Industrialisierte Oberflächenentwicklung mit Oracle ADF bei der Bundesagentur für Arbeit

Ralf Ernst
Bundesagentur für Arbeit
Nürnberg

Schlüsselworte
ADF, JSF, JDeveloper, UX, Usability, UI, GUI

Einleitung
Während sich in den letzten Jahren im Bereich des Backends eine immer effizientere und industrialisierte Entwicklung von Komponenten auf Basis diverser etablierter Frameworks wie Spring, EJB3, JPA oder Web-Services etabliert hat, folgt die Entwicklung von graphischen Benutzeroberflächen (GUIs) immer noch vielfach einem eher individuellen arts-and-craft Anspruch. Eine Industrialisierung der Entwicklung von GUIs ist aber ebenso notwendig wie machbar.
Voraussetzung hierfür sind weitgehende Standardisierung der UI, eine verbindliche Referenzarchitektur und die Automatisierung wiederkehrender Aufgaben.
Die IT kann dies nur gemeinsam mit ihrem Kunden erreichen.
Die Motivation, die Voraussetzungen und das Vorgehen für die Einführung einer industrialisierten Oberflächenentwicklung in der BA auf Basis von Oracle ADF werden im folgenden Dokument beschrieben.

Beweggründe / Ausgangslage
Das IT-Systemhaus der Bundesgentur für Arbeit (BA) betreibt eine der größten IT-Systemlandschaften Deutschlands. Neben Standardsoftware für client-zentrierte Office Funktionalitäten, einem ERP System für Auszahlungen und Verwaltung und einer Business Intelligence Plattform für statistische Auswertungen zählen hierzu rund 100 zentralisierte Applikationen mit zusammen mehr als 10000 Masken auf Basis von JEE bzw. C++/Corba, die die spezielle Geschäftslogik der BA und somit ihr eigentliches Kerngeschäft implementieren.
Die mehr als 100.000 Benutzer dieser Applikationen müssen zur Erledigung Ihrer Geschäftsprozesse oft eine Vielzahl von Anwendungen nutzen.
Neben der Neuentwicklung sogenannter rollenbasierter Oberflächen für spezielle Geschäftsprozesse in Servicecentern und Eingangszonen mussten verschiedene geschäftskritische ältere Verfahren auf Basis von C++/Corba und JEE modernisiert werden und aktuelle Anforderungen der Fachseite mit Neuentwicklungen realisiert werden.
Eine Analyse der bestehenden GUI-Architekturen und Entwicklungsprozesse ergab

- Auch durch den Einsatz heterogener Technologien und Frameworks (C++, Swing, JSP, JSF) bedingt, existierte weder ein einheitliches Look&Feel, noch ein einheitliches Bedienmuster der BA-Anwendungen. Dies erschwerte den Benutzern die Aufgabenerledigung erheblich und forderte von Ihnen einen teilweise erheblichen Einarbeitungsaufwand.
- Auf Entwicklungsseite existierte im Gegensatz zum Backend für die Entwicklung von GUIs weder eine gültige Referenzarchitektur noch ein definierter Entwicklungsprozess. Entwickler von Webbasierenden GUIs mussten oft eine Vielzahl von Technologien und Frameworks beherrschen und integrieren (z.B.: Struts, JSP, JavaScript, CSS, HTML)
- Die Zielarchitektur aktueller Web-Frameworks unterscheidet sich in hohem Maße. Vergleicht man die Architektur von beispielsweise javascriptbasierenden HTML5, JavaFX oder JSF findet man kaum Gemeinsamkeiten. Im Gegensatz zu tieferen Applikationsschichten hat sich

auch noch kein definierter Industriestandard entwickelt, was schon die schier unüberblickbaren Vielfalt von GUI-Frameworks und Technologien verdeutlicht. Daher fehlen vielfach auch Pattern und Best-practices zu Themen wie data-binding, state-handling oder event-processing.

Erwartungen an Uis und GUI-Technologien

Die Erwartungen an Benutzeroberflächen und deren Technologie von Benutzern, IT-Architekten und dem Management sind sehr unterschiedlich.

Benutzer vergleichen heutzutage eine Enterprise UI mit Anwendungen, die sie tagtäglich im Internet benutzen. Im Zuge dieser *„consumerization of information technology"* definiert das WWW also mittlerweile, wie eine moderne UI auszusehen hat. Ein populäres Beispiel hierfür ist die Google Searchbar und deren auto-complete Mechanismus.

