BACKBASE

Widget Architecture 3

Training

Back-end

- Service SDK
- Direct Integration
- Content Services
- Targeting
- Security

Front-end

- WA3 Overview
- Backbase in Angular World
- Backbase Design System

Foundation

- Introduction to Backbase
- CXS Basics

Objectives

What are we going to cover

- To get familiar with the new Widget Architecture 3
- Differences between Angular and Backbase specific frontend development
- To get familiar with Backbase Design System

BACKBASE

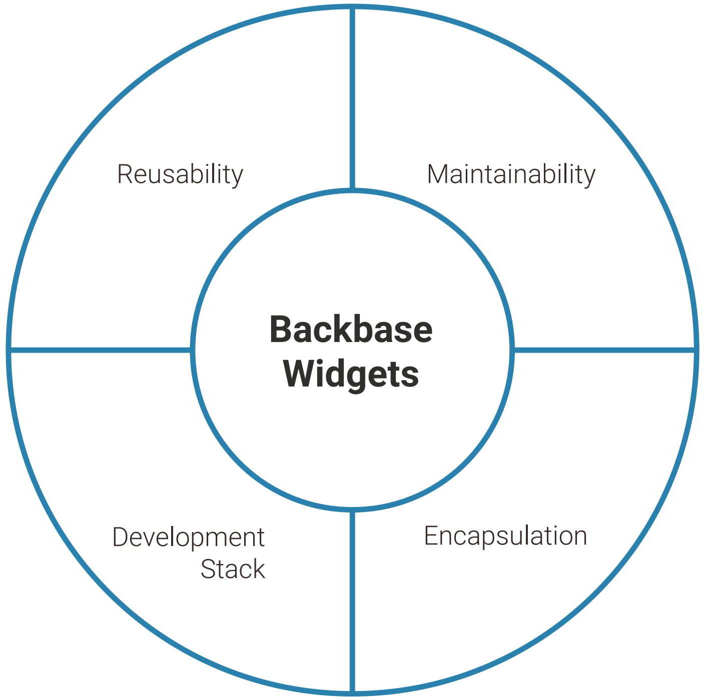
Widget Architecture 3 **Overview**

Why **Backbase Widgets?**

The Backbase Difference

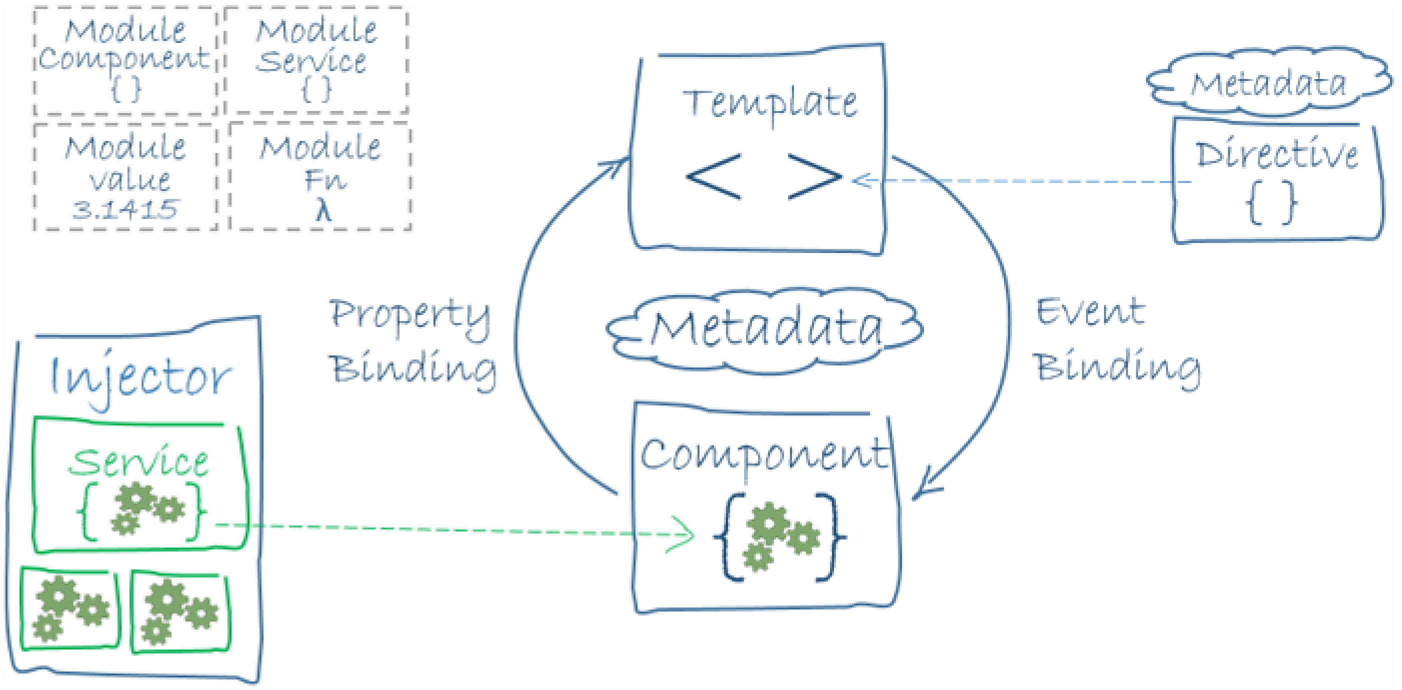
Widgets are **manageable components** that implement **reusable** pieces of **business logic**.

Business users can compose a **web application** combining them to build a Single Page Application (**SPA**).



Development Stack

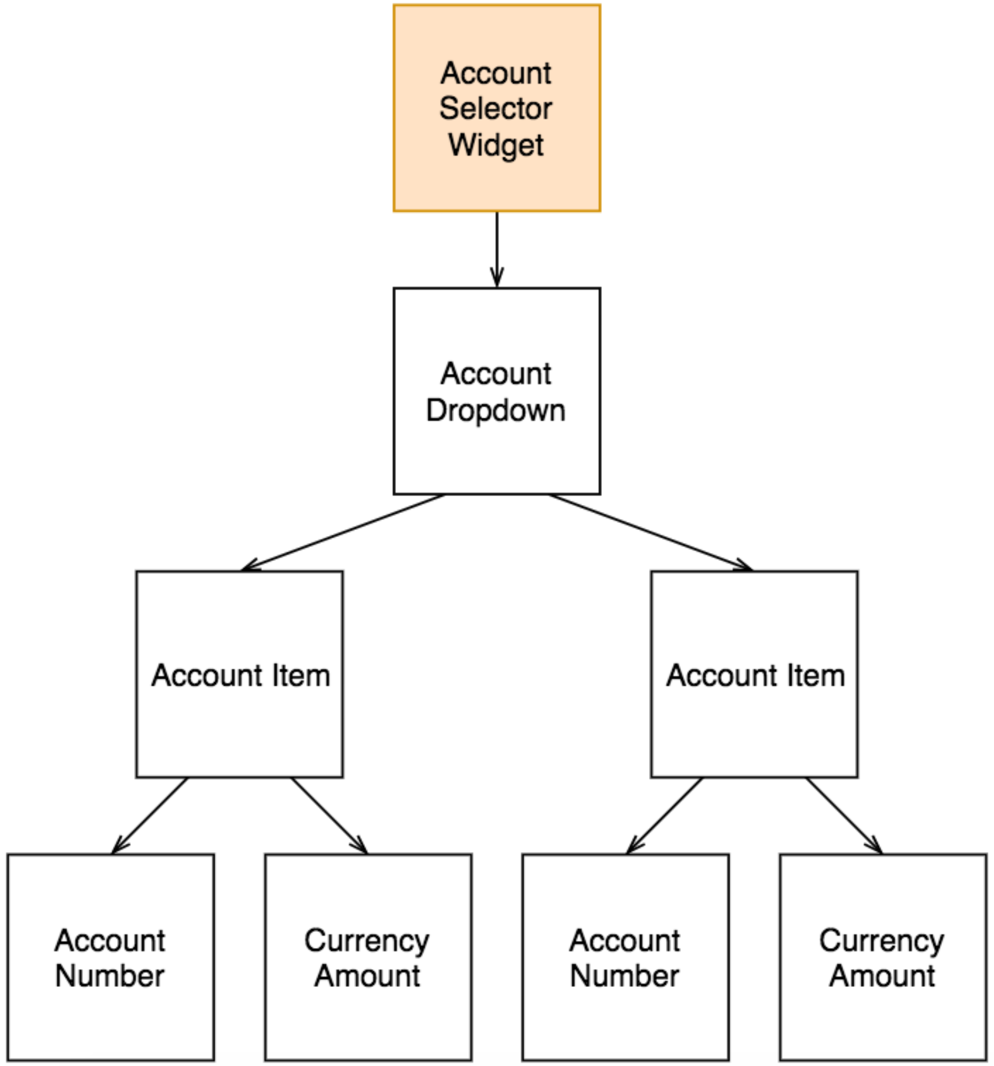
Widget Architecture 3 - Widget Architectural Overview



