Hochschule für Technik und Wirtschaft Dresden
Fakultät Geoinformation
Masterstudiengang Geoinformation und Management

Masterarbeit

Bereitstellung eines GUI-Prototyps für ein webbasiertes archäologisches Informationssystem unter besonderer Berücksichtigung von Usability-Aspekten

Eingereicht von: Nadine Magdalinski
Seminargruppe: 10/063/71
Matrikelnummer: 28941

1. Gutachter: Prof. Dr.-Ing. Frank Schwarzbach (HTW Dresden)
2. Gutachter: Dr. Felix Schäfer (Deutsches Archäologisches Institut)

Eingereicht am: 18.12.2013

Inhaltsverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
BGG	Behindertengleichstellungsgesetz
BITV	Barrierefreie Informationstechnik-Verordnung
BTU	Brandenburgische Technische Universität
CISAR	Cottbuser Informationssystem für Archäologie und Bauforschung
COTS	Components Off The Shelf
CSS	Cascading Style Sheet
DAI	Deutsches Archäologisches Institut
DFG	Deutsche Forschungsgemeinschaft
EKIDES	Ergonomics Knowledge and Intelligent Design System
GIMP	GNU Image Manipulation Program
GIS	Geoinformationssystem
GPL	General Public License
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ID	Identifikator
iDAI.field	kein Akronym
IEA	International Ergonomics Association
IfE	Ingenieurbüro für Ergonomie Prof. Schmidtke
iOS	kein Akronym, Apple iOS, früher iPhone OS oder iPhone Software
JS	JavaScript
JSON	JavaScript Object Notation
JSP	Java Server Pages
MathML	Mathematical Markup Language
MVC	Model-View-Controller
OO	Objektorientierung
OOP	Objektorientierte Programmierung

OpenInfRA	Open Information System for Research in Archaeology
REFA	Verband für Arbeitsgestaltung, Betriebsorganisation und Unternehmensentwicklung
RM-ODP	Reference Model of Open Distributed Processing
SQuaRE	Software product Quality Requirements and Evaluation
UTF-8	Universal Character Set Transformation Format 8-bit
VDI	Verein Deutscher Ingenieure
W3C	World Wide Web Consortium
WHATWG	Web Hypertext Application Technology Working Group
WYSIWYG	What You See Is What You Get
XML	Extensible Markup Language

1 Einleitung

Bei OpenInfRA (Open Information System for Research in Archaeology) handelt es sich um ein webbasiertes archäologisches Informationssystem, das zurzeit in dem gleichnamigen Projekt der Deutschen Forschungsgemeinschaft (DFG) in Zusammenarbeit zwischen der Brandenburgischen Technischen Universität (BTU) Cottbus-Senftenberg, der Hochschule für Technik und Wirtschaft (HTW) Dresden und dem Deutschen Archäologischen Institut in Berlin (DAI) entwickelt wird.

Die Anwendung OpenInfRA integriert verschiedenste digitale Daten der Archäologie und der Bauforschung in einem Gesamtsystem, das zur Informationsauskunft genutzt wird. So werden z. B. Ausgrabungsdaten aus verschiedenen existierenden Datenbanken und Bilder mit digitalen Karten und wissenschaftlichen Publikationen verknüpft, die über verschiedene Ansichten über ein webbasiertes System verfügbar gemacht werden.

Nachdem Anfang 2013 die erste Projektphase zur Systemspezifikation abgeschlossen wurde, läuft zurzeit die zweite Projektphase zur eigentlichen Systemimplementierung an. Im Rahmen der Systemspezifikation sind verschiedene Prototypen für isolierte Anwendungsbereiche bzw. Module entstanden, so z. B. ein Prototyp für einen Kontextbrowser, eine auf Geoinformationssystemen (GIS) aufbauende Nutzeroberfläche zur Darstellung dreidimensionaler Daten und ein Webformular zur Barrierefreiheit.

Zurzeit fehlt ein Prototyp, der die Erkenntnisse aus den isoliert entwickelten Prototypen zusammenfasst und integriert. Auch gibt es noch keinen Prototypen, der schwerpunktmäßig das Layout, die Bedienung und das Design der geplanten OpenInfRA-Anwendung zeigt und dabei explizit Usability-Anforderungen umsetzt.

Dieser fehlende Prototyp wird in dieser Arbeit unter besonderer Berücksichtigung der Usability (Gebrauchstauglichkeit) entworfen und implementiert, und für das Projekt OpenInfRA bereitgestellt. Er wird im Folgenden als GUI (Graphical User Interface) Prototyp bezeichnet.

Zu Beginn der Arbeit wird der Stand von Wissenschaft und Technik zu den Bereichen Usability und Webdesign (Kapitel 2) erarbeitet. Nach den grundlegenden Definitionen (Abschnitt 2.1) werden relevante Normen, Gesetze und Verordnungen (Abschnitt 2.2) und Entwurfsprinzipien basierend auf der Mensch-Computer-Interaktion (Abschnitt 2.3) vorgestellt und Anforderungen an die Usability (Abschnitt 2.4), Gestaltungsprinzipen (Abschnitt 2.5) und geeignete Testverfahren (Abschnitt 2.6) für Usability präsentiert.

Anschließend werden die Grundlagen des Prototyping für Software definiert (Kapitel 3), verschiedene Prototyping-Arten (Abschnitt 3.1) und unterstützende Werkzeuge (Abschnitt 3.2) vorgestellt. Den Abschluss des Kapitels bildet die Beschreibung der relevanten Technologien für die GUI-Prototyp-Entwicklung (Abschnitt 3.3).

Um den Kontext für die Umsetzung eines Prototyps für ein webbasiertes archäologisches Informationssystem zu schaffen, wird in Kapitel 4 zunächst das Projekt OpenInfRA vorgestellt. Das schließt die Systemspezifikation aus der ersten Projektphase (Abschnitt 4.1) genauso mit ein wie die Beschreibung der Vorgängersysteme (Abschnitt 4.2) und die bereits entwickelten Prototypen (Abschnitt 4.3).

Kapitel 5 analysiert die OpenInfRA-Systemspezifikation in Bezug auf Usability-Anforderungen und Prototyping, um daraus den Diskursbereich des zu entwickelnden GUI-Prototyps (Abschnitt 5.1) und die relevanten Vorgaben für den Entwurf abzuleiten (Abschnitt 5.2). In Abschnitt 5.3 wird die GUI bzgl. Seitenlayout, Seitenstruktur und Design entworfen und auf das Model-View-Controller-Konzept (MVC) angewendet.

Der Entwurf bildet die Grundlage für die Implementierung des GUI-Prototyps (Kapitel 6) auf Basis der Hypertext Markup Language (HTML), Cascading Stylesheets (CSS) und JavaScript (JS). Besondere Beachtung finden neben den aufgetretenen Problemen die dabei verwendeten Technologien (Abschnitt 6.1), die Umsetzung des MVC-Konzepts (Abschnitt 6.2) und der Designvorgaben des DAI (Abschnitt 6.3). Das Kapitel endet mit einem Review der umgesetzten Anforderungen (Abschnitt 6.4) und einer Beschreibung der zwischen den beiden Iterationen durchgeführten informellen Nutzerbefragung (Abschnitt 6.5).

Abschnitt 6.5 fasst die gewonnenen Erkenntnisse zusammen und bewertet sie. Darüber hinaus wird identifiziert, wie die Erkenntnisse und der implementierte GUI-Prototyp in die aktuellen OpenInfRA-Entwicklungen einfließen können. Dazu zählen neben den technischen Erkenntnissen aus Kapitel 6 insbesondere die in Abschnitt 2.3 erarbeiteten Entwurfsprinzipien zur Mensch-Computer-Interaktion und die Anforderungen an die Usability (Abschnitt 2.4), die gleichzeitig auch für das Testen in OpenInfRA verwendet werden können, und die für die Usability relevanten Normen und Verordnungen.

Der entwickelte GUI-Prototyp kann anhand eines Beispielworkflows (Anhang A) online ausprobiert werden (http://magdalinski.eu/mscmag_v2) und liegt als Quellcode vor (Anhang C).

2 Grundlagen Usability und Webdesign

In diesem Kapitel werden die grundlegenden Begriffe Webdesign, Usability und Ergonomie definiert. Dazu werden relevante Normen, Gesetze und Verordnungen, Entwurfsprinzipien zu Mensch-Computer-Interaktion und Anforderungen zur Usability vorgestellt. Es werden Gestaltungsprinzipien und die Grundlagen zum Usability-Testen beschrieben.

2.1 Definitionen

Webdesign befasst sich einerseits mit dem inhaltlichen und strukturellen Aufbau von Webseiten (Konzeption) und andererseits mit deren Aussehen und der Gestaltung (Layout und Komposition). Dazu zählen auch das Entwerfen und Optimieren von Interaktionselementen und das Beachten von Usability-Aspekten. (Rohles, 2013)

Grundlagen zu Layout, Komposition und Usability finden sich in diesem Kapitel; deren Umsetzung und die Konzeption in Kapitel 5.