Ein weiterer Begriff, der in diesem Zusammenhang gebraucht wird, ist die *„joy-of-use"*. Ein Benutzer soll bei der Bedienung einer Benutzeroberfläche Freude haben, die Bedienung einer Anwendung soll also intuitiv und vorhersehbar sein. Anstelle spezieller Lösungen und Tricks ist eine Beschränkung der Anzahl möglicher Bedienmuster (z.B. Master-Detail) auf das unbedingt Notwendige und eine leicht durchschaubare einheitliche Navigationsstruktur, die sich konsistent in allen Anwendungen wiederfindet, ein Schlüssel für benutzerfreundliche Oberflächen.

Schließlich muss eine GUI in der BA den aktuellen Erfordernissen und Gesetzen zur Barriere-freiheit genügen, also auch von (seh-)behinderten Benutzern problemlos bedienbar sein.

IT-Architekten erwarten auch von einer GUI-Technologie ein hohes Maß an Wiederverwendung im Kontext einer ansonsten serviceorientierten Architektur. Dies beinhaltet die Integration mit bestehenden Architekturen, ebenso wie die Unterstützung künftig mehr prozessorientierter Applikationen. Eine zukunftsorientierte GUI-Technologie muss im Intranet wie im Internetportal auf verschiedensten Endgeräten lauffähig sein. Eine saubere Trennung und damit Entkopplung der einzelnen Schichten einer Applikation (*„separation-of-concerns"*) und eine komponentenorientierte Architektur ist hierfür die Voraussetzung.

Aus Sicht des **Managements** bemisst sich der Geschäftswert einer Applikation sehr stark an der UI. Aufgrund der in der BA üblichen langen Produktlebenszyklen (ca. 10 Jahre) muss eine GUI-Technologie zukunftssicher und über diesen langen Produktzyklus supportbar sein. In dieser Zeit müssen Provisionierung der Anwendung und Weiterentwicklung / Wartung kostengünstig möglich sein.

Ein industrialisierterer Ansatz für UI-Entwicklung

Industrialisierung bedeutet in der IT nichts anderes als die möglichst weitgehende Standardisierung von Lösungen, die Automatisierung wiederkehrender Aufgaben.

Im Kontext UI-Entwicklung erfolgte die Standardisierung durch die Definition eines sowohl für die IT als auch dem Bedarfsträger verbindlichen Design-Guides der das Aussehen, den Aufbau, die Navigationsstruktur und die minimal notwendigen Interaktionselemente festlegte.

Daneben erfolgte die Definition einer Referenzarchitektur für UIs und die Produktauswahl des zukünftig einzigen Frameworks für die Implementierung – Oracle ADF.

Um sicherzustellen, dass die entsprechenden Vorgaben auch umgesetzt und nicht im Zuge der Entwicklung neuer Applikationen konterkariert würden, wurden zentrale BA-Skins entwickelt, sowie alle Grundelemente der Masken als verbindliche Templates bereitgestellt.

Die Referenzarchitektur sollte wiederum in einem allgemeingültigen Framework auf Basis von ADF (BADF) implementiert werden, um eine Standardisierung grundlegender Aufgabenstellungen wie data-binding, state-handling und event-processing über verschiedene IT-Projekte hinweg zu erzwingen.

Darüber hinaus war ein einheitliches Tooling zu entwickeln, welches Deployment, Staging und automatisierte GUI-Tests unterstützt.

Der BA Designguide

Der BA-Designguide wurde unter Führung des Bedarfsträgers, also der Fachabteilung der Zentrale der BA entwickelt. D.h. die Fachabteilung definierte, wie die UIs , die diese zukünftig von der IT erhält aussehen und was diese leisten.

Insoweit fungiert der BA-Designguide auch als wichtiges Kommunikationsmedium zwischen IT und Fachabteilung für künftige Anwendungen, da er die möglichen UI-Elemente und Interaktionsmuster zukünftiger BA-Anwendungen enthält und von der Fachabteilung quasi als Katalog für künftige Anwendungen genutzt werden kann, was die Formulierung von Fachkonzepten und Anforderungen an die IT wesentlich vereinfacht.