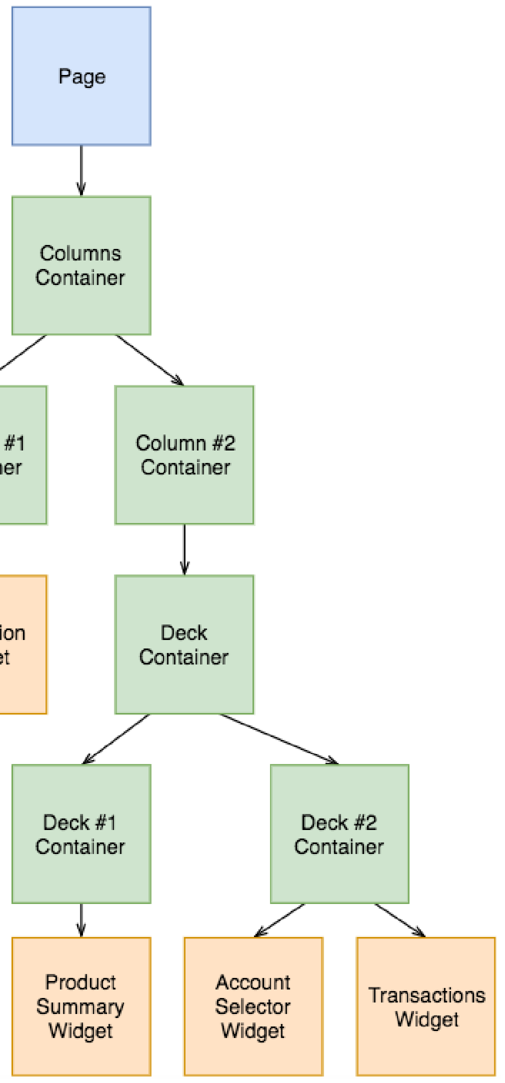
BACKBASE

Widgets and Containers

Widget Architecture 3 - A typical frontend application

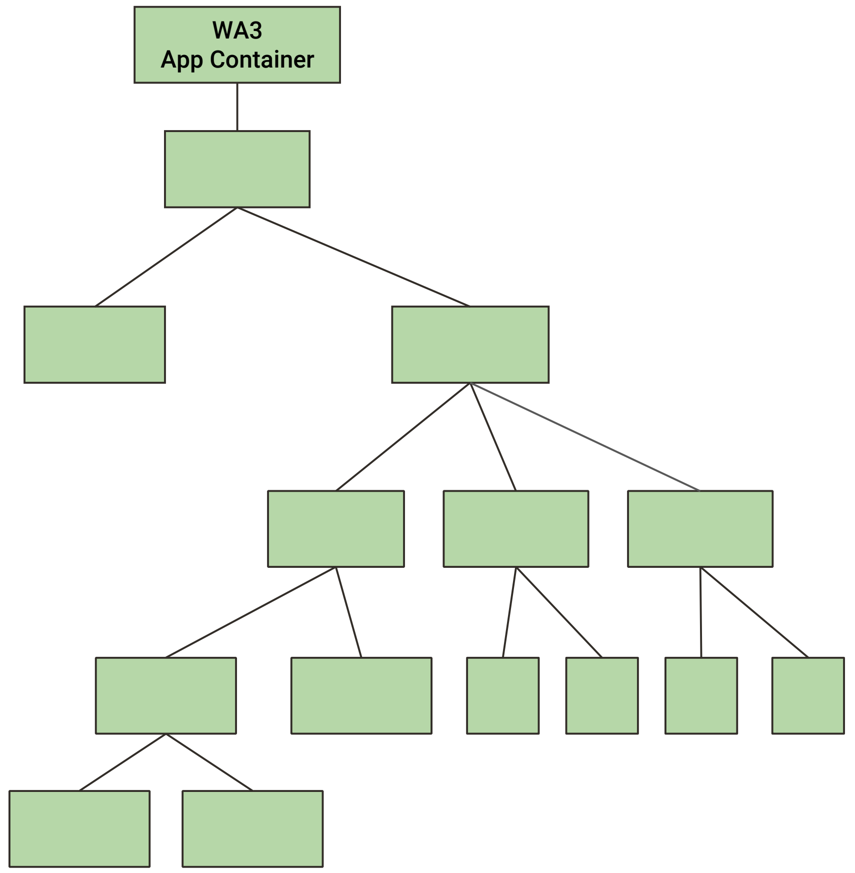


Widget Architecture 3 - Widgets and Containers



Widget Architecture 3 - Single Page Application

- Improved UX
- Faster Rendering
- Fewer Requests
- Less complex state management
- Deep linking into widgets and states
- Better security



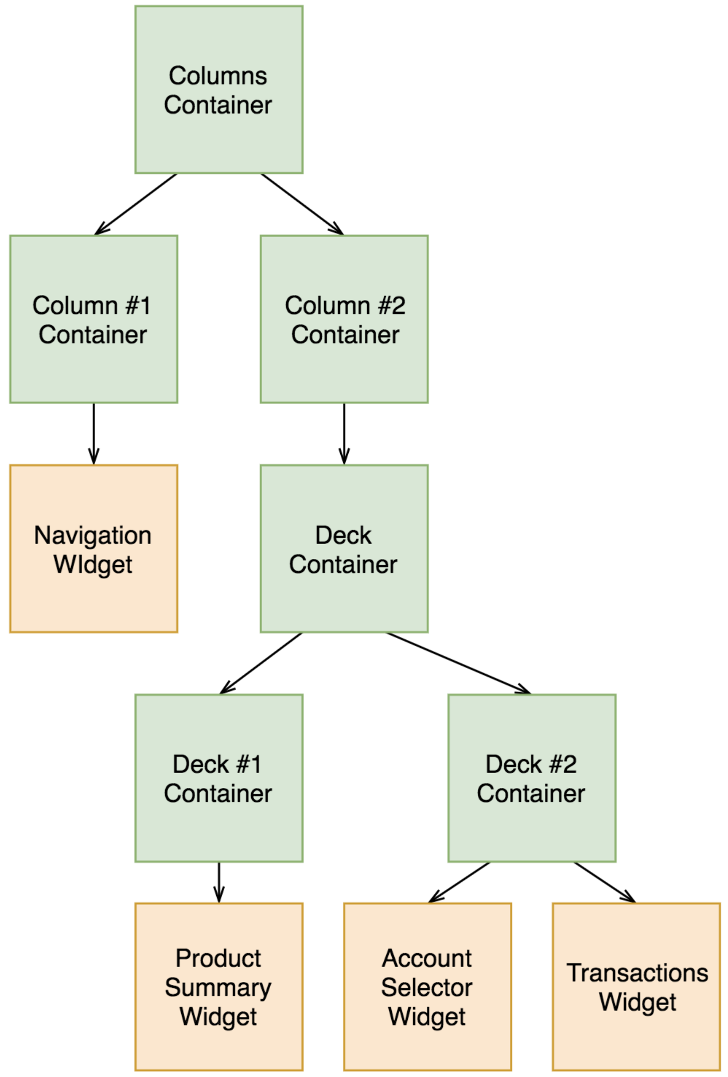
BACKBASE

Simplified model

```
{
 name: 'page_1238612638',
 properties: {},
 children: [{
   name: 'bb-columns-container-ang_-_...',
   properties: {
     classId: 'ColumnsContainerComponent',
     columnClassNames: 'aa,bb',
     numberOfColumns: 2,
   },
   children: [{
     name: 'widget-_-234928374234',
     properties: {
       classId: 'ColorWidgetComponent',
       color: 'lawngreen',
     }
   }, {
     name: 'bb-color-widget-ang_-_...',
     properties: {
       classId: 'XtraColorWidgetComponent',
       color: 'lightblue',
       size: 234,
     }
   }],
 }],
}
```

Dynamic Rendering

Backbase in Angular World - Dynamic Rendering



Backbase in Angular World - Dynamic Template

```
<columns-container>
 <column>
   <navigation-widget properties="{ 'color': 'red' }"></navigation-widget>
 </column>
 <column>
   <deck-container>
     <deck>
       <product-summary-widget properties="{}"></product-summary-widget>
     </deck>
     <deck>
       <account-selector-widget properties="{}"></account-selector-widget>
       <transactions-widget properties="{}"></transactions-widget>
     </deck>
   <deck-container>
 </column>
</columns-container>
```

Backbase in Angular World - bb-root component

<bb-root> component will render app template based on application model stored on the backend.

```
@Component({
 selector: 'bb-todo-app',
 template: `
   <bb-root></bb-root>
  `,
})
export class AppComponent {}
```

Backbase in Angular World - bb-chrome & bb-area

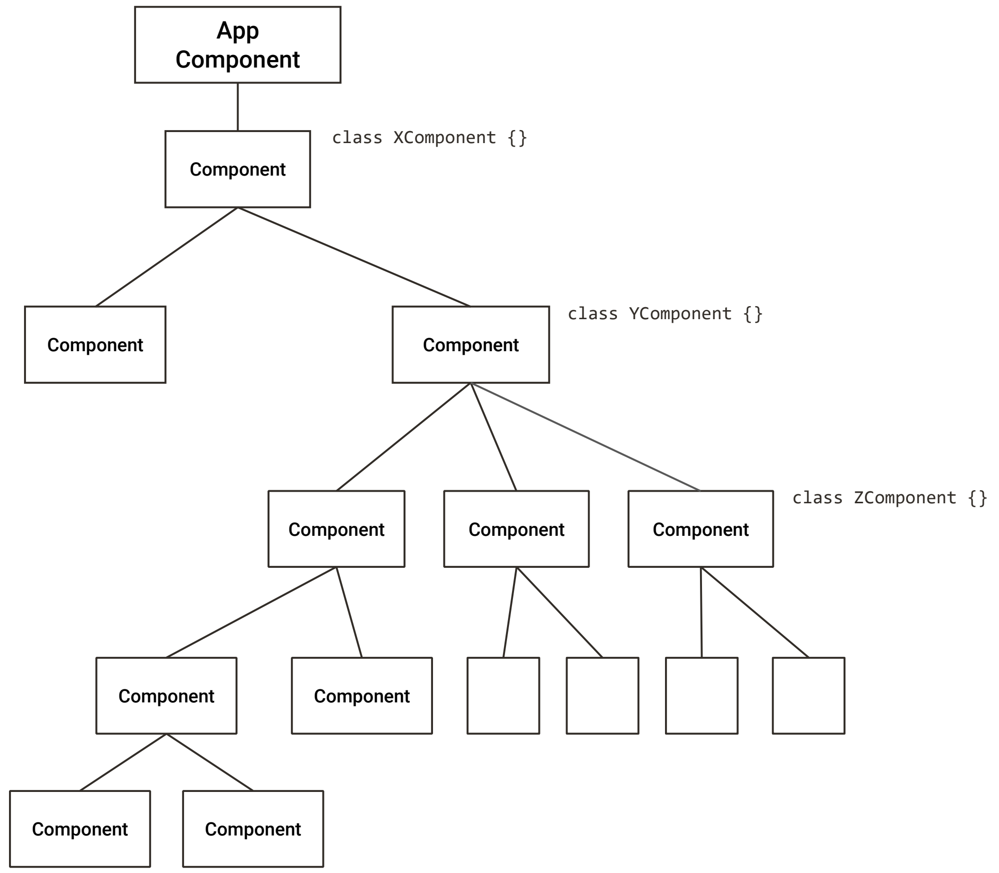
- bb-chrome purpose:
    - has data-id attribute which makes item selectable in Manager
    - dynamically renders widget or container
    - injects item-specific services via Angular hierarchical Dependency Injection