Der englische Begriff „Usability“ wird als Gebrauchstauglichkeit, Bedienbarkeit oder Benutzerfreundlichkeit ins Deutsche übersetzt. Diese Übersetzungen drücken allerdings nicht vollständig aus, was der Begriff tatsächlich in Bezug auf die Mensch-Maschine-Interaktion impliziert. Was eigentlich mit Usability gemeint ist, ist das gesamte Nutzererlebnis bei der Handhabung und Interaktion mit einer Software, Webseite oder anderen interaktionsfähigen „Gegenständen“ wie z. B. Prozesse, Bücher oder Werkzeuge („Usability“, 2013). Nielsen und Loranger (2006, S. XVI) definieren Usability als „ein Qualitätsmerkmal, wie einfach etwas zu benutzen ist“. Es gehe darum „wie schnell Menschen die Benutzung eines Gegenstandes erlernen können, wie effizient sie während seiner Benutzung sind, wie leicht sie sich diese merken können, wie fehleranfällig der Gegenstand ist und wie er den Nutzern gefällt“. Nach der Definition bräuchte ein Gegenstand den der Nutzer nicht nutzen kann oder nutzen möchte, gar nicht zu existieren.

Ein weiterer Begriff, der im Zusammenhang mit Web Usability von Bedeutung ist, ist „Ergonomie“. Die International Ergonomics Association definiert Ergonomie folgendermaßen:

> „Ergonomics (or human factors) is the scientific discipline concerned with the understanding of interactions among humans and other elements of a system, and the profession that applies theory, principles, data and methods to design in order to optimize human well-being and overall system performance.“ (International Ergonomics Association, 2013)

Die REFA, der Verband für Arbeitsgestaltung, Betriebsorganisation und Unternehmensentwicklung, definiert Ergonomie als eine „wissenschaftliche Disziplin, die Grundlagen, Regeln und Methoden zur menschengerechten Gestaltung der Arbeit entwickelt und für die Anwendung bereitstellt“ (REFA Bundesverband e. V., 2011).

Neben der reinen Wortbedeutung sind die Begriffe „Usability“ und „Ergonomie“ und deren Definition im Bereich von Software-Anwendungen durch zahlreiche Normen, Gesetze und Verordnungen geprägt.

2.2 Normen, Gesetze und Verordnungen

Die große Bandbreite an Normen und Regelungen im Bereich der Usability für Softwareanwendungen veranschaulich die hohe Bedeutung, die diesem Bereich mittlerweile zukommt. Nachfolgend werden einige der Wichtigsten erläutert.

DIN EN ISO 9241 Ergonomie der Mensch-System-Interaktion

Die DIN EN ISO 9241 (ISO/TC 159, 2011) ist eine Normenreihe, die sich mit Richtlinien zur Mensch-Computer-Interaktion befasst. Von besonderer Bedeutung sind die Teile 11 und 110, die sich mit den Grundsätzen der Dialoggestaltung und den Anforderungen an die Gebrauchstauglichkeit, also der Usability der Software beschäftigen. Es werden sieben Grundsätze zur Dialoggestaltung formuliert:

- **Aufgabenangemessenheit**

  Es sollen geeignete Funktionalitäten eingesetzt und unnötige Interaktionen minimiert werden.

- **Selbstbeschreibungsfähigkeit**

  Die Verständlichkeit des Systems soll durch Hilfen / Rückmeldungen unterstützt werden.

- **Lernförderlichkeit**

  Um eine minimale Einarbeitungszeit zu erreichen, soll der Benutzer angeleitet werden. Es sollen geeignete Metaphern verwendet werden.

- **Steuerbarkeit**

  Der Dialog soll durch den Benutzer gesteuert werden können.

- **Erwartungskonformität**

  Die Anwendung soll konsistent und an das Benutzermodell angepasst sein.

- **Individualisierbarkeit**

  Das System soll sich an den Benutzer und an seinen Arbeitskontext anpassen lassen.

- **Fehlertoleranz**

  Die Funktionsweise des Systems soll auch bei unvorhergesehenen Fehlern aufrechterhalten bleiben.

In Teil 11 werden drei Leitkriterien für die Gebrauchstauglichkeit einer Software bestimmt:

- „**Effektivität** [zur Lösung einer Aufgabe]:

  Maße der Effektivität setzen die Ziele oder Teilziele des Benutzers ins Verhältnis zur Genauigkeit. [...]

- **Effizienz** [der Handhabung des Systems]:

  Maße der Effizienz setzen den erreichten Grad der Effektivität ins Verhältnis zum Aufwand an Ressourcen. Der relevante Aufwand kann psychische oder physische Beanspruchung, Zeit, Material oder monetäre Kosten enthalten. [...]

- **Zufriedenheit** [der Nutzer einer Software]:

  Maße der Zufriedenstellung beschreiben das Ausmaß, in dem Benutzer von Beeinträchtigungen frei sind, und ihre Einstellungen zur Nutzung des Produkts. [...]“

  (ISO, 1998b)

Die Teile 11 – 17 und 110 befassen sich mit Themen der Softwareergonomie und definieren und erläutern Usability- Kriterien, während sich z. B. die Teile 3, 4, 9, 302, 303 und 305 mit Anforderungen an die Hardware beschäftigen. Normen zur Arbeitsumgebung werden in den Teilen 5 und 6 beschrieben.

Verordnung über Sicherheit und Gesundheitsschutz bei der Arbeit an Bildschirmgeräten (BildscharbV)

Diese Verordnung gilt für die Arbeit an Bildschirmgeräten. Der Anhang der Verordnung formuliert relevante Anforderungen zur Gebrauchstauglichkeit u. a. in der Kategorie „Bildschirmgerät und Tastatur“:

> „1. Die auf dem Bildschirm dargestellten Zeichen müssen scharf, deutlich und ausreichend groß sein sowie einen angemessenen Zeichen- und Zeilenabstand haben.
>
> 2. Das auf dem Bildschirm dargestellte Bild muß stabil und frei von Flimmern sein; es darf keine Verzerrungen aufweisen.
>
> 3. Die Helligkeit der Bildschirmanzeige und der Kontrast zwischen Zeichen und Zeichenuntergrund auf dem Bildschirm müssen einfach einstellbar sein und den Verhältnissen der Arbeitsumgebung angepaßt werden können.“
>
> (BildscharbV, 1996)

Weitere maßgebliche Anforderungen werden in der Kategorie „Zusammenwirken Mensch – Arbeitsmittel“ definiert:

> „20. Die Grundsätze der Ergonomie sind insbesondere auf die Verarbeitung von Informationen durch den Menschen anzuwenden.

> 21. Bei Entwicklung, Auswahl, Erwerb und Änderung von Software sowie bei der Gestaltung der Tätigkeit an Bildschirmgeräten hat der Arbeitgeber den folgenden Grundsätzen insbesondere im Hinblick auf die Benutzerfreundlichkeit Rechnung zu tragen:
>
> 21.1 Die Software muss an die auszuführende Aufgabe angepasst sein.
>
> 21.2 Die Systeme müssen den Benutzern Angaben über die jeweiligen Dialogabläufe unmittelbar oder auf Verlangen machen.
>
> 21.3 Die Systeme müssen den Benutzern die Beeinflussung der jeweiligen Dialogabläufe ermöglichen sowie eventuelle Fehler bei der Handhabung beschreiben und deren Beseitigung mit begrenztem Arbeitsaufwand erlauben.
>
> 21.4 Die Software muss entsprechend den Kenntnissen und Erfahrungen der Benutzer im Hinblick auf die auszuführende Aufgabe angepasst werden können."
>
> (BildscharbV, 1996)

Behindertengleichstellungsgesetz (BGG) und Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz (BITV)

Das Behindertengleichstellungsgesetz fordert in § 11 zur barrierefreien Informationstechnik:

> „(1) Träger öffentlicher Gewalt im Sinne des § 7 Abs. 1 Satz 1 gestalten ihre Internetauftritte und -angebote sowie die von ihnen zur Verfügung gestellten grafischen Programmoberflächen, die mit Mitteln der Informationstechnik dargestellt werden, nach Maßgabe der nach Satz 2 zu erlassenden Verordnung schrittweise technisch so, dass sie von behinderten Menschen grundsätzlich uneingeschränkt genutzt werden können. […]
>
> (2) Die Bundesregierung wirkt darauf hin, dass auch gewerbsmäßige Anbieter von Internetseiten sowie von grafischen Programmoberflächen, die mit Mitteln der Informationstechnik dargestellt werden, durch Zielvereinbarungen nach § 5 ihre Produkte entsprechend den technischen Standards nach Absatz 1 gestalten."
>
> (BGG, 2002)

Auf Grundlage dieser Gesetzesvorlage formuliert die BITV (BITV 2.0, 2011) konkrete Anforderungen, wie Barrierefreiheit umgesetzt werden soll, basierend auf vier Grundprinzipien:

> „1 Wahrnehmbarkeit - Die Informationen und Komponenten der Benutzerschnittstelle sind so darzustellen, dass sie von den Nutzerinnen und Nutzern wahrgenommen werden können.
>
> 2 Bedienbarkeit - Die Komponenten der Benutzerschnittstelle und die Navigation müssen bedient werden können.