Das Team zur Definition des BA-Designguides setzte sich neben einem UX-Experten größtenteils aus fachlichen Analysten und Anwendern zusammen.

Ziel war die Definition eines allgemein verwendeten Anwendungsrahmens und einer generischen Navigationsstruktur sowie die Reduktion möglicher Interaktionsmuster (z.B. Master-Detail) auf das unbedingt notwendige Maß.

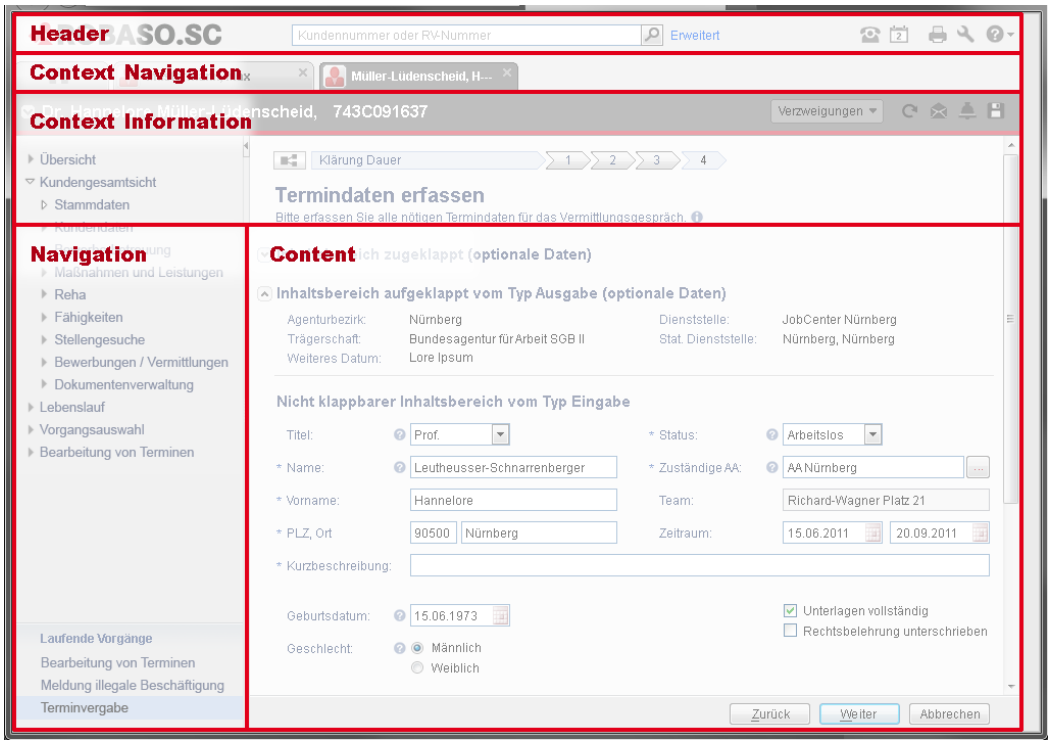

Abb.1:standardisierte Seitenbereiche des BA-Designguides

Begleitet wurde das Team von zwei technischen Experten, die jedoch lediglich zur generellen technischen Machbarkeit der Vorschläge und Entwürfe Stellung nahmen.
Neben allgemeine Regelungen zur Ergonomie und Barrierefreiheit definiert der BA-Designguide nunmehr einen standardisierten Seitenaufbau in Header, Kontext Navigation, Kontext-Übersicht, Navigationsbereich und Inhaltsbereich, sowie ein einheitliches Navigationskonzept.
Die Standardisierung des Inhaltsbereichs erfolgte durch sog. Gestaltungsmuster, die die gewünschte Funktionalität optimal unterstützen.Ziel hierbei war wiederum so wenige Gestaltungsmuster wie möglich zu verwenden, um für den Anwender konsistente, wiedererkennbare Oberflächen auch über verschiedene Anwendungen hinweg zu garantieren. Gestaltungsmuster definieren die verschiedenen Formulare und Tabellen-Views (z.B. Master-Fetail, 2 spaltiges Formular, etc.) innerhalb des Inhaltsbereichs einer Seite.
Weiterhin wurde Aussehen und Verhalten von Interaktions-Komponenten (input-fields, lists, enumerations, choices, texts, buttons, …) spezifiziert, um die gewünschte Funktionalität zu unterstützen und zuletzt die gewünschte Darstellung von Meldungs- und Fehlerdialogen spetifiziert.
Die in Adobe-Photoshop designten Maskenfragmente wurden dann in korrespondierende ADF-Templates übernommen. Mittels ADF-Skinning konnte das gewünschte Look & Feel exakt realisiert werden.

