("region", { name: "Order details" })).toBeInTheDocument()

    const backButton = screen.getByRole("button", { name: "Back" })
    expect(backButton).toBeInTheDocument()

    userEvent.click(backButton)

    await waitFor(() => {
      expect(screen.getByText("All orders")).toBeInTheDocument()
    })
  })
}
)

```

```

import React from "react"
import { BrowserRouter, Link, Redirect, Route, Switch } from "react-router-dom"
import Dashboard from "./ui/pages/dashboard/dashboard"
import OrderDetails from "./ui/pages/order-details/order-details"
import "./vehicle-order-tracker.scss"

const VehicleOrderTracker = () => {
  return <BrowserRouter>
    <header>
      <Link to="/">
        
        Vehicle Order Tracker Mockup
      </Link>
    </header>

    <main>
      <Switch>
        <Route path="/dashboard" render={() => <Dashboard />} />
        <Route path="/orders/:orderId" render={() => <OrderDetails />} />
        <Redirect exact from="/" to="/dashboard" />
      </Switch>
    </main>

    <footer>
      <p>Rita Castro | 2022</p>
    </footer>
  </BrowserRouter>
}

export default VehicleOrderTracker

```

```

import React from "react"
import { render, screen, waitFor, within } from "@testing-library/react"
import userEvent from "@testing-library/user-event"
import VehicleOrderTracker from "./vehicle-order-tracker"

describe("Vehicle order tracker for Remote React Summit 2021", () => {
  it("renders the Dashboard and we can 'View details' of an Order and come 'Back' to the Dashboard", async () => {
    render(<VehicleOrderTracker />)

    const dashboard = screen.getByRole("region", { name: "Dashboard" })
    expect(dashboard).toBeInTheDocument()

    const overviewTabs = within(dashboard).getByRole("tablist")
    expect(overviewTabs).toBeInTheDocument()

    expect(within(dashboard).getByText("All orders")).toBeInTheDocument()
    expect(within(dashboard).getByLabelText("Search for an order")).toBeInTheDocument()
    expect(within(dashboard).getByLabelText("Filter orders")).toBeInTheDocument()

    const scullysOrder = within(dashboard).getByLabelText("Dana Scully's order")
    expect(scullysOrder).toBeInTheDocument()

    const viewDetailsButton = within(scullysOrder).getByRole("link", { name: "View details" })
    userEvent.click(viewDetailsButton)

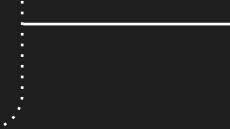
    expect(screen.queryByRole("region", { name: "Dashboard" })).not.toBeInTheDocument()
    expect(screen.getByRole("region", { name: "Order details" })).toBeInTheDocument()

    const backButton = screen.getByRole("button", { name: "Back" })
    expect(backButton).toBeInTheDocument()

    userEvent.click(backButton)
    ...

    await waitFor(() => {
      expect(screen.getByText("All orders")).toBeInTheDocument()
    })
  })
}
)

```



```

import React, { useEffect, useState } from "react"
import { useParams } from "react-router"
import { getOrderDetails } from "../../services/get-orders"
import BackButton from "../../atoms/back-button/back-button"

const OrderDetails = () => {
  const params = useParams()

  const [order, setOrder] = useState()

  useEffect(() => {
    setOrder(getOrderDetails(params.orderId))
  }, [])

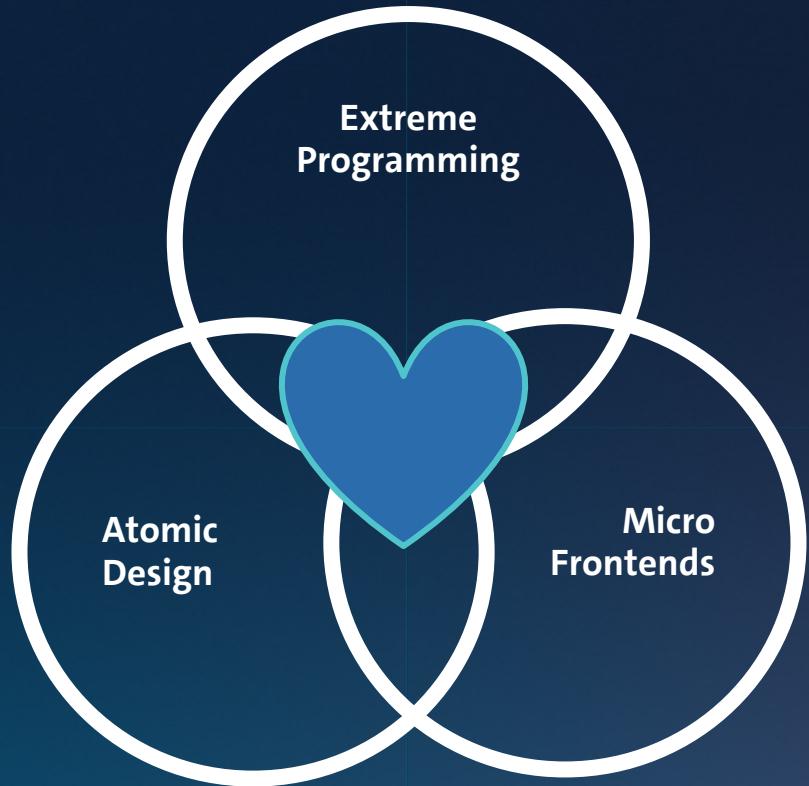
  return (
    <div role="region" aria-label="Order details">
      <BackButton />
      <OrderSummary order={order} />
      <OrderStatusTracker order={order} />
    </div>
  )
}

export default OrderDetails

```

The Dealer's journey





To independently deliver high quality software that gets composed into a greater whole featuring consistent and solid UIs, at a faster pace than ever before.

*Any day of the week.
Including Fridays.*

Thank you!

 rita.castro@vwds.pt

 <https://github.com/ritamcastro>

 <https://www.linkedin.com/in/ritamcastro>

 ritamcastro@gmail.com



© Rita Castro 2022