3 Verständlichkeit - Die Informationen und die Bedienung der Benutzerschnittstelle müssen verständlich sein.

4 Robustheit - Inhalte müssen so robust sein, dass sie von möglichst allen Benutzeragenten, einschließlich assistiver Technologien, zuverlässig interpretiert werden können."

(BITV 2.0, 2011)

Die Bedeutung der BITV für OpenInfRA wurde bereits in einer Bachelorarbeit (Harazim, 2013) der HTW Dresden umfangreich untersucht und dargestellt und wird in der vorliegenden Arbeit nicht weiter berücksichtigt.

Normenreihe 250xx: Software Engineering – Qualitätskriterien und Bewertung von Softwareprodukten (SQuaRE)

In dieser Normenreihe werden Qualitätsanforderungen und Prüfbestimmungen für Software definiert, und wie diese Anforderungen an das Produkt geprüft werden können („ISO/IEC 25000", 2013). Die ISO/IEC 25000 (ISO/IEC, 2005) stellt einen Einstieg in und einen Leitfaden durch die Normenreihe dar und führt in das SQuaRE-Modell ein. Außerdem wird die Reihe in den Kontext verwandter Standard eingeordnet. Eine wichtige Norm dieser Reihe ist z. B. die ISO/IEC 25051 (ISO/IEC, 2006), in der es um die Softwareproduktbewertung und die Qualitätsanforderungen an kommerzielle serienmäßig produzierte Softwareprodukte (COTS) und Prüfanweisungen geht.

DIN EN ISO 14915: Software-Ergonomie für Multimedia-Benutzungsschnittstellen

Die Norm ISO 14915 gliedert sich in drei Teile:

- Teil 1: Gestaltungsgrundsätze und Rahmenbedingungen (ISO, 2002a)
- Teil 2: Multimedia-Navigation und Steuerung (ISO, 2003)
- Teil 3: Auswahl und Kombination relevanter Medien (ISO, 2002b)

Es werden folgende Grundsätze speziell für die Gestaltung von Multimedia-Anwendungen formuliert (und können allerdings auch auf Benutzungsschnittstellen im Allgemeinen übertragen werden):

- Eignung für das Kommunikationsziel

  Die Präsentation der Informationen soll sowohl für die Erreichung der Ziele des Anbieters als auch für die Ziele oder Aufgabe des Benutzers geeignet sein.

- Eignung für Wahrnehmung und Verständnis

  Die übermittelte Information soll leicht verständlich und erfassbar sein. Hier spielen Erkennbarkeit, Unterscheidbarkeit, Klarheit, Lesbarkeit, Konsistenz, Kompaktheit und Verständlichkeit (s. auch ISO, 1998a) eine Rolle.

- Eignung für die Exploration

  Der Benutzer soll mit geringen oder keinen Vorkenntnissen über Art, Umfang oder Struktur der angebotenen Informationen in der Lage sein, die gewünschten Informationen oder Funktionalitäten zu finden.

- Eignung für die Benutzungsmotivation

  Die Anwendung soll durch ihre Gestaltung die Aufmerksamkeit des Benutzers auf sich ziehen und ihn zur Interaktion motivieren.

VDI-Richtlinie 5005: Software-Ergonomie in der Bürokommunikation (zurückgezogen)

Der Verein Deutscher Ingenieure formuliert mit der VDI-Richtlinie 5005 (VDI, 1990) einen übergreifenden Modellrahmen für die Software-Ergonomie. Es werden drei wichtige Eigenschaften definiert, die eine Software-Anwendung erfüllen muss:

- **Kompetenzförderlichkeit**

  Das System soll den Benutzer entsprechend seiner Kompetenzen fordern und fördern, ihn jedoch auch nicht überfordern. Der Kenntnisstand des Nutzers soll durch das System sinnvoll erweitert werden.

- **Handlungsflexibilität**

  Es soll alternative Wege geben eine Aufgabe zu lösen. Auch wenn sich die Aufgabenstellung ändert soll die Arbeit noch effizient und effektiv durchgeführt werden können. Das System soll sich an den Kenntnisstand und an die Leistungsfähigkeit des Nutzers anpassen lassen.

- **Aufgabenangemessenheit**

  Eine bestimmte Aufgabe muss sich mit dem System vollständig bearbeiten lassen. Das sollte schnell und ohne großen Aufwand möglich sein.

Selbst wenn die in diesem Abschnitt beschriebenen Regelungen und Normen in einem Softwareprojekt nicht zwingend berücksichtigt werden müssen, können ihnen dennoch wertvolle Hinweise und Anforderungen zur Usability und Ergonomie bzgl. der Mensch-System-Interaktion entnommen und angewendet werden. Auf ihnen basierend können Leitfäden zur Entwicklung und dem Design von Softwareanwendungen/Webseiten gebildet werden und müssen (bzw. können in diesem Umfang oft gar nicht) nicht in jedem Projekt wieder von Grund auf neu erhoben werden. Sie können außerdem als Grundlage für die Prüfung von Zwischenständen und Endprodukt genutzt werden.

2.3 Entwurfsprinzipien zur Mensch-Computer-Interaktion

Jeder Anwender möchte, dass ein System „gut“ (s. z. B. VDI-Richtlinie 5005, Abschnitt 2.2) zu benutzen ist, sowohl im Sinne von Usability, als auch bzgl. der Funktionalität. Damit dies jedoch der Fall ist, müssen sowohl bereits in der Anforderungserhebung und -analyse und der Spezifikation, als auch beim Entwurf eines Systems einige grundlegende Regeln berücksichtigt werden. Preim und Dachselt (2010) formulieren Prinzipien für den Softwareentwurf, die sich aus der Berücksichtigung der allgemeinen und kognitiven Aspekte der Mensch-Computer-Interaktion ergeben. Diese werden nachfolgend vorgestellt. Teilweise spiegeln sie Konzepte und Regeln wieder, die bereits in Abschnitt 2.2 im Rahmen von Normen und Richtlinien ansatzweise vorgestellt wurden. Der Vollständigkeit halber werden auch diese wieder aufgegriffen und im Rahmen der Prinzipien ausführlicher vorgestellt.

2.3.1 Kenntnis potenzieller Benutzer und ihrer Aufgaben

Während die weiteren Prinzipien nach Preim und Dachselt (2010) in der Reihenfolge ihrer Anwendung im Entwurf unabhängig sind, steht diese hier jedoch an erster Stelle, da sie die Voraussetzung für die weiteren Prinzipen zum Systementwurf darstellen.

Bei der benutzerorientierten Entwicklung eines Systems steht der Anwender im Mittelpunkt. Es ist daher essentiell, die späteren Anwender zu kennen, also einen Überblick über deren Fähigkeiten und Bedürfnisse bzgl. des Systems zu haben. Die Identifikation von Benutzergruppen nach Gelegenheitsnutzern / Vielnutzern, erfahrenen / unerfahrenen bzw. professionellen Nutzer, oder auch die Art der Nutzung (häufige Nutzung, dauerhafte Nutzung) ist wichtig. Ebenso sind Kenntnis über die Analyse von Aufgaben und Arbeitsabläufen der Benutzer zu beachten. Idealerweise (aus Entwicklersicht) handelt es sich bei den identifizierten Workflows um klar strukturierte, hierarchische Abläufe, oft allerdings um unstrukturierte Prozesse und Aktivitäten. Auch die Relevanz und Häufigkeit von Aufgaben spielt eine Rolle.

Nutzerbefragungen potentieller Nutzergruppen können hier Aufschluss geben. Dafür muss eine repräsentative Auswahl von Anwendergruppen für die Befragung getroffen werden.

Oft gestaltet sich der Wissenstransfer jedoch auf Grund mangelnder Kommunikationsbereitschaft oder z. B. der Vertraulichkeit von Informationen als schwierig. Oft sind Fachgebiete sehr komplex und für Außenstehende (Entwickler) schwer verständlich. Zusätzlich können auch identifizierte Benutzergruppen sehr inhomogen sein.

Es bietet sich an, die Dokumentation von Aufgaben, Arbeitsabläufen und Nutzerbedürfnissen in natürlicher Sprache durchzuführen, da dies sowohl leichter für die Benutzer als auch die Entwickler ist. Um relevante Informationen, Suchmöglichkeiten und Arbeitsabläufe in der Benutzerschnittstelle effizient anbieten zu können, muss dieses Wissen möglichst feingranular erhoben werden. Da dieses Wissen nie vollständig erhoben werden kann, ist bei der

Systementwicklung ein gutes Antizipationsvermögen (‚Was könnte vom Anwender im System benötigt werden?') von Vorteil.