- bb-area purpose:
    - has data-area attribute which makes item droppable target in Manager

Backbase in Angular World - bb-chrome & bb-area

```
<bb-chrome>
  <columns-container>
    <bb-chrome>
      <column>
        <bb-area data-area=”...”>
          <bb-chrome data-id=”...”>
            <navigation-widget properties="{ 'color': 'red' }"></navigation-widget>
          </bb-chrome>
        </bb-area>
      </column>
    </bb-chrome>
  </columns-container>
</bb-chrome>
```

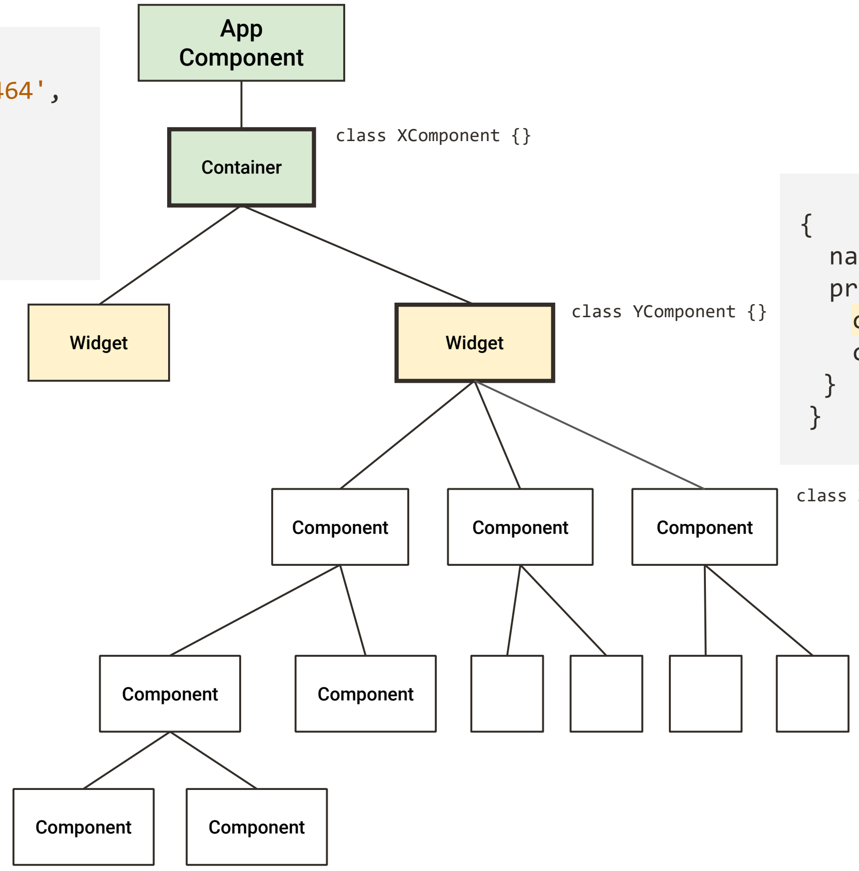
Angular **Component Tree**



application

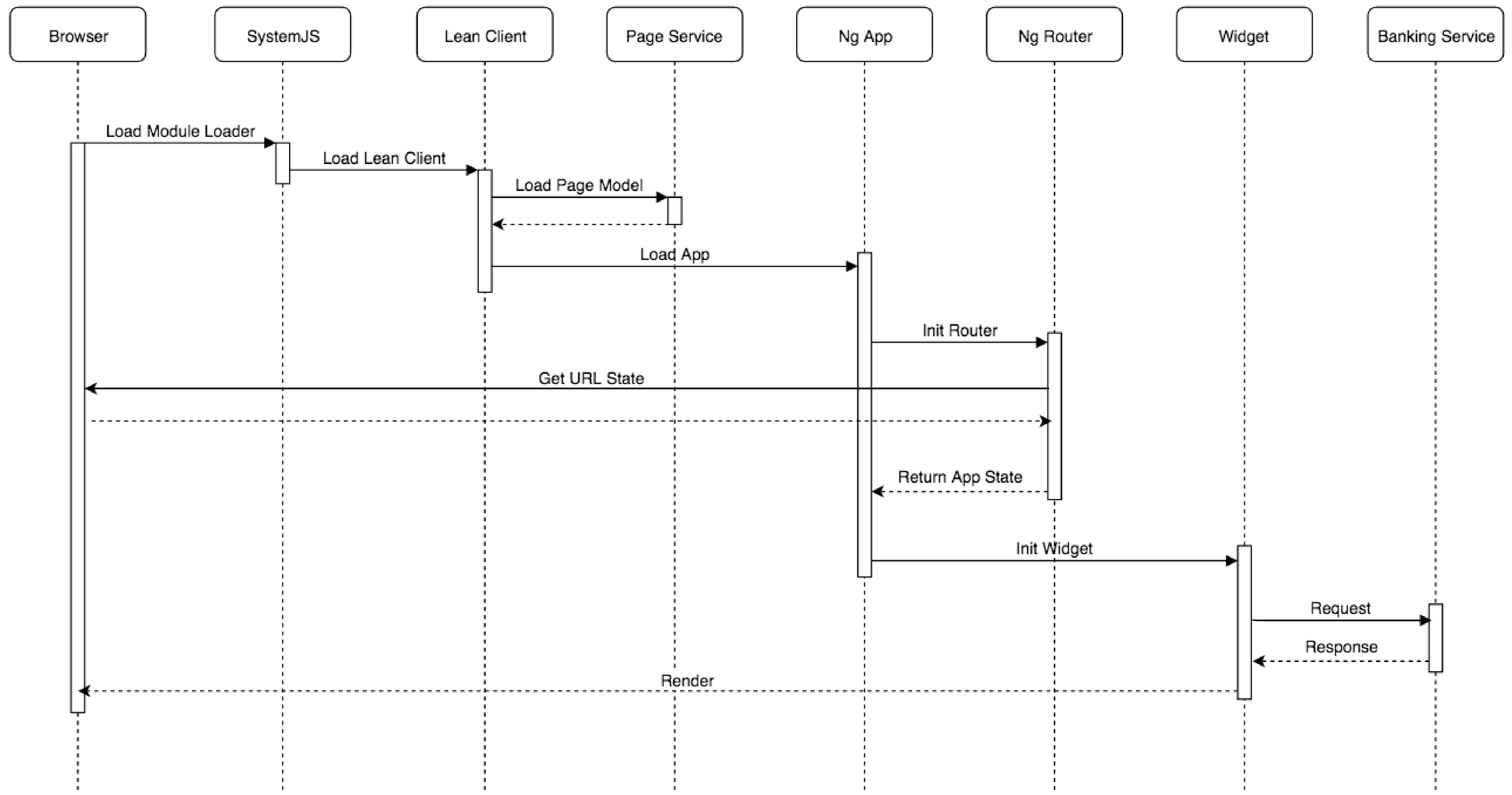
A **manageable components** tree

```
{
  name: 'container-_-7237243464',
  properties: {
    classId: 'XComponent',
    prop: 'qwerty',
  }
}
```

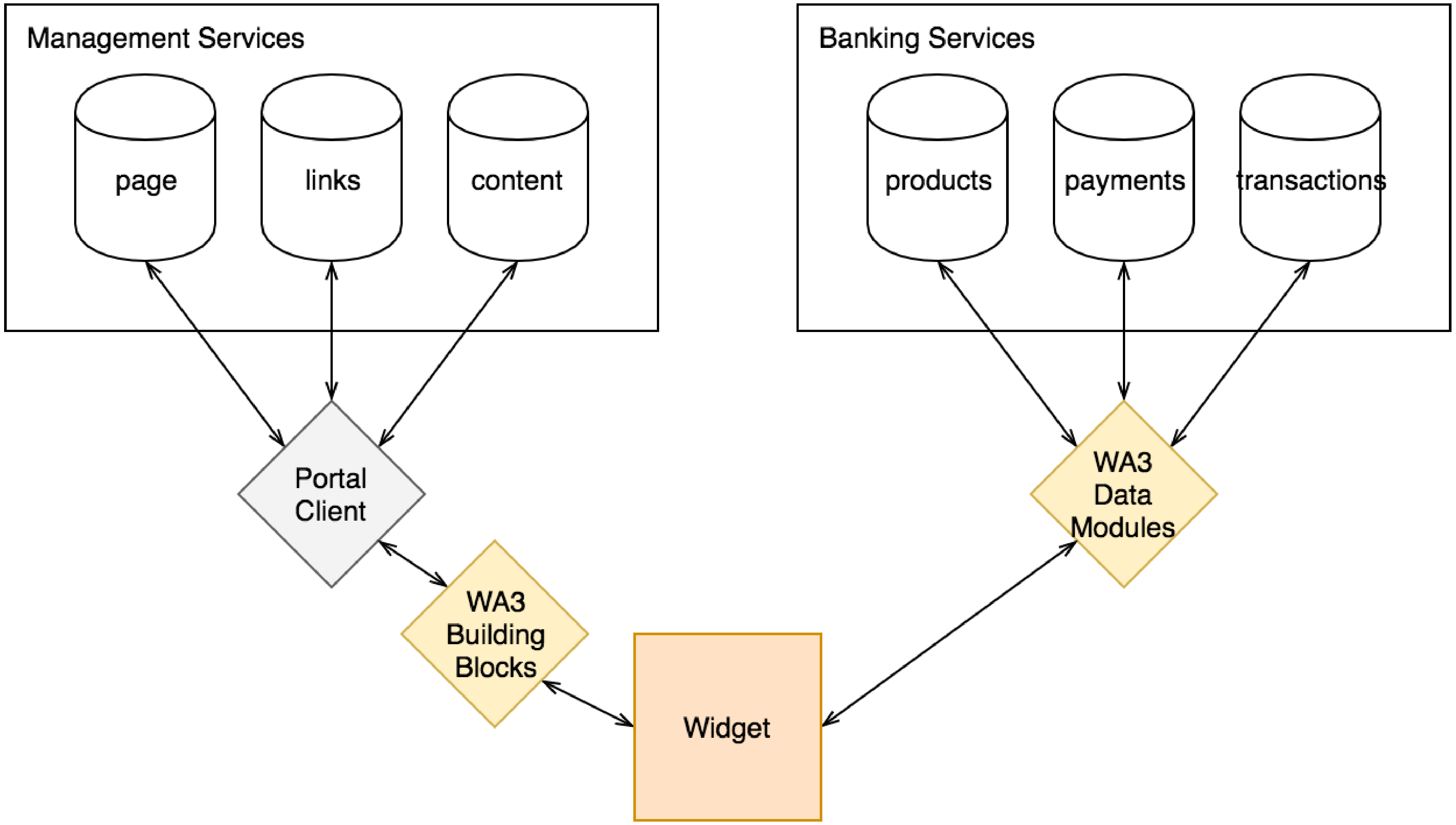


```
{
  name: 'widget-_-234928374234',
  properties: {
    classId: 'YComponent',
    color: 'lawngreen',
  }
}
```