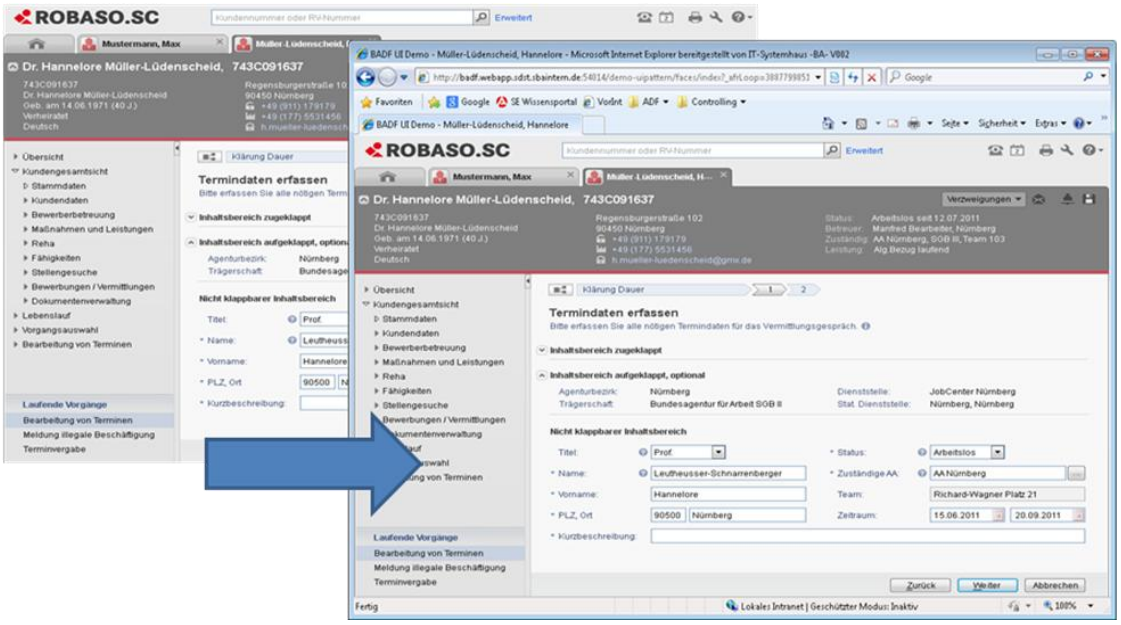

Abb.2:Vom Photoshop Screendesign zur ADF-Anwendung

Produktauswahl
Als GUI-Framework für die Implementierung des Design-Guides wurde Oracle ADF ausgewählt. Für Oracle ADF sprach insbesondere, dass mit Technologien wie wiederverwendbaren und zentral mittels ADFLibrary provisionierbaren Templates in Verbindung mit Skinning die Vorgaben des Design Guides nahezu 1:1 umsetzbar waren.
Daneben konnte mit dem Konzept der Dynamic-Tabs die Forderung der Fachseite erfüllt werden, verschiedene Kontexte (z.B. mehrere Kundendaten, Prozesse, …) in verschiedenen Bereichen der Anwendung zugleich laufen zu lassen.