(Preim & Dachselt, 2010)

2.3.2 Unterstützung beim Aufbau mentaler Modelle

Bei der Benutzung des Interface soll der Nutzer sich ein mentales Modell vorstellen können, das er mit einer ihm bekannten Umgebung assoziiert. Durch Metaphern werden Analogien zu bekannten alltäglichen Nutzungsszenarien geschaffen. Objekte und Funktionen sollen nicht imitiert sondern abstrahiert werden und so einen möglichst einfachen Transfer von Arbeitsabläufen und eine intuitive Bedienung erlauben. Der Bedarf und die Fähigkeiten des Nutzers bestimmen die Wahl einer passenden Metapher (Stapelkamp, 2010). Tabelle 1 zeigt einige wichtige Interface-Metaphern und deren typischen Verwendungszweck.

Tabelle 1: Wichtige Interface-Metaphern und typische Verwendungszwecke (S. 131, Preim & Dachselt, 2010).

Metapher	Verwendung	Wichtige Bestandteile
Haus-Metapher	Informationssysteme, Spiele	Türen, Gänge, Treppen, Etagen
Raum-Metapher	Informationssysteme, Spiele	Verschiedene Orte, Reisemöglichkeiten, Verkehrsmittel, 3D-Ansichten und 3D-Interaktionsmöglichkeiten
Atlas-Metapher	Lehrsysteme in Bereichen, in denen Bilder dominieren (z. B. Anatomie)	Großformatige detaillierte Bilder, Einhaltung der Konventionen, Bildunterschriften, Seiten
Desktop-Metapher	Bürosysteme	Ordner, Postein- und –ausgang, Mülleimer
Buch-Metapher	Lehr- und Lernsysteme	Seiten, Kapitel, Verzeichnisse
Karten-Metapher	Geografische Informationssysteme	(rasterorientierte) Landkarten, Legenden, Darstellung der einzelnen Informationen entsprechend den Konventionen in gedruckten Dokumenten

Durch den Einsatz von Metaphern kann der Lernaufwand für den Anwender verringert werden. Gleichzeitig wird die Zahl der Fehler bei der Bedienung reduziert und das Beibehalten von Bedienhandlungen unterstützt. Aber auch auf die Entwicklung der Anwendung kann die Verwendung von Metaphern einen strukturierenden Einfluss haben.

Metaphern für die Verwendung können z. B. durch die Erstellung von Diagrammen typischer Bedienhandlungen gefunden werden. Im Einsatz sollten Tests durchgeführt werden, welche Assoziationen die Nutzer bei der Anwendung des Systems entwickelt und wie die mentalen Modelle der Nutzer aufgebaut sind.

(Preim & Dachselt, 2010)

2.3.3 Terminologie der Benutzer verwenden

Während es auch in anderen Bereichen der Systementwicklung natürlich von Bedeutung ist, ist es bei der Benutzungsschnittstelle geradezu unabdinglich, die Sprache des Anwenders zu verwenden. Die Verwendung unüblicher Begriffe, von Mehrdeutigkeiten oder die falsche Verwendung von Begriffen sollten vermieden werden, da dadurch die Akzeptanz der Anwendung erschwert wird. Die Fachsprache der Anwender soll hingegen konsequent und korrekt zur Anwendung kommen und auch bei visuellen Komponenten (z. B. Gestaltung von Icons) berücksichtigt werden. Auch inhaltlich sollte man die Denkweise der Nutzer verstehen und antizipieren. Die Verwendung der richtigen Terminologie erhöht die Akzeptanz der Anwendung und stärkt das Vertrauen in ihre Kompetenz.

Um einen Überblick über die Fachterminologie der Anwender zu bekommen, muss in Aufgaben- und Benutzeranalysen auf wichtige Begriffe, Symbole und grafische Darstellungen geachtet werden. Auch eine gezielte Analyse z. B. von Texten aus dem Anwendungsbereich ist sinnvoll.

Dieses Prinzip sollte projektweit zur Anwendung kommen, d. h. auch in Gesprächen mit Benutzern, bei der Formulierung von Hinweisen in Statuszeilen und Hilfesystemen und in allen Dokumenten.

(Preim & Dachselt, 2010)

2.3.4 Reduktion der kognitiven Belastung

Die Aufmerksamkeit und das Arbeitsgedächtnis eines Benutzers sind begrenzt. Das macht sich vor allem dann bemerkbar, wenn sich der Nutzer neben seinen eigenen Zielen noch weitere Informationen merken oder er zwischen vielen unterschiedlichen Ansichten und Kontexten wechseln muss. Daher sollten Dialoge und Formulare einfach gestaltet und gut strukturiert sein. Komplexe Arbeitsabläufe können evtl. in mehrere kleine Schritte zerlegt und durch eine sinnvolle Nutzerführung unterstützt werden. Die wichtigsten Einstellungen, Modi und Systemzustände sollten dauerhaft angezeigt werden. Wichtige und häufig genutzte Informationen sollten sinnvoll hervorgehoben werden, um eine aufwändige Suche nach benötigten Informationen zu vermeiden. Um Frustration und / oder Fehlern vorzubeugen, sollten systembedingte Verzögerungen und Wartezeiten weitgehend reduziert werden. Lassen sie sich nicht vermeiden, soll der Nutzer zumindest ein Feedback erhalten.

(Preim & Dachselt, 2010)

2.3.5 Strukturierung der Benutzungsschnittstelle

Wie bereits im vorhergehenden Abschnitt ansatzweise beschrieben, ist eine gute Strukturierung der Informationen für eine effiziente Nutzung durch den Anwender unerlässlich. Zusammengehörige Informationen sollen direkt erkannt und verarbeitet werden können. Die Struktur der Benutzungsschnittstelle wird zum einen durch das generelle Layout der Applikation und zum anderen durch die Detailgestaltung von Dialogen, Menüs und Werkzeugleisten bestimmt. Dabei sollte man sich vom Groben ins Feine vorarbeiten und zuerst Entscheidungen zur Strukturierung des Layouts in Inhaltsbereiche, Dialoge und Toolbars fällen. Erst dann sollte man sich mit der Strukturierung auf Detailebene (Auswahl und Anordnung von Bedienelementen, Wahl von Farben, Platzierung von Linien und Dialogelementen, usw.) beschäftigen. Sinnvolle Maßnahmen zur Strukturierung sind u. a.:

- Zusammenfassen von Bedienelementen in Gruppen und Visualisieren dieser Organisation
- Bildung von Gruppen durch Beschriftung, Rahmen und räumliche Nähe
- Kommandos hierarchisch strukturieren in Menüs (gleichzeitig sichtbare Menüeinträge befinden sich auf einer Ebene und gehören inhaltlich zusammen)
- Anordnung von Icons und zusammenfassen zu Gruppen (Werkzeugleisten)

Generell sollten Gestaltgesetze (Thesmann, 2010) berücksichtigt werden, wie z. B.:

- Markante symmetrische Formen und eine ausgewogene Verteilung werden als angenehm wahrgenommen
- Objekte gleicher Form (Schrift, Farbe) werden als gleichrangig wahrgenommen

Da es nicht *die* richtige Strukturierung einer Benutzerschnittstelle gibt, bietet sich ein prototypbasiertes Verfahren zur schrittweisen Feinspezifizierung und Abstimmung verschiedener Entwürfe einer Benutzerschnittstelle mit Entwicklern und ggf. Anwendern an.

(Preim & Dachselt, 2010)

2.3.6 Kombination visueller und textueller Elemente

Eine Benutzerschnittstelle vereint häufig bildhafte und textuelle Darstellungen in unterschiedlichen Ausprägungen. Manchmal sind Bilder zur Veranschaulichung eines Sachverhalts besser geeignet als Text und andersherum. Oft ist eine Kombination von beschrifteten und kommentierten Bildern am sinnvollsten. Daher sollten Menüeinträge z. B. durch einprägsame Icons ergänzt werden.

(Preim & Dachselt, 2010)

2.3.7 Sichtbarkeit von Systemzuständen und möglichen Aktionen

Bei einer grafischen Benutzeroberfläche „existieren“ für den Nutzer nur die Interaktionsmöglichkeiten, die er sehen kann. Daher müssen Systemzustände und die im jeweiligen Systemzustand möglichen Handlungen sichtbar und erkennbar sein, um eine fehlerhafte Bedienung zu vermeiden. Lässt sich ein System nur mit Hilfe unterschiedlicher Modi realisieren, sollen diese klar erkennbar und unterscheidbar sein. Verfügbare Aktionen der Modi sollen ein- bzw. ausgeblendet werden. Ansonsten sollten z. B.

- wichtige, durch den Nutzer gesetzte Parameter immer sichtbar sein,
- nicht nutzbare Funktionen deaktiviert werden (begründeter Hinweis an den Nutzer),
- Statusanzeigen (bereits im Layout vorsehen) verwendet werden und
- zulässiger Wertebereich und Strukturen bei einer Eingabe erkennbar sein.