WA3 Angular SPA - Sequence diagram



Widget Architecture 3 - Architecture Overview

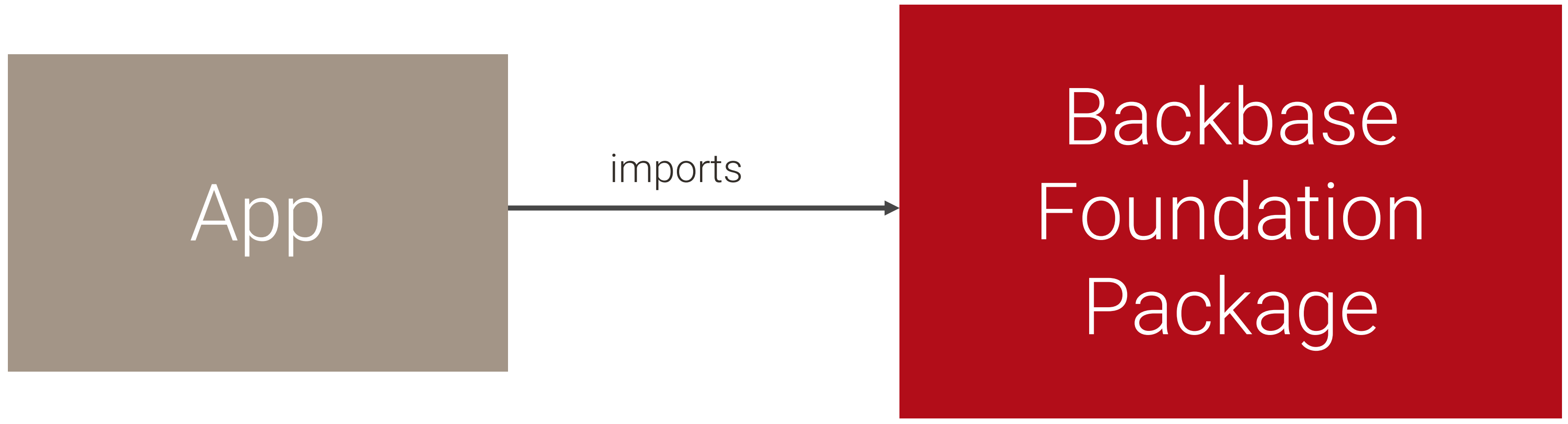


BACKBASE

WA3 Foundation

WA3 - **Backbase Foundation Package**

"The Backbase Foundation Package is the ***infrastructure*** *provider that enables the implementation of the* ***architecture****."*



WA3 - **Backbase Foundation Package**

npm install **@backbase/foundation-ang**

- *\/core* Core WA3 package
- *\/ui* Backbase UI Components Library (Angular)
- *\/data-http* WA3 data-module core
- *\/containers* panel, column, deck and tab container

WA3 Foundation - Take away

- Dynamically render Angular application based on model retrieved from the backend

- Provide Backbase specific injectable Angular services for widgets and containers
    - ItemModel
    - PageConfigService (staticResourcesRoot, apiRoot, locale...)
    - PortalContentService
    - EventBusService
    - NavigationService
    - forms

Widget Architecture 3 - Take Away

- Component
- Widget instance running in the Angular app
- Backbase Services available in the Angular app

BACKBASE

Workshops

- How to set up a **Widget Development Environment**
- Building the **Peachtree Bank Portal**

BACKBASE

Backbase in **Angular World**

Back-end

- Service SDK
- Direct Integration
- Content Services
- Targeting
- Security

Front-end

- WA3 Overview
- Backbase in Angular World
- Backbase Design System

Foundation

- Introduction to Backbase
- CXS Basics

BACKBASE

Benefits of Backbase WA3

Values that Backbase Frontend Architecture **adds** to Angular?

- Dynamic creation of Angular SPA based on the Backbase model
- Manageability of Angular SPAs (possibility for non developers to manage banking apps in Experience Manager)
- Banking UI Components and services (eg Currency Input, IBAN formatter pipe, ...)
- Extendibility Services - provides possibility to create widgets where templates / services can be customized easily

Values that Backbase Frontend Architecture **adds** to Angular?

- Enterprise Angular based software development life cycle for multiple (distributed) independent teams (project / app / widget schematics and Angular Package Format CLI)

- Data module auto generator - tool that auto generates Angular http data module based on RAML spec

- Backbase Content Integration (Backbase content services Angular services / widgets)

- Backbase Design System Integration

BACKBASE

Widget Properties

Backbase in Angular World - Widget Properties

Backbase widgets have properties that are manageable. To get values of those properties in widget component, ItemModel service is available

```
import { ItemModel } from '@backbase/foundation-ang/core';


@Component({
selector: 'my-widget',
template: `
  {{color | async}}
`,
})
export class AppComponent {
  constructor(private model: ItemModel) {}
  color: Observable<PropertyValue | undefined> = this.model.property('color');
}
```

Backbase Routing

Backbase in Angular World - Issues with Angular Routing

- Because Backbase apps are dynamically rendered, routing configuration for the app module can't be known at the time of development.

- Hardcoded outlet names into the widget code may cause issues in dynamically rendered app

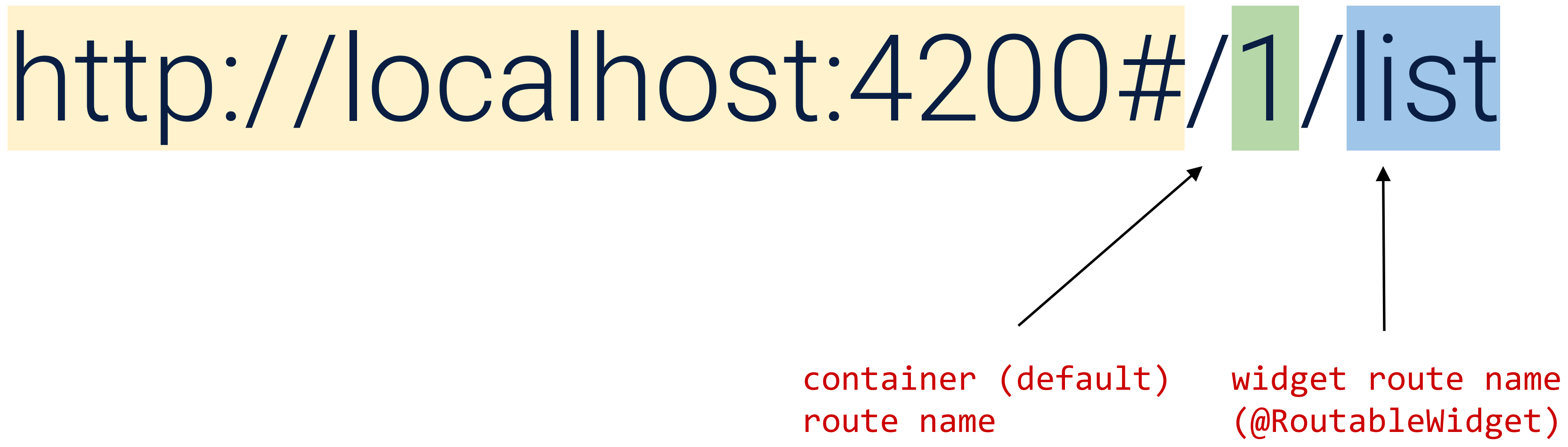
Backbase in Angular World - Why Backbase Routing?

- to dynamically generate a router configuration
- to dynamically generate a named router outlet
- a simple way to navigate with these outlets.

Backbase in Angular World - Backbase Routing Configuration

- App routing will be defined by containers
- By default, every container will dynamically get a route name (eg: “1”)
- To rename panel routes a new **route** property should be added.

Backbase in Angular World - Backbase Routing Configuration


http://localhost:4200#/1/list
container (default) route name
widget route name (@RoutableWidget)

Backbase in Angular World - Routable Widget

```
...
import { RoutableWidget } from '@backbase/foundation-ang/core';
...
@Component(...)
@RoutableWidget({
 routes: [
   { path: '', redirectTo: 'list', pathMatch: 'prefix' },
   { path: 'list', component: TodoListContainerComponent },
   { path: ':id', component: TodoDetailContainerComponent },
   { path: 'edit/:id', component: TodoFormContainerComponent },
 ],
})
export class TodoWidgetComponent {
 ...
}
```

Backbase in Angular World - Router outlet

```
@Component({
   selector: 'bb-todo-widget',
   template: `
       <bb-router-outlet></bb-router-outlet>
   `,
   ...
})
@RoutableWidget(...)
export class TodoWidgetComponent {
...
}
```

This creates a dynamically named router outlet. By default it is the primary router outlet, but if there is a property set on the widget called "outletName", the value of that property will be used instead.