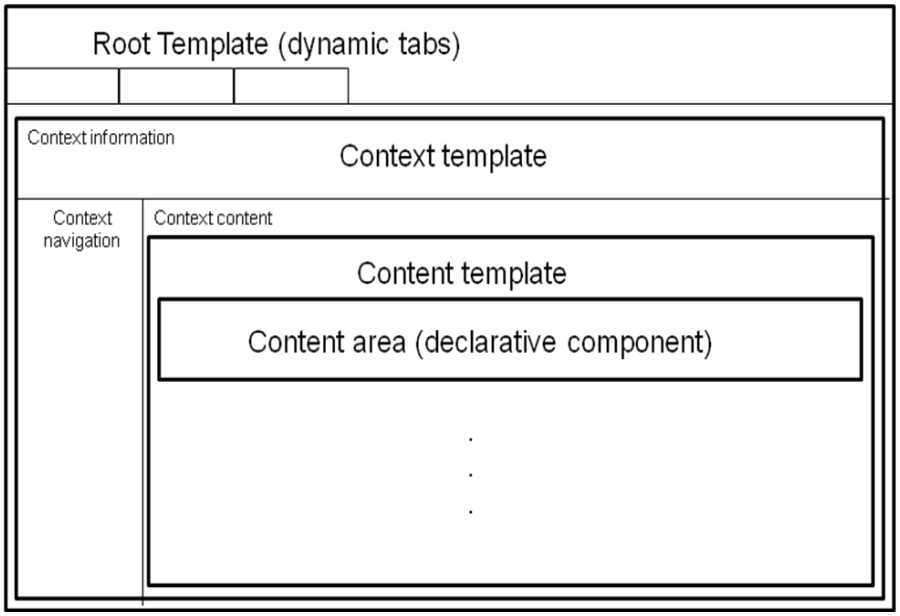

Abb.3:Mapping der Seitenfragmente auf korrespondierende, wiederverwendbare Templates

Daneben war aus Sicht der gewünschten Standardisierung wichtig, dass ADF eines der wenigen GUI-Frameworks am Markt ist, dass in sich komplett und vollständig ist, also ohne zusätzliche Erweiterungen Funktionalitäten wie Ajax, Server-Push und ein mit nativen Clients vergleichbares Dialog-Framework bietet. Es wird kein zusätzliches JavaScript für diese Funktionalitäten benötigt, von den in ADF enthaltene JS-Funktionalitäten wird vollständig abstrahiert.
Ein modularer Aufbau der GUI wird durch wiederverwendbare Controller mit integrierter Zustandsverwaltung und definierten Schnittstellen und den sogenannten contextual events zur losen Kopplung einzelner Seitenregionen ermöglicht.
Vorteilhaft war auch die native Integration von ADF Taskflows in das Portal Oracle Web Center auf dessen Basis der Internetauftritt der BA (http://arbeitsagentur.de) realisiert ist. Die Zukunftsfähigkeit von Oracle ADF ist durch die JSF-Renderer gegeben, da ausgehend von dem selben Source-Code völlig unterschiedliche Codes je nach Endgerät und Browser erzeugt werden können. So wird mittlerweile bei einem angeschlossenen IPad reines HTML5 gerendert, es existieren aber auch spezielle Renderer zur Unterstützung von Tools für Sehbehinderte wie Lunar und Jaws.
Wichtig war für die BA auch, dass Oracle-Produkte wie der Enterprise Manager oder die Fusion Apps vom Hersteller selbst in ADF implementiert wurden, da sich die BA hiervon eine kontinuierliche Weiterentwicklung und Langzeit-Support versprach. Die zwischenzeitlich erfolgte Vorstellung von ADF Mobile bestätigt diese Annahme.
Zuletzt stand mit dem Oracle JDeveloper auch eine IDE zur Verfügung, die eine WYSIWYG Entwicklung in Verbindung mit den zentral zur Verfügung gestellten Komponenten erlaubte.

Referenzarchitektur
Das Framework Oracle ADF implementiert den JSF-Standard und basiert somit auf dem etablierten MVC-Pattern für GUIs. Die Model-Schicht innerhalb von ADF, welche von den darunterliegenden Daten-Services abstrahieren soll wurde von Oracle auf die Zusammenarbeit mit dem proprietären objektrelationalen Mappingframework Oracle ADF Business Components hin optimiert.
Da in der BA jedoch standardmäßig Pojos als DTOs vom Backend zur GUI-Schicht verwendet werden, musste das Binding-Framework entsprechend angepasst werden.