(Preim & Dachselt, 2010)

2.3.8 Angemessene Rückkopplung

Bei der Bedienung des Systems soll der Nutzer als unmittelbare Reaktion auf seine Handlungen ein sinnvolles aussagekräftiges Feedback erhalten, sonst kann es zu Fehlhandlungen des Nutzers wie z. B. Doppelaktionen kommen. Bei mehreren Aktionen des Nutzers soll das Feedback in der entsprechenden Reihenfolge der Aktionen angezeigt werden. Folgende Feedbacks sind u. a. möglich:

- Buttons verändern ihr Aussehen, wenn sie aktiviert werden
- Menüeinträge werden hervorgehoben, wenn sie aktiviert werden
- Selektionen werden hervorgehoben (auch in Grafiken und Text)
- Cursor in Sanduhr verwandeln oder quantitative Anzeige als z. B. Ladebalken (bei Verzögerungen von Kommandoausführungen)

Insbesondere bei länger andauernden Aktionen sind skalierbare Rückkopplungen wünschenswert, die sich z. B. minimieren lassen.

(Preim & Dachselt, 2010)

2.3.9 Konsistenz

Das Prinzip der Konsistenz einer Anwendung wurde bereits in der ISO 9241 unter dem Begriff Erwartungskonformität vorgestellt (s. Abschnitt 2.2), d. h. die Bedienung und die Dialoge einer Applikation sollen bestimmten Regeln folgen und die Erwartungen des Anwenders erfüllen. Durch sinnvolle automatisch- oder nutzer-initiierte Vorschauen (Previews) können evtl. „Überraschungen“ des Nutzers entschärft werden.

Inkonsistenzen, vor allem zwischen gleichartigen Bedienelementen führen häufig zu Bedienfehlern. Bewusst eingesetzt, können sie allerdings auch die Unterscheidbarkeit unterstützen. Nicht gleichartige Bedienelemente müssen sich deutlich in der Darstellung oder zumindest im Feedback (z. B. selektierbare und nicht selektierbare Teile einer Grafik) voneinander abheben. Folgende Konsistenzen sind zu beachten:

- Sprachliche Konsistenz

  (verwendete Terminologie, z.B. Benennung von Buttons, Menüeinträgen, Dialogelementen, in Tooltips und Nachrichtenfenstern)

- Strukturelle Konsistenz

  (Anordnung von Bedienelementen, z. B. von Einträgen in Menüs und von Buttons in Dialogen und Toolbars)

- Grafische Konsistenz

  (Nutzung von Farben (Ränder von Icons, Titel von Fenstern, Schriftfarbe,…), Nutzung von visuellen Effekten (Schatten, Linienstärke), Einsatz von Icons und grafischen Elementen)

- Interaktionskonsistenz

  (Verhalten des Systems, einheitliche Reaktionen auf Betätigung bestimmter Funktionstasten, Maustasten usw.)

Für die Umsetzung einer konsistenten Anwendung ist in der Softwareentwicklung eine frühe und sorgfältige Planung notwendig. Die erforderlichen Regeln müssen feingranular dokumentiert werden und während der Entwicklung allen Beteiligten bekannt sein. Evtl. können Bedienelemente, Basisklassen usw. bereits vor der Implementierung definiert und zur Verfügung gestellt werden.

Der Nachweis und die Verbesserung der Erwartungskonformität einer Anwendung kann nur durch umfangreiches gezieltes Testen erreicht werden. Benutzer sollen dabei typische Workflows bearbeiten und ihre dabei gemachten Erfahrungen und Erwartungen kommentieren (s. Abschnitt 2.6).

(Preim & Dachselt, 2010)

2.3.10 Abbruch und Rückgängigmachen von Aktionen

Aktionen im System müssen abgebrochen und rückgängig gemacht werden können, damit der Anwender unbefangen arbeiten und Aktionen ausprobieren kann. Die Möglichkeit zum Abbruch einer Aktion ist besonders bei zeitaufwändigen Aktionen wichtig.

Rückgängig machen sollte am besten für mehrere bzw. alle Aktionen einer Sitzung möglich sein. Besonders bei Lösch- und Änderungsvorgängen muss der Nutzer explizit darauf hin-

gewiesen werden, wenn ein Rückgängig machen nicht möglich ist. Gegebenenfalls sollte es auch möglich sein ein Rückgängig machen wieder rückgängig zu machen. Auch das Zurücksetzen auf Standartwerte sollte möglich sein.

(Preim & Dachselt, 2010)

2.3.11 Berücksichtigung von Fehlern

Anwender machen bei der Bedienung eines Systems unweigerlich Fehler. Die Benutzerschnittstelle sollte allerdings nie in einen Zustand kommen können, in der der Benutzer überhaupt keine Möglichkeit mehr sieht, die Arbeit sinnvoll fortzusetzen, bzw. ihm keine entsprechenden Optionen mehr angeboten werden. Arbeitsergebnisse dürfen nicht „aus Versehen“ verloren gehen können, auch nicht durch eigenes Verschulden. Fehler sind oft durch geeignete Gestaltung vermeidbar:

- Vermeidung ungeeigneter Beschriftung in Dialogen, Formularen und Menüs
- Einsatz von geführten Interaktionen (z. B. Wizards)
- Zerlegung komplexer Arbeitsabläufe in mehrere, kleine Schritte
- Erzwingen von syntaktisch korrekten Eingaben
- Auto-Vervollständigung von Eingaben

Zusätzlich zu den Vermeidungsstrategien können Sicherheitsvorkehrungen getroffen werden, die die Fehlbedienungen des Systems verringern bzw. verhindern, wie z. B. das automatische Speichern von Sicherheitskopien oder die Wiederherstellbarkeit von Systemzuständen. Durch Constraints können Freiheitsgrade des Benutzers eingeschränkt werden (z. B. ist ein vollständiges Herausschieben des Cursors aus dem Bildschirm via Mausbewegung nicht möglich).

Durch Fehlermeldungen soll der Benutzer dabei unterstützt werden auftretende Fehler zu bemerken, ihre Ursache zu verstehen und sie zu beheben. Die Meldungen sollen z. B. in Europa benutzerorientiert, konstruktiv und positiv in einem schlichten Stil formuliert sein. Sie sollen so spezifisch wie möglich sein und können zusätzlich Hilfe-Buttons, ausführliche Diagnosen des Fehlers und Hinweise auf Behebung / Vermeidung des Fehlers enthalten. Die Präsentation der Fehlermeldung sollte sich nach der Schwere des Fehlers richten, so dass z. B. Meldung bei schweren Fehlern nicht einfach weggeklickt werden können.

(Preim & Dachselt, 2010)

2.3.12 Adaptierbarkeit der Schnittstelle

Ein weiterer wichtiger Aspekt beim Mensch-Computer-Interaktionszentrierten Systementwurf ist die Anpassbarkeit bzw. die Konfigurierbarkeit der Benutzerschnittstelle an die Bedürfnisse

und Präferenzen des Anwenders. Individuelle flexible Einstellungsmöglichkeiten erleichtern die langfristige Nutzung der Software und leisten Beiträge zur Barrierefreiheit (s. Abschnitt 2.2).

Zum einen soll die Benutzerschnittstelle auf die unterschiedlichen Belange gelegentlicher und erfahrener Benutzer angepasst werden können. Durch folgende beispielhafte Anpassungen wird auch für heterogene Nutzergruppen die Nutzung über längere Zeiträume hinweg verbessert:

- Einfacher Zugriff auf Funktionen (z. B. über Menüs)
- Effizienter Zugriff z. B. über Tastaturkürzel oder Icons in einer sichtbaren Werkzeugleiste
- Schneller Zugriff auf häufige Aktionen
- Zusammenhängende Kommandos in Makros zusammenfassen (evtl. mit Unterstützung)

Zum anderen soll die Benutzerschnittstelle bzgl. der unterschiedlichen Wahrnehmungsfähigkeiten der Anwender angepasst werden können. Im Allgemeinen ist bei grafischen Benutzeroberflächen die Sehfähigkeit entscheidend. Die Anpassung von Schriftgrößen und die Auswahl von Farben ermöglicht z. B auch sehbehinderten und älteren Menschen die Nutzung einer Anwendung.

Des Weiteren soll eine Anwendung auch an die unterschiedlichen Umgebungen und Hardwarevoraussetzungen angepasst werden können. Die große Vielfalt von unterschiedlichen Endgeräten mit unterschiedlichen Bildschirmgrößen und -auflösungen, mit denen eine Anwendung genutzt werden kann, stellt besondere Anforderungen. Bedienelemente sollen unterdrückt oder initiiert und beliebig auf dem Bildschirm platziert werden können. Bereiche einer Anwendung sollen flexibel skalierbar sein. Unterschiedliche Hardwarevoraussetzungen erfordern gerade bei grafischen Elementen oft Kompromisse zwischen Darstellungsgeschwindigkeit und -qualität. Die Ausprägung dieser Kompromisse soll der Benutzer entsprechend seiner Hardware einstellen können. Ideal ist eine automatische Analyse der Hardware durch die Applikation und entsprechende Anpassungsvorschläge.