Backbase in Angular World - Router Links

As with the Angular Router, there are **two different ways** to link within the widget:

1. **with a link directive**
2. **with a service**

In Angular this is the "**routerLink**" directive, and the "**Router.navigate**" method.

In Backbase this is the "**bbRouterLink**" directive, and the "**RouterService.navigate**" method

Both Backbase bbRouterLink directive and RouterService.navigate method have the exact same API, except that they automatically add the correct router outlet name to the navigation

Backbase in Angular World - Using bbRouterLink directive

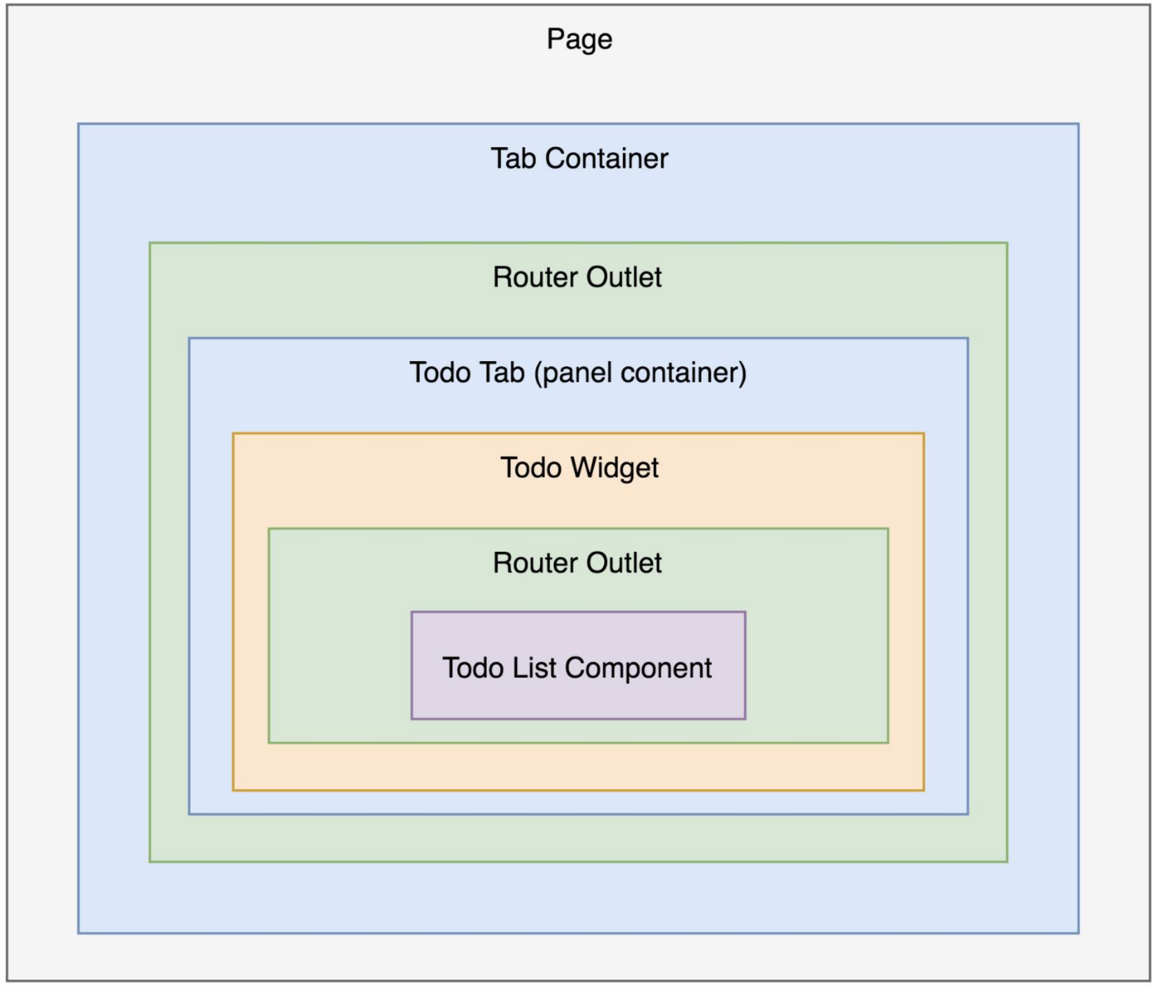
```
@Component({
   selector: 'bb-todo-widget',
   template: `
       <a [bbRouterLink]="edit/1">Edit Todo #1</a>
       <bb-router-outlet></bb-router-outlet>
   `,
   ...
})
@RoutableWidget(...)
export class TodoWidgetComponent {
...
}
```

Backbase in Angular World - RouterService.navigate Method

```
@Component({
   selector: 'bb-todo-widget',
   template: `
       <a (click)="editTodo(1)">Edit Todo #1</a>
       <bb-router-outlet></bb-router-outlet>
   `,
   ...
})
@RoutableWidget(...)
export class TodoWidgetComponent {
   constructor(private RouterService routerService) { }

   function editTodo(id: number) {
       this.routerService.navigate(['edit', id]);
   }
}
```

Backbase in Angular World - Example Use Case



BACKBASE

Inter-Widget Communication

Backbase in Angular World - Inter-Widget Communication

In order to communicate two widgets we will need:

1. **A widget that outputs some information (eg: account id)**
2. **A second widget that reads that information and performs an action**

- Outputting information can be achieved by using Angular Outputs.
- The target widget can read data from the URL (That represents application's router **state**)

How to connect them?

Backbase in Angular World - Output Handlers

- Angular service
- Subscribes to an output stream of a widget
- Performs some action when the output stream emits
- Configurable via the Widget’s model.xml

Backbase in Angular World - Output Handlers

```
<property name="output.{{output-name}}">
 <value>{{output-handler}}:{{config}}</value>
</property>
```

output-name: is the name of the @Output of the widget component

output-handler: is the key identifying which output handler to pass the output-stream to (we'll use the value "navigation" for the default navigation handler, which is part of core-ang)

config: is passed to the output handler to configure the action to take. The format of the config depends on the output handler.

Backbase in Angular World - Output Handlers

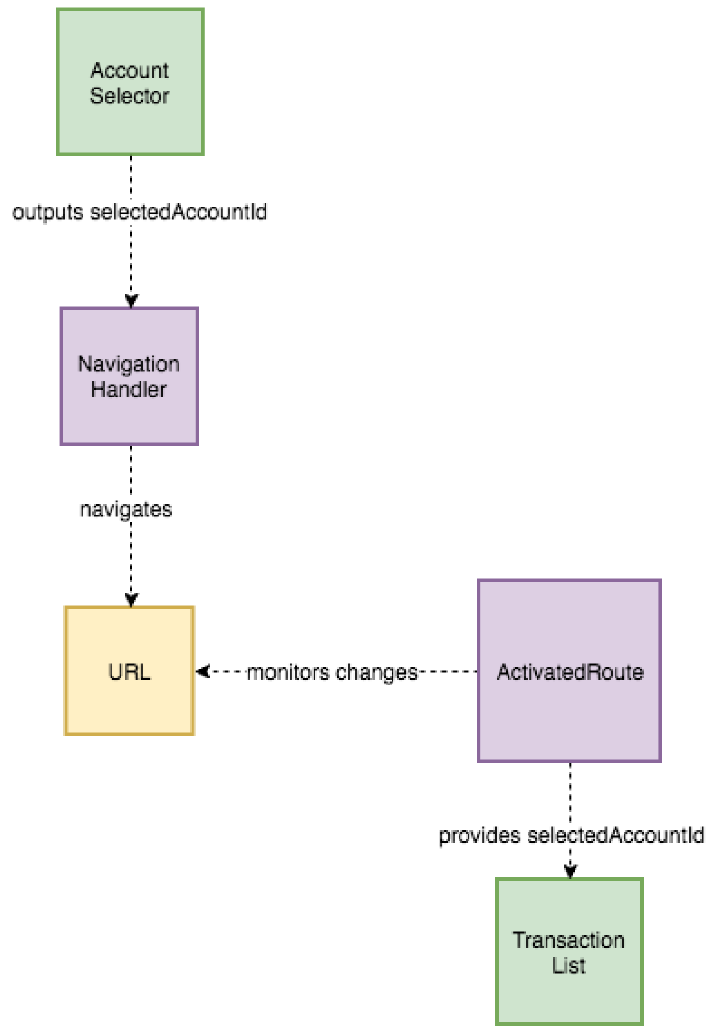
In the case of the "navigation" handler, the config is the *instance* name of the container where the target widget is placed.

Eg: to configure the product summary widget to navigate to the transaction list, you should set the model property as

```
<property name="output.selectedAccount">
 <value>navigation:transaction-deck-123</value>
</property>
```

*Where transaction-list-123 is the instance name of the transaction deck container panel in the portal. The navigation handler **will automatically navigate to the selected widget**, and **will also set a route parameter** of the same name as the output-name.*