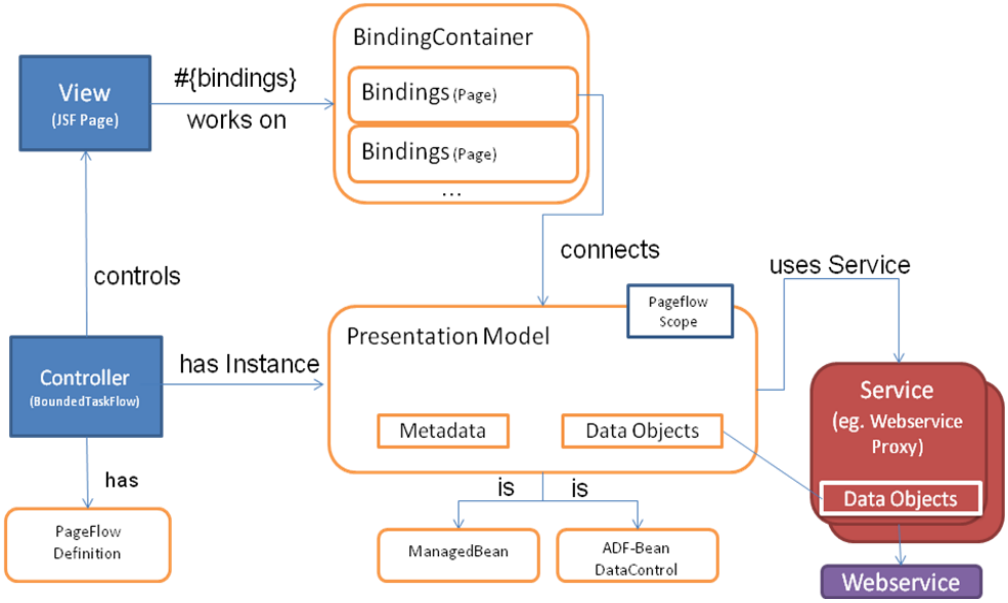

Abb. 4: POJO Binding

Jeder Taskflow wird als bounded Taskflow (statemachine) implementiert und von einem JSF Page-fragment repräsentiert. Jeder Taskflow sollte dabei für die potentielle Wiederverwendung in anderen Applikationen in einer ADF-Library deploybar sein. Wiederverwendbare Teile eines Page-Fragments werden wiederum als deklarative Komponenten implementiert.
Die UI-Komponenten werden im view, request oder backingBean-scope zu ihren Backing-Beans gebunden, der eigentlich UI-state wird mittels Managed-Beans im pageFlow, session oder application-scope gehalten.
Ein weiterer wichtiger Grundsatz ist, kein HTML-markup in den JSF-Seiten zu verwenden, da ansonsten die durch den jeweiligen JSF-renderer gegebene Plattformunabhängigkeit verloren geht.

BADF

BADF implementiert den Design Guide auf Basis der Referenzarchitektur.Motivation für das Framework war, zu erzwingen, dass diese Vorgaben tatsächlich in den Projekten umgesetzt wird Grundfunktionalitäten wie die Navigation oder Exception-Handling werden mittels sogenannter Taskflow Templates zentral bereitgestellt. Seiten werden durch dem Design-Guide entsprechende page-templates und deren korrespondierenden bounded-taskflows implementiert.
Sämtliche Templates, deklarative Komponenten und Skins werden zentral mit ADF-Libraries bereitgestellt und stehen den Entwicklern im JDeveloper sofort zur Verfügung.
Es existieren zwei unterschiedliche Skins für BA-Anwendungen, optimierte Skins für Sehbehinderte (insbesondere optimiert für das Tool Lunar) sowie Skins für Internet-Benutzer.

BADF implementiert die Referenzarchitektur bezüglich Binding und Zustandsverwaltung von Java Pojos und EJBs, liefert Supportklassen für Contextual Events und enthält Basisklassen für immer wiederkehrende Fälle wie z.B. Exception Handling. Es wurde eine Security Integration mit einem eigenem OPSS provider basierend auf JAAS und Java Standard-Security realisiert.
Zuletzt wurde eine Werkzeugstraße für den zentralen Buildprozess auf Basis von ojdeploy, Continuous Integration und automatisierte Tests implementiert.

(B)ADF im realen Leben
Mittlerweile sind drei Projekte mit Hilfe des Design-Guides und BADF erfolgreich realisiert und in Produktion, mehrere weitere Projekte sind aktuell kurz vor dem Go-Live. Oracle ADF als Enabler hierfür hat in den letzten beiden Jahren erheblich an Reife gewonnen und die Benutzerakzeptanz ist gut.
Die Vereinheitlichung der UIs unterschiedlicher Verfahren mit dem Design-Guide und BADF hat sich bewährt, Änderungen im Design-Guide z.B. bzgl. Style können zentral vorgenommen werden und gelten für alle Produkte und Projekte, die BADF nutzen.
Die gewählte Referenzarchitektur unterstützt dabei sowohl prozessunterstützende Anwendungen wie traditionelle eher datenzentrierte CRUD-Anwendungen.
Schon zu Beginn der Entwicklung ist das spätere Zieldesign klar erkennbar, wie folgender Screen-Shot verdeutlicht:

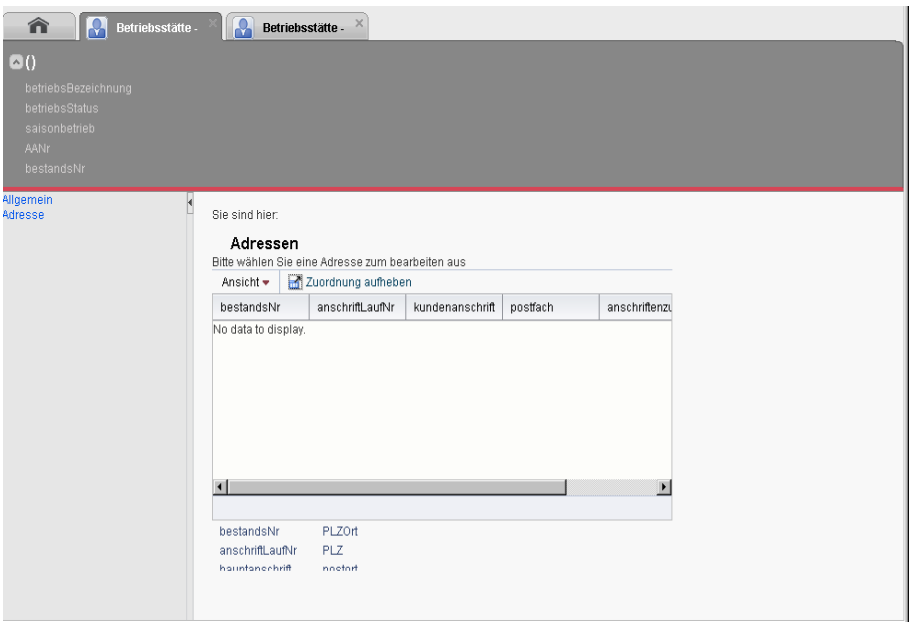

Abb. 5: Screenshot eines Entwicklungsprojekts mit BADF in einer sehr frühen Phase

Zusammenfassend bleibt festzustellen, dass eine industrialisierte GUI-Entwicklung durchaus möglich ist. Voraussetzungen hierfür sind eine größtmögliche Standardisierung der UI, die Einhaltung einer strikten Referenzarchitektur und die weitgehende Automatisierung bzw. Implementierung von wiederkehrenden Aufgaben.
Festzuhalten bleibt, dass der gewünschte Standardisierungsgrad nur zusammen mit der die IT beauftragenden Fachabteilung gelingen kann, da hierdurch natürlich die Gestaltungsmöglichkeiten einer Anwendung –gewollt- eingeschränkt werden. Dies nahm die Fachabteilung der BA angesichts der Vorteile wie massiver Kostenersparnis in der Entwicklung und einer durch die Vereinheitlichung optimierten Benutzerfreundlichkeit (intuitiv, sauber, leicht erlernbar) in Kauf. Die IT erreicht außer einer signifikanten Verringerung des Entwicklungsaufwands bei der Entwicklung von GUIs auch die Vereinfachung von Test und Betrieb, sowie verbesserte Supportbarkeit im Fehlerfall und erheblich verringerte Aufwände bei Weiterentwicklung und Pflege im zukünftigen Applikationslifecycle.
Oracle ADF fungierte bei diesem Vorgehen als Enabler, die Konzepte des BA-Design-Guides waren hiermit vollständig und exakt umsetzbar.
Eine Übertragbarkeit des in der BA gewählten Ansatzes hängt allerdings insbesondere von der Kundenbasis und der Anzahl und Größe der zu unterstützenden Applikationen ab und sollte keinesfalls als allgemeingültig verstanden werden.

Kontaktadresse:
Ralf Ernst
IT-Systemhaus der BA
Regensburgerstr. 104
D-90478 Nürnberg

Telefon: +49 (0) 0911-179 2379
Fax: +49 (0) 0911-179 904765
E-Mail ralf.ernst@arbeitsagentur.de
Internet: www.arbeitsagentur.de