Sieht eine Anwendung unterschiedliche Nutzerrollen und entsprechend unterschiedliche Berechtigungen vor, muss die Ausprägung der GUI anpassbar sein. Bei eingeschränkter Funktionalität, z. B. eines normalen Anwenders gegenüber einem Administrator, sollte die Benutzerschnittstelle entsprechend anders aussehen. Auch der unterschiedliche Geschmack von Anwendern darf nicht außer Acht gelassen werden. Viele wollen die Oberfläche der Anwendung individuell gestalten und sollten im Sinne der „User Experience“ auch die Möglichkeit dazu haben. Anpassungen sollten in Nutzer-Profilen gespeichert werden können und nicht nur temporär verfügbar sein.

(Preim & Dachselt, 2010)

2.3.13 Adaptivität

Das System analysiert die Interaktion (z. B. Häufigkeit von wiederkehrenden Aktionen) des Benutzers und initiiert automatisch eine Anpassung nach gefundenen Mustern. Im Prinzip ist dadurch eine weitreichende Unterstützung des Anwenders möglich. Allerdings gestaltet sich dies in der Praxis problematisch, da in der Regel der Benutzer versucht das System zu erlernen und sich den Vorgaben des Systems anzupassen. Dabei kann eine Adaptivität des Systems für Verwirrung sorgen. Bewährt hat sie sich hingegen bei e-Commerce-Anwendungen wie eBay und Amazon.

(Preim & Dachselt, 2010)

Je nach Einsatzgebiet sollen bzw. können beim Design von Webseiten und -applikationen auch interkulturelle Belange berücksichtigt werden. Eine barrierefreie Gestaltung ist, wenn auch nicht zwingend vorgeschrieben, auch für nicht-behördliche Webauftritte vorzusehen, um einen gleichberechtigten Zugang zu Informationen für alle zu gewährleisten. Natürlich spielen bei der Entwicklung auch immer wirtschaftliche Aspekte eine Rolle, so dass es oft zu Kompromissen bei Entwicklungszeiten und -aufwänden kommt. Auch bestehen häufig Diskrepanzen zwischen Zielen und Anforderungen, da hier häufig unterschiedliche Stakeholder (z. B. spätere Benutzer / Nicht-Benutzer) aufeinander treffen. Kompromisse müssen allerdings nicht zwangsläufig zu einem schlechten System führen, es können auch vernünftige Lösungen diskutiert, gefunden und umgesetzt werden (Preim & Dachselt, 2010).

Die vorgestellten Prinzipien stellen grundlegende Regeln für den Entwurf ergonomischer und gebrauchstauglicher Anwendungen dar. Teilweise besteht in der konkreten Implementierung allerdings doch Bedarf nach konkreten, sehr feingranularen Anforderungen zu einem bestimmten Bereich / Prinzip. Diese Formulierung von Soll-Anforderungen wurde im großen Stil von Schmidtke & Jastrzebska-Fraczek (2013b) durchgeführt.

2.4 Anforderungen zur Usability

In dem Buch „Ergonomie. Daten zur Systemgestaltung und Begriffsbestimmungen“ definieren Schmidtke & Jastrzebska-Fraczek (2013b) in dem Kapitel „Software-Gestaltung“ ca. 300 Soll-Vorgaben, die aus einschlägigen Normen, Standards und Fachbüchern extrahiert wurden. Teilweise basieren die Vorgaben auch auf Expertenurteil. Die Daten und Begriffe dieses Buch sind Teil des EKIDES-Systems (Ergonomics Knowledge and Intelligent Design System) der Technischen Universität München. Es handelt sich dabei um ein Wissensmanagement System, das zum einen eine umfangreiche Datensammlung zu Arbeitsaufgaben, Umweltfaktoren, Betriebshandbüchern und Dienstvorschriften usw. enthält, zum anderen aber

auch Module für rechnergestützte Berechnungen, Prüfungen und Analysen anbietet. Die Gestaltungsprozesse von Mensch-Maschine-Systemen sollen durch das System möglichst weitgreifend unterstützt werden, um die Belastung des Menschen bei der Interaktion zu reduzieren (Schmidtke & Jastrzebska-Fraczek, 2013a).

Tabelle 2 zeigt als Beispiel die Soll-Vorgaben, die von Schmidtke und Jastrzebska-Fraczek zur Formblattdarstellung zusammengetragen werden.

Tabelle 2: Soll-Vorgaben zur Formblattdarstellung (Kapitel 2.4.2, Schmidtke & Jastrzebska-Fraczek, 2013b).

Nr.	Beschreibung	Soll-Vorgabe	Quelle
1	Formulardarstellung	für Formulare sollten spezifische Masken verfügbar sein	IfE
2	Formularsicherung	Formularbeschriftung kann nicht durch Fehleingabe zerstört werden	IfE
3	Datenfeldabgrenzung	Datenfelder sind so abzugrenzen, dass sich Daten von Titelzeilen deutlich unterscheiden	IfE
4	Datenfeldkennzeichnung	Datenfelder sind so zu kennzeichnen, dass kein Zweifel darüber entstehen kann, welcher Datentyp in das Feld gehört	DIN EN ISO 9241-12
5	Kennzeichnungsbegriffe	Datenfeldkennzeichnungen müssen den Feldinhalt beschreiben und mit denen übereinstimmen, die für den gleichen Datentyp in anderen Formblättern verwendet werden	DIN EN ISO 9241-17
6	Position der Datenfeldkennzeichnung	die Datenfeldkennzeichnung ist stets über oder links neben dem Datenfeld zu positionieren	DIN EN ISO 9241-12, DIN EN ISO 9241-17
7	Datendichte auf dem Bildschirm	es sollten nicht mehr als 40 % des möglichen Anzeigefeldes mit Daten ausgefüllt sein	DIN EN ISO 9241-17
8	Unterteilung langer Zeichenfolgen	Zeichenfolgen aus Buchstaben und Ziffern sollten in Gruppen von 3 bis 4 Zeichen aufgeteilt und durch eine Leerstelle oder einen Bindestrich voneinander getrennt werden	DIN EN ISO 9241-12

9	Leerzeichen und Null-stellen	über Tastatur eingegebene Leerzeichen müssen von Nullstellen (keine Eingabe) unterscheidbar angezeigt werden, wenn das die Informations-aufnahme fördert	IfE
10	Erkennung von Eingabe-fehlern	Plausibilitätsprüfung über Grenz-werte, -bereiche o. Ä. möglich	IfE

In dieser Feingranularität sind Soll-Vorgaben zur Software-Gestaltung für die Bereiche softwareergonomische Hinweise, Benutzerhandlungen und Benutzerführung, Textbearbeitung und Textdarstellung, Formblattbearbeitung und Formblattdarstellung, Tabellenbearbeitung und Tabellendarstellung und Bearbeiten und Darstellung von Grafiken vorhanden. Zusätzlich existiert ein weiteres Kapitel zum Thema „Web-Internetportal“, das ähnlich feingranular und umfangreich aufgebaut ist und relevante Anforderungen enthält.

2.5 Gestaltungsprinzipien

Ergänzend und teilweise komplementär spielen neben bzw. zur Umsetzung der Anforderungen und Grundlagen zur Ergonomie und der Gebrauchstauglichkeit die Gestaltgesetze eine Rolle, die in Abschnitt 2.3.5 bereits kurz angesprochen werden. Die Gestaltgesetze richten sich nach Erkenntnissen der Gestaltpsychologie und formulieren Regeln zur Anordnung und Form von Objekten (Thesmann, 2010).

- Gesetz der Ähnlichkeit

  Ähnlichkeiten z. B. bezüglich Farbe, Größe, Helligkeit, Bewegungsrichtung, -geschwindigkeit, Verhaltensweisen, Texturen und Geräuschen fördern das Erkennen von zusammengehörigen Elementen.

- Gesetz der Nähe

  Räumlich nah zusammenliegende Elemente werden eher als zusammengehörig wahrgenommen.

- Gesetz der Geschlossenheit

  Einfache und bekannte Formen unterstützen den kognitiven Hang des Menschen dazu, mehrere Objekte verstärkt als Einheit wahrzunehmen.

- Gesetz der Kontinuität

  Reize, die erkennbar eine Fortsetzung vorhergehender Reize sind, werden als strukturierte Gruppierung von Einzelelementen wahrgenommen (z. B. hierarchische Menüs).

- Gesetz des gemeinsamen Schicksals

  Je ähnlicher ein gleiches Verhalten (Bewegung, Rotation, Erscheinen, Starten, Stoppen, Pulsieren, usw.) von Elementen vorhanden ist, desto stärker ist deren Wahrnehmung als Gruppe.

- Gesetz der Erfahrung

  Bekannte Elemente, mit denen man bereits Erfahrung gesammelt hat, können in der Wahrnehmung automatisch vervollständigt werden.

- Gesetz der Prägnanz

  Wesentliche Elemente können durch die Gestaltung von Faktoren wie Größe, Farbe, Form und Hintergrundkontrast so hervorgehoben werden, dass sie sich deutlich vom Rest abheben.