Backbase in Angular World - Output handlers summary



BACKBASE

Workshops

- Creating the **Peachtree Bank Accounts** Page

BACKBASE

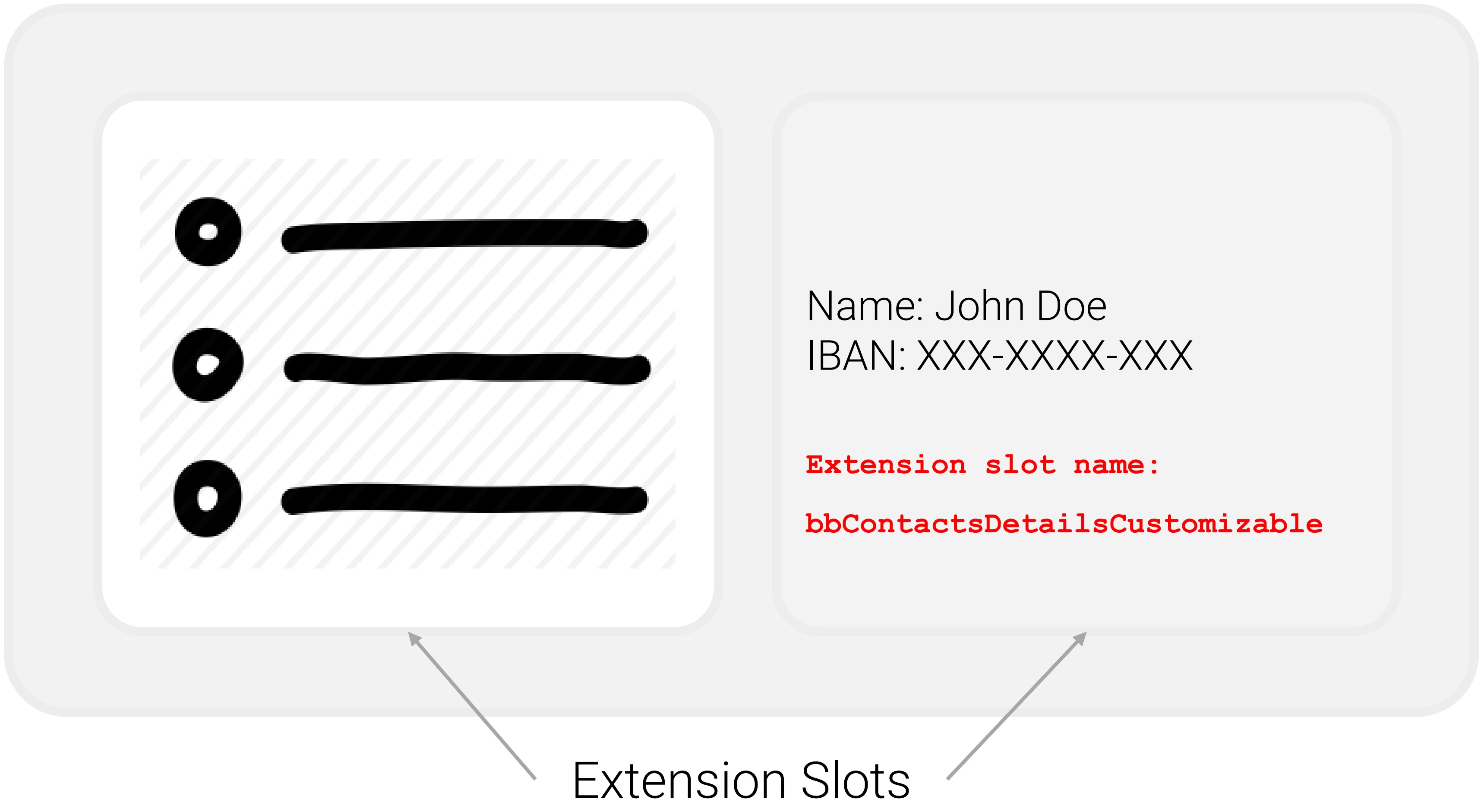
Widget Customization

Backbase in Angular World - Widget Customization

Things that you can **customize** within a widget:

- Properties
- Template
- Data

Backbase in Angular World - Customizing Widget Template



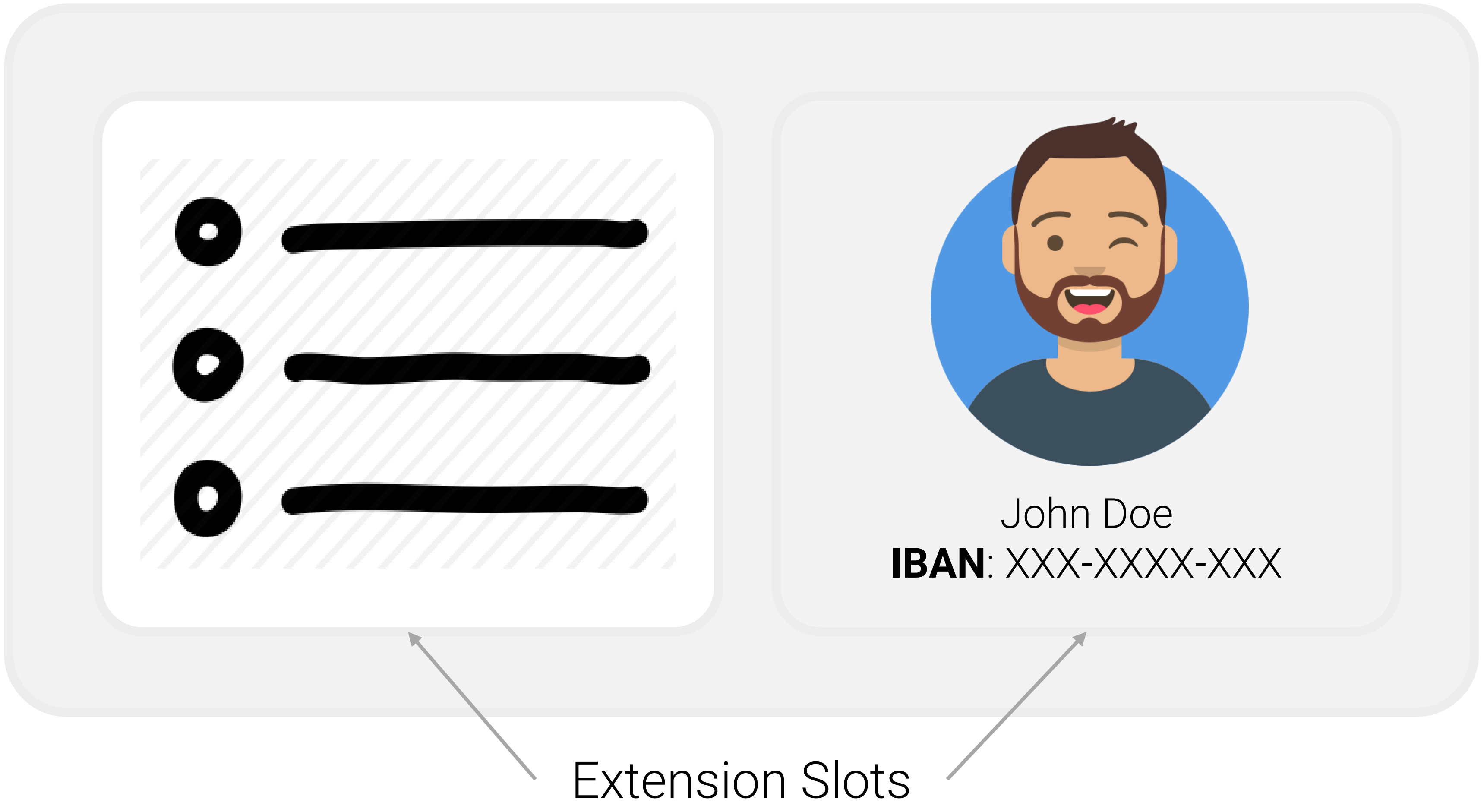
Backbase in Angular World - Customizing Widget Template

You may add custom widget templates in your app.component.ts file. Check the example listed below:

```
<ng-template bbContactsDetailsCustomizable let-hostRef let-contact="context">
   <bb-avatar-ui></bb-avatar-ui>
</ng-template>
```

- You need to know in advance which extension slots are being exposed for that particular widget. (eg: bbDummyContactsDetailsCustomizable)
- Backbase Widgets extension slots are listed on Widget Documentation
- Custom Widgets can (and should) also have extension slots

Backbase in Angular World - Customized Widget Template



BACKBASE

Workshop

Extending **existing widgets**

BACKBASE

Internationalization

WA3 - **i18n**

- Angular i18n
- AOT Support
- Standalone development mode supports different locales

Adding a translation key

```
<h1 i18n="site header|An introduction header for this sample@@introductionHeader">Hello i18n!</h1>
```

- **@@introductionHeader**: This is the custom id for your key.

  This id **MUST** be unique.

- **An introduction header for this sample**: This is your key's description

- **site header**: This is the meaning of your key

Extracting translations keys

```
npm run ng -- xi18n
```

- output: the master messages.xlf file
- xliff format is widely known by translators
- a messages.xlf file per locale (eg: messages.en-US.xlf file)
- npm i ngx-i18nsupport (3rd party)