- Gesetz der Symmetrie

  Symmetrisch angeordnete Elemente werden als eine Einheit wahrgenommen.

  (Thesmann, 2010)

2.6 Testen der Usability

Usability-Testen wird hauptsächlich mit menschlichen Probanden durchgeführt, die üblicherweise eine feste Aufgabenstellung bekommen und mit der zu testenden Software ein bestimmtes Problem lösen sollen. Als Probanden kommen sowohl normale potentielle Nutzer (Nutzertest) oder sehr erfahrene Experten auf dem zu testenden Gebiet (Expertentest) in Frage. (Rampl, 2007; Sarodnick & Brau, 2006)

Die Probanden müssen bewusst ausgewählt werden und die zukünftigen Nutzergruppen repräsentativ sein. Es gilt die Daumenregel, dass fünf bis sechs Probanden etwa 80 Prozent der Usability-Probleme aufdecken. (Sarodnick & Brau, 2006)

Üblicherweise wird die Lösung der Aufgabe per Video aufgezeichnet und/oder von einem Testleiter begleitet, der während des Tests die Tester dazu auffordert laut zu denken, um so Einsicht in deren Handlung- und Lösungsstrategien zu bekommen (Rampl, 2007; Sarodnick & Brau, 2006). Darüber hinaus werden häufig Fragebögen verwendet, die den Test begleiten und/oder eine abschließende Bewertung erfragen. Details zum Aufbau solcher Fragebögen liefern Sarodnick & Brau (2006).

Bei der Methode „Constructive Interaction“ (Sarodnick & Brau, 2006) wird die gestellte Aufgabe von zwei Probanden gelöst, die ihre Vorgehensweise laut diskutieren.

Alternativ zum lauten Denken gibt es noch die Methode des „Teaching back“ (Rampl, 2007), die darauf basiert, dass ein Nutzer, der sich bereits mit der Software vertraut gemacht hat, einem anderen Nutzer erläutert, wie die Software funktioniert.

Diese Nutzer- und Expertentests können durch technische Maßnahmen unterstütz werden, wie z. B. die Eingabeprotokollierung während der Aufgabenlösung. Dabei werden z. B. die Mausklicks registriert oder gemessen, wie lange der Nutzer auf einer bestimmten Ansicht oder einer Webseite verweilt. (Sarodnick & Brau, 2006)

Technisch aufwändigere Verfahren werden durch Nielsen & Pernice (2010) beschrieben. Dabei wird aufgezeichnet, wo die Nutzer auf einer Webseite oder einem Anwendungsfenster hinschauen („Eyetracking“).

Wenn es um die Analyse von Usability auf Webseiten geht, kommen einfache Verfahren, wie die Analyse der Webserver-Logfiles in Frage, die Auskunft über oft besuchte Unterseiten und Ein- und Ausstiegspunkte in Webseiten liefern (Rampl, 2007).

Grundsätzlich wird in induktive und deduktive Tests unterschieden. In induktiven Tests werden Prototypen und Vorabversionen bezüglich Schwachstellen und Verbesserungen für die Gestaltung untersucht. In deduktiven Tests werden verschiedene Varianten eines Prototyps verglichen. Für das GUI-Prototyp-Usability-Testen sind also induktive Tests relevant. (Sarodnick & Brau, 2006)

3 Grundlagen Prototyping

Innerhalb der Softwareentwicklung ist es schwierig dem Begriff „Prototyping“ eine einheitliche Definition zuzuordnen. Pomberger & Blaschek (1996) formulieren die folgende Definition als Grundlage für eine weiterführende, differenziertere Betrachtung:

> „Ein *Software-Prototyp* ist ein – mit wesentlich geringerem Aufwand als das geplante Produkt hergestelltes – einfach zu änderndes und zu erweiterndes ausführbares Modell des geplanten Software-Produkts, das nicht notwendigerweise alle Eigenschaften des Zielsystems aufweisen muß, jedoch so geartet ist, daß vor der eigentlichen Systemimplementierung der Anwender die wesentlichen Systemeigenschaften erproben kann. *Prototyping* umfaßt alle Arbeiten, die zur Herstellung solcher Prototypen notwendig sind.“ (Pomberger & Blaschek, 1996)

Des Weiteren muss ein Prototyp in jedem Fall ausführbar sein und muss schnell und billig realisiert werden können. Je nach Verwendungszweck, Einsatzbereich und Ziel gibt es unterschiedliche Arten des Prototyping und von Prototypen.

Prototyping muss nicht Teil des Softwareentwicklungsprozesses sein, kann aber sehr effektiv in diesen integriert werden. Vor allem in iterativen Modellen der Softwareentwicklung bietet sich ein prototyporientiertes Verfahren oft an. Abbildung 1 zeigt, wie sich Prototyping z. B. in die klassischen Phasen des Software-Life-Cycle-Modells (Pomberger & Blaschek, 1996) integriert.

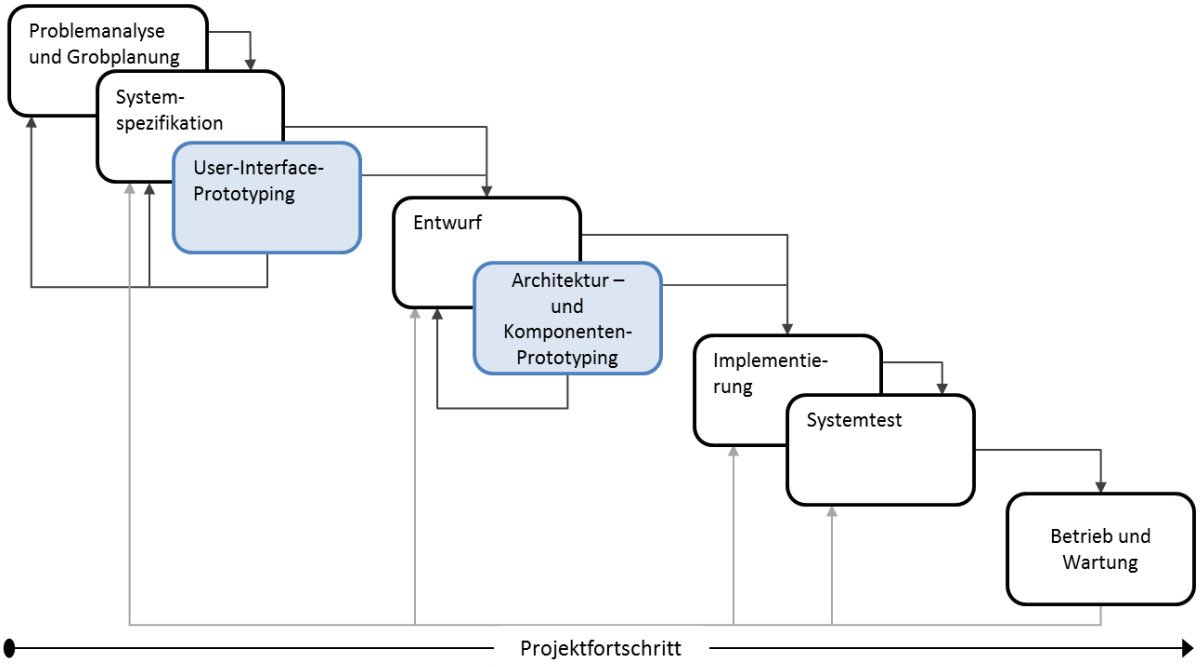


Abbildung 1: Prototyping in der Softwareentwicklung (nach Pomberger & Blaschek, 1996, S. 25).

Ebenso kann es aber auch seinen Platz in einem Spiral-Modell (Balzert, 2008) oder anderen Vorgehensmodellen zur Softwareentwicklung haben. Beim Prototyping selbst handelt es sich im Idealfall um einen inkrementellen Prozess innerhalb der Softwareentwicklung (s. Abbildung 2).

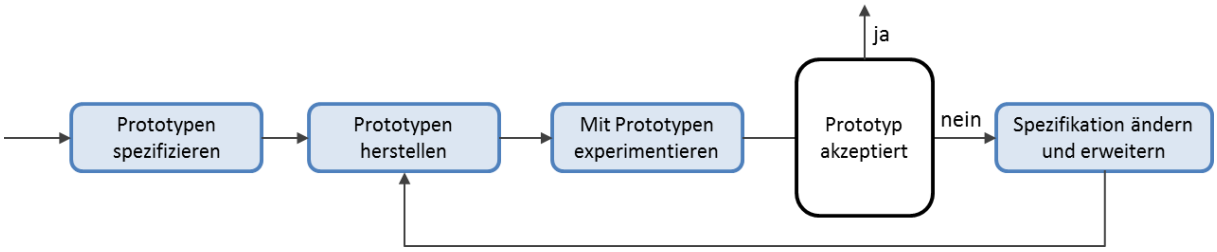


Abbildung 2: Prototyping als inkrementeller Prozess (nach Pomberger & Blaschek, 1996, S. 25).