BACKBASE

Workshop

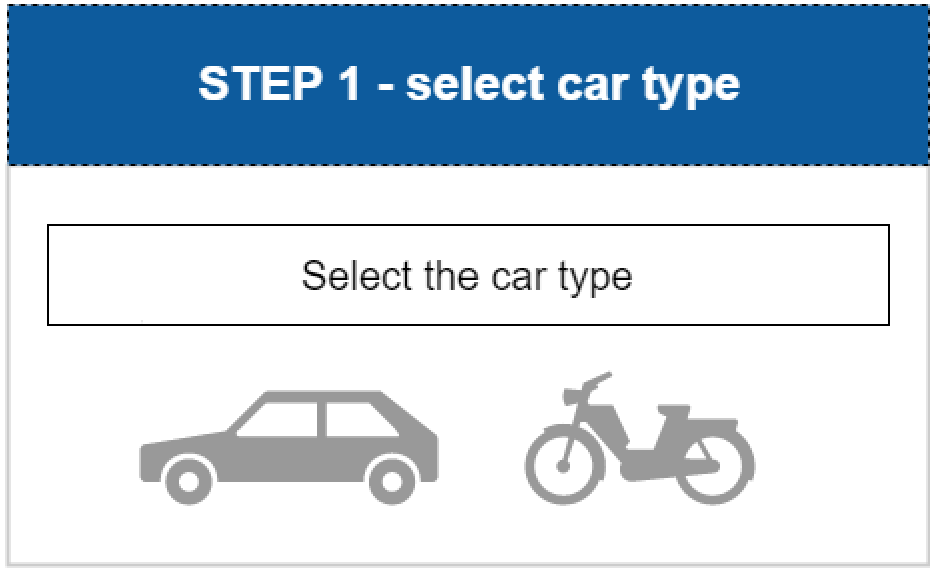
Localizing the customized template

Content Management

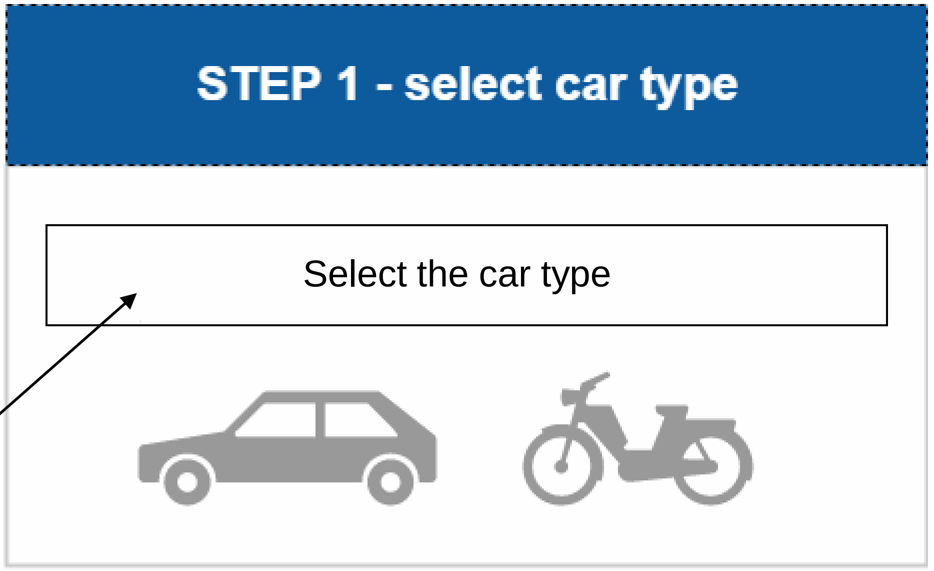
Backbase in Angular World - Content Management

Content can be managed easily and efficiently by making use of WA3 Content Management Features:

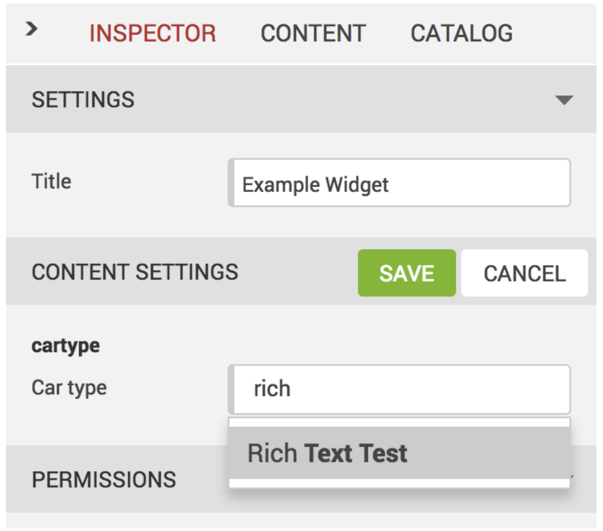
- Content Snippets
- Content Slots
- Independent Production



Backbase in Angular World - Content Management Example



content slot = "cartype"



Title: Example Widget

cartype

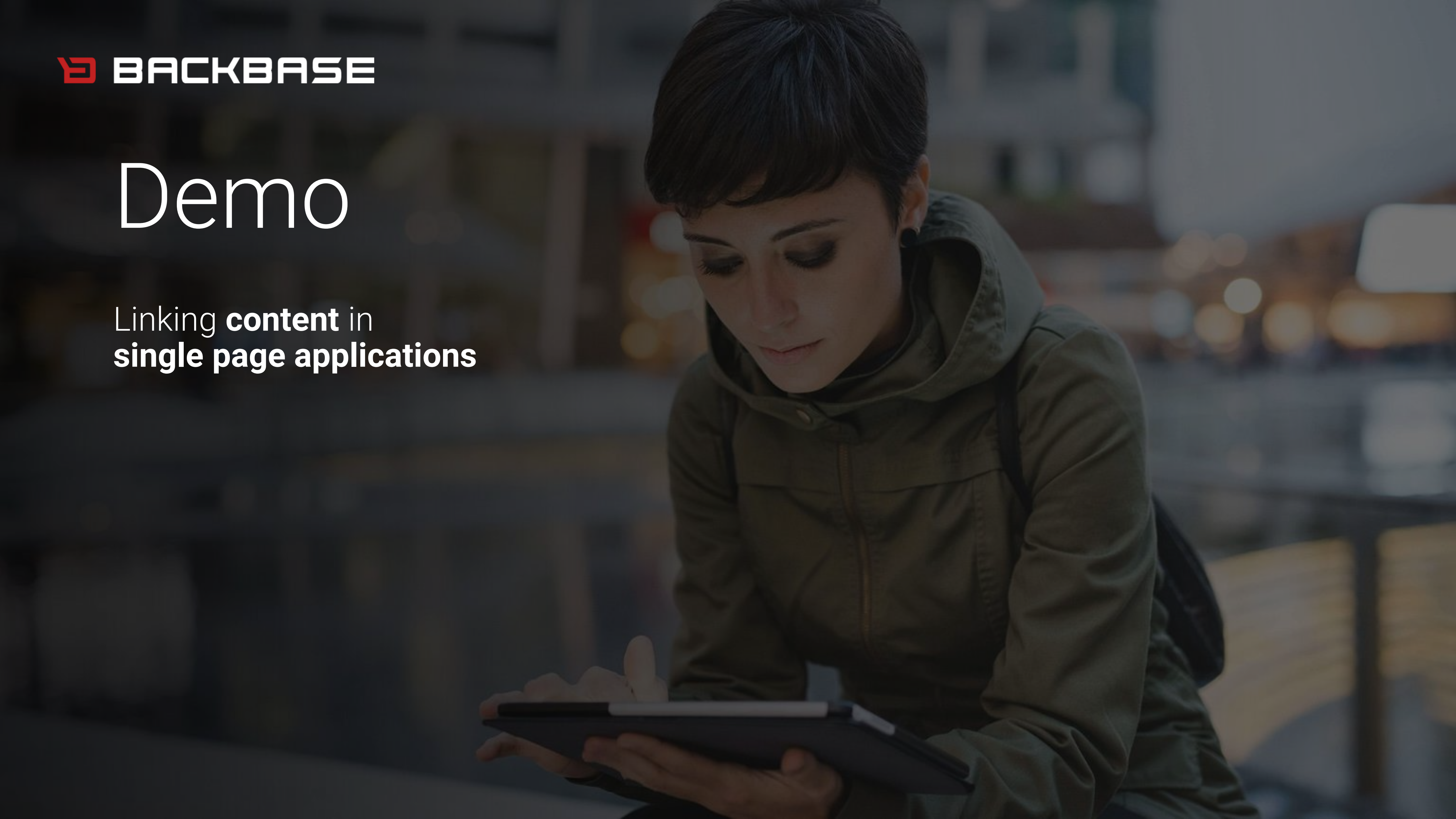
Car type: rich

Rich Text Test

BACKBASE

Demo

Linking **content** in
single page applications



BACKBASE

Workshop

Developing a **Todo Widget**

Backbase Design System
Developing your app
with Theming and UI Components

Back-end

- Service SDK
- Direct Integration
- Content Services
- Targeting
- Security

Front-end

- WA3 Overview
- Backbase in Angular World
- Backbase Design System

Foundation

- Introduction to Backbase
- CXS Basics

Backbase Design System

Understand

A collection of **guidelines and principles** bringing together the relationship between **design and development**.

There are 4 key items that constitute the Backbase Design System:

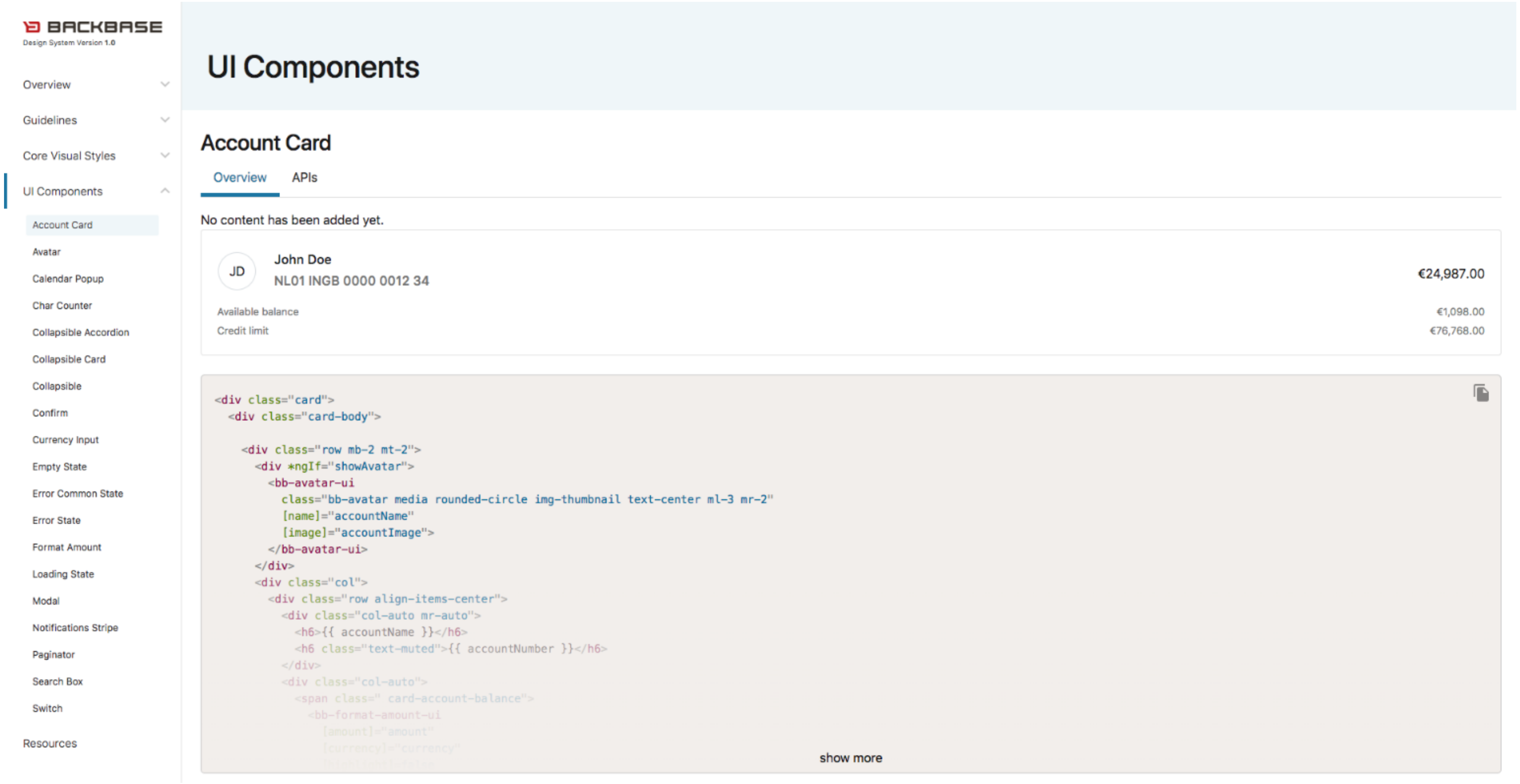
- UI Components — The front-end implementation that constitutes UI Collections (e.g. ui-ng, ...etc)
- Theming — A set of stylistic rules and code that establishes consistency, scalability, and richness in "Look and Feel"
- Sketch Design Kit — The UX Designer's working document for creating visual representations of a project's "Look and Feel"
- Design System (Dev Kit) — The developer and designer point-of-reference for a project's design and UI