Dieser Prozess zeigt bereits deutlich die Anwendungsbereiche von Software-Prototypen auf. In Form eines Prototypen überschneiden sich die Entwickler- und die Anwendersicht. Ein Prototyp kann daher als solide Kommunikationsbasis für die Qualitätssicherung und Spezifizierung von Anforderungen und die Lösung von Entwicklungsproblemen herangezogen werden. Nicht nur für funktionale Anforderungen an die Software, sondern auch für nichtfunktionale Anforderungen im Bereich der Software-Ergonomie kann ein Prototyp zur Spezifizierung und Qualitätssicherung eingesetzt werden. Ein Prototyp trägt damit wesentlich zur Risikominimierung bei und ist ein wichtiger Bestandteil der Softwareentwicklung.

Andererseits dürfen Prototypen kein Ersatz für eine konkrete und vollständige Anforderungsanalyse, -erhebung und -dokumentation sein. Sie sollten nur ergänzend und unterstützend herangezogen werden und können dann die späteren Entwicklungsarbeiten wesentlich erleichtern und präzisieren. Durch die Implementierung von Prototypen kann es natürlich auch zu zusätzlichen Kosten und Entwicklungszeiten kommen (da Prototypen evtl. vollständig oder teilweise nicht weiterverwendet werden können), die sich allerdings durch eine entsprechende Fehlerminimierung bei der Implementierung bereits im Vorfeld relativieren.

3.1 Arten des Prototyping

Es gibt je nach Verwendungszweck und Ziel verschiedene Arten von Prototyping und Prototypen.

Exploratives Prototyping

Beim explorativen Prototyping wird ein einem ersten Schritt ein Prototyp mit Grundfunktionalitäten entwickelt, basierend auf ersten Vorstellungen über das spätere System. Hierbei arbeiten Anwender und Entwickler eng zusammen, wobei die Entwickler dabei einen Einblick in den Anwendungsbereich des Systems erhalten.

Das Augenmerk liegt bei dem entstehenden Prototyp nicht auf der Qualität, sondern auf der Funktionalität, der leichten Anpassbar- und Veränderbarkeit und der Kürze der Entwicklungszeit des Prototyps.

Der Prototyp dient als Basis für die Diskussion über Lösungsansätze und Realisierbarkeiten der umgesetzten Vorstellungen. Er wird zur Unterstützung bei der Entwicklung der Systemspezifikation herangezogen. Als Ergebnis liegt eine vollständige und übersichtliche Systemspezifikation vor.

(Pomberger & Blaschek, 1996)

Experimentelles Prototyping

Um die Tauglichkeit von Teilspezifikationen, von Architekturmodellen und von Lösungsideen für einzelne Systemkomponenten zu klären, kann experimentelles Prototyping eingesetzt werden. Als Basis zur Erstellung des Prototyps dienen Vorstellungen zur Zerlegung des Systems. Wechselwirkungen zwischen den Systemkomponenten, das Zusammenwirken von Schnittstellen der einzelnen Systemkomponenten und die Flexibilität der Systemzerlegung hinsichtlich der Erweiterbarkeit können anhand von Anwendungsbeispielen erprobt werden.

Verantwortlich für das Prototyping sind hier vor allem die Entwickler, die den Prototypen als Technik zur Unterstützung beim System- und Komponentendesign nutzen und anhand dessen den experimentellen Nachweis der Tauglichkeit erbringen. Auch hier steht die Funktionalität, nicht die Qualität, des Prototyps im Vordergrund. Das Ziel ist eine vollständige Spezifikation von Teilsystemen als Grundlage für die Implementierung des Systems.

(Pomberger & Blaschek, 1996)

Evolutionäres Prototyping

Bei dieser Art des Prototyping verschwimmt die Linie zwischen Prototyp und Produkt. Für die bereits geklärten Nutzeranforderungen wird ein Prototyp entwickelt, der nach und nach in iterativen Schritten weiterentwickelt wird. Auf Basis der jeweiligen Ergebnissprototypen werden mit den Anwendern weitere Anforderungen erarbeitet und in den nächsten, erweiterten Prototypen integriert, bis hin zum fertigen Produkt. Zum einen kann ein solcher Prototyp den Prozess der Systemspezifikation teilweise erleichtern, andererseits kann er auch dazu führen, dass Anforderungen evtl. unvollständig erfasst werden, insbesondere was das Zusammenspiel unterschiedlicher Komponenten und Systemteile betrifft. In der Regel werden hier keine Wegwerfprototypen entwickelt, sondern der Prototyp wird zum fertigen Produkt ausgebaut. (Pomberger & Blaschek, 1996)

Vertikales Prototyping

Alle Ebenen eines Systems werden für einen ausgewählten konkreten Bereich der Software implementiert. Dieser Teil des Systems ist dann mit allen spezifizierten Funktionen und dem

entsprechenden gewünschten Verhalten vorhanden und kann getestet werden. Diese Teil-Prototypen können bereits erstellt werden, während die Spezifikation für andere Teile des Systems noch nicht abgeschlossen ist. Diese Art des Prototyping eignet sich besonders für die Klärung von offenen Funktionalitäts- und Implementierungsfragen. („Prototyping (Softwareentwicklung)“, 2013)

Horizontales Prototyping

Im Gegensatz zum vertikalen Prototyping wird nur eine ausgewählte spezifische Ebene des Gesamtsystems erstellt, diese jedoch möglichst vollständig. Ein klassisches Beispiel ist der Oberflächenprototyp (s. u.). („Prototyping (Softwareentwicklung)“, 2013)

Oberflächenprototyp

Beim Oberflächenprototyp wird die GUI, also die Nutzeroberfläche eines Systems, erstellt, weitestgehend ohne die darunterliegenden Funktionalitäten. („Prototyping (Softwareentwicklung)“, 2013)

Vollständiger Prototyp

Bei einem vollständigen Prototyp sind alle wesentlichen Funktionen des geplanten Systems vollständig verfügbar. Die bei der Erstellung des Prototypen gewonnen Erkenntnisse bilden die Basis für die endgültige Systemspezifikation. Für Softwaresysteme werden kaum vollständige Prototypen hergestellt. (Pomberger & Blaschek, 1996)

Unvollständiger Prototyp

Es werden einzelne Aspekte des Gesamtsystems, wie z. B. einzelne Benutzerschnittstellen und Systemkomponenten umgesetzt, um deren Systemtauglichkeit und Machbarkeit zu testen. (Pomberger & Blaschek, 1996)

Wegwerfprototyp

Dieser Prototyp wird nicht für die Implementierung des endgültigen Systems verwendet sondern läuft im Laufe der Entwicklung der Software ab. (Pomberger & Blaschek, 1996)

Wiederverwendbarer Prototyp

Im Gegensatz zum Wegwerfprototyp werden wesentliche Teile des oder der im Laufe des Entwicklungsprozesses erstellten Prototypen für die endgültige Implementierung des Systems genutzt. (Pomberger & Blaschek, 1996)

Mock-up

Während es sich bei einem Prototypen um eine mehr oder weniger umfangreiche funktionsfähige Abbildung des Gesamtsystems handelt, stellt ein Mock-up nur eine rudimentär-

funktionale Simulation bzw. Nachbildung der Benutzerschnittfläche eines zu erstellenden Softwaresystems dar. Es erfolgt keine eigentliche Implementierung, sondern eine Visualisierung der Oberfläche mit Hilfe von Grafikprogrammen. Ein Mock-up wird oft sehr früh in der Entwicklungsphase erstellt und erleichtert die Anforderungserhebung und -spezifizierung. („Mock-up“, 2013)

3.2 Werkzeuge für GUI-Prototyping

Da es sich bei GUI-Prototypen oft um kurzlebige Wegwerfprototypen handelt, ist eine schnelle und billige Erstellung essentiell. Gleichzeitig sollen sie allerdings auch ihren Zweck erfüllen und das Aussehen und / oder Verhalten des Softwareprodukts möglichst zielführend visualisieren. Daher kommen oft Werkzeuge bei der Erstellung von Prototypen zum Einsatz.

Werkzeuge für Mock-ups

Bei Mock-ups handelt es sich meist um funktionslose „Bilder“ der Benutzerschnittstelle ohne Interaktionsmöglichkeiten. Diese können in der Regel mit einem beliebigen Grafikprogramm, wie z. B. gimp[1], Microsoft Paint[2] oder Adobe Photoshop erstellt werden. Es gibt allerdings auch auf Mock-ups spezialisierte Werkzeuge, die dem Nutzer das Erstellen gezielt erleichtern, indem sie z. B. bereits Formen- und Icon-Bibliotheken für verschiedene Plattformen (z. B. Android, Windows, iOS) oder ein Oberflächendesign über Drag and Drop bereitstellen. Ein solches Werkzeug ist Evolus Pencil (s. Abbildung 3). Es handelt sich hierbei um freie und Open Source Software die unter der GPL veröffentlicht ist.

[1] GNU Image Manipulation Program (http://www.gimp.org/)

[2] Grafiksoftware, integriert in das Betriebssystem Microsoft Windows