BACKBASE

UI Components

Backbase UI Components Library

Backbase UI



UI Components

Account Card

Overview APIs

No content has been added yet.

JD John Doe
NL01 INGB 0000 0012 34 €24,987.00

Available balance	€1,098.00
Credit limit	€76,768.00

```
<div class="card">
  <div class="card-body">

    <div class="row mb-2 mt-2">
      <div *ngIf="showAvatar">
        <bb-avatar-ui
          class="bb-avatar media rounded-circle img-thumbnail text-center ml-3 mr-2"
          [name]="accountName"
          [image]="accountImage">
        </bb-avatar-ui>
      </div>
      <div class="col">
        <div class="row align-items-center">
          <div class="col-auto mr-auto">
            <h6>{{ accountName }}</h6>
            <h6 class="text-muted">{{ accountNumber }}</h6>
          </div>
          <div class="col-auto">
            <span class=" card-account-balance">
              <bb-format-amount-ui
                [amount]="amount"
                [currency]="currency"
                [highlight]=false
```

show more

BACKBASE

Backbase Theme

Backbase Theme

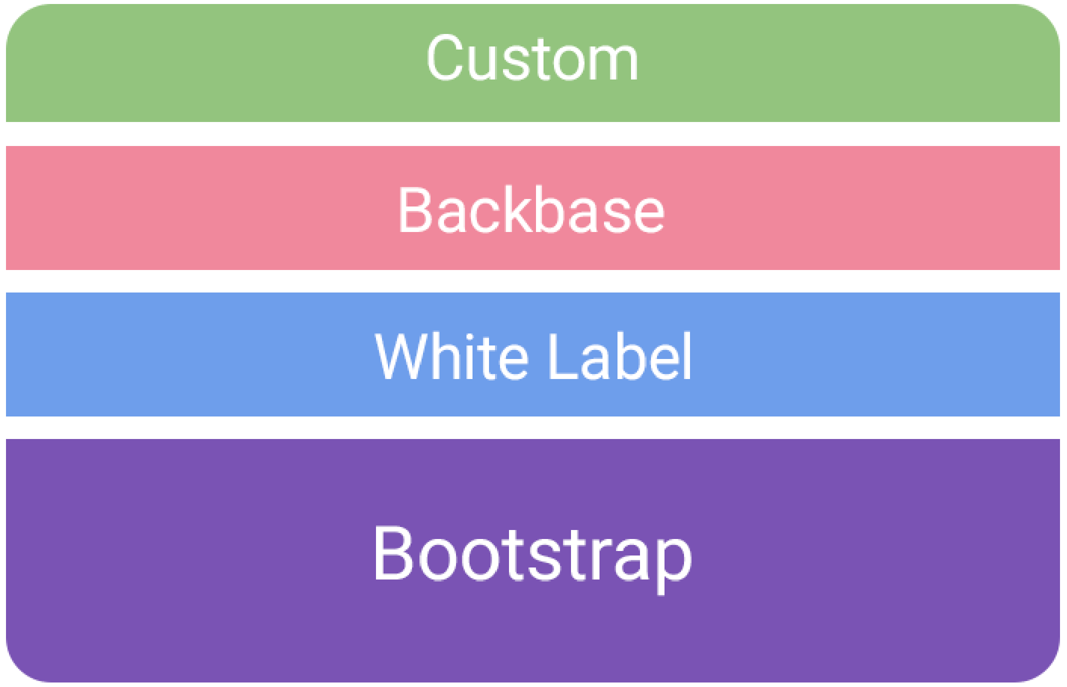
Backbase UI

Understanding Backbase Theme

Backbase provides theming along with its UI Components

Including:

- Default styling and utilities from Bootstrap 4
- Material Design Icon fonts
- Backbase Layer of customization on top of Bootstrap

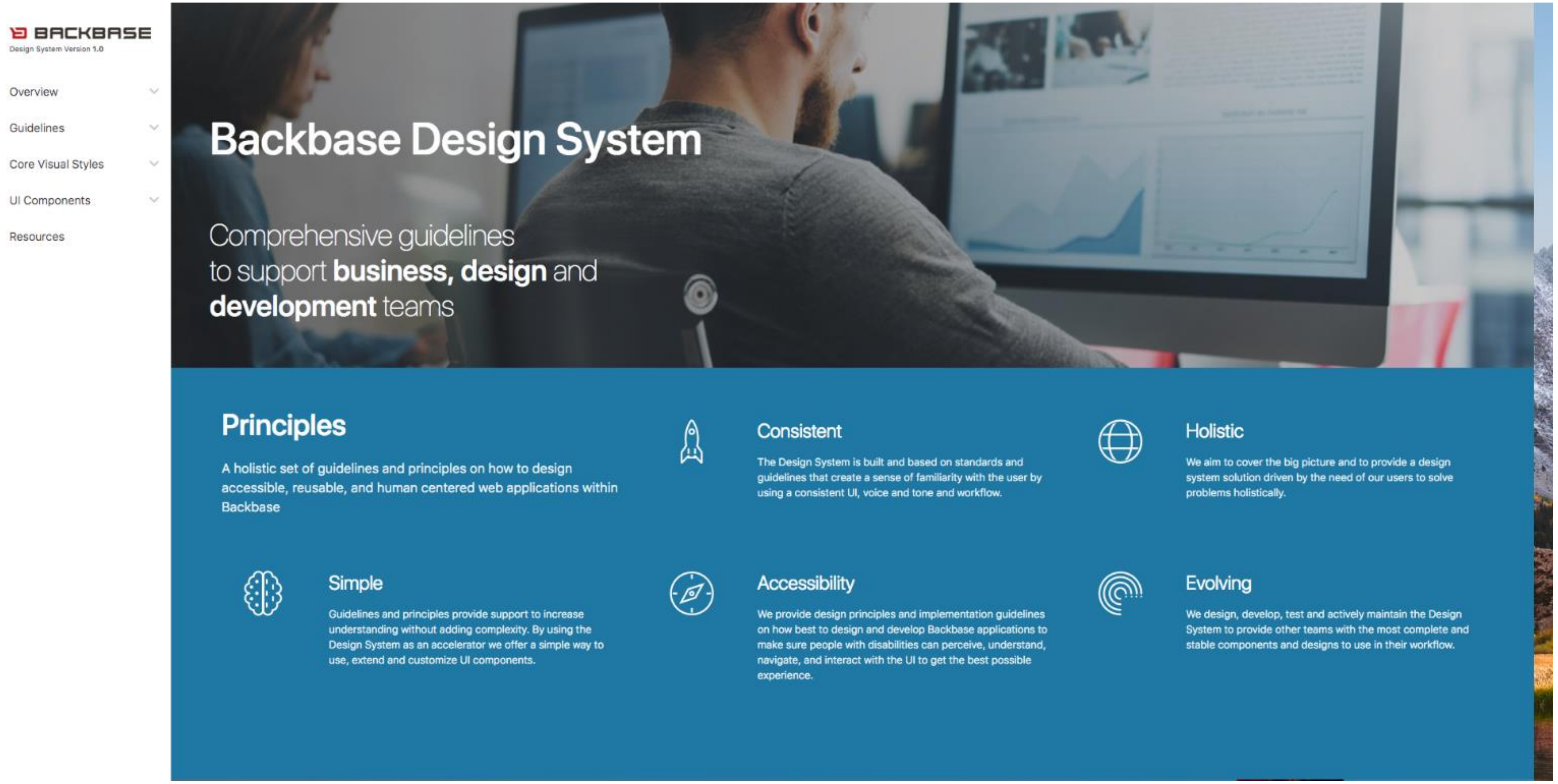


BACKBASE

Design System (Dev Kit)

Backbase Design System

Dev Kit



Backbase Design System

Comprehensive guidelines to support **business, design** and **development** teams

Principles

A holistic set of guidelines and principles on how to design accessible, reusable, and human centered web applications within Backbase

Consistent

The Design System is built and based on standards and guidelines that create a sense of familiarity with the user by using a consistent UI, voice and tone and workflow.

Holistic

We aim to cover the big picture and to provide a design system solution driven by the need of our users to solve problems holistically.

Simple

Guidelines and principles provide support to increase understanding without adding complexity. By using the Design System as an accelerator we offer a simple way to use, extend and customize UI components.

Accessibility

We provide design principles and implementation guidelines on how best to design and develop Backbase applications to make sure people with disabilities can perceive, understand, navigate, and interact with the UI to get the best possible experience.

Evolving

We design, develop, test and actively maintain the Design System to provide other teams with the most complete and stable components and designs to use in their workflow.

BACKBASE

Demo

Backbase **Design System**

BACKBASE

Thank you!

+31 (0) 652 00 0000
www.backbase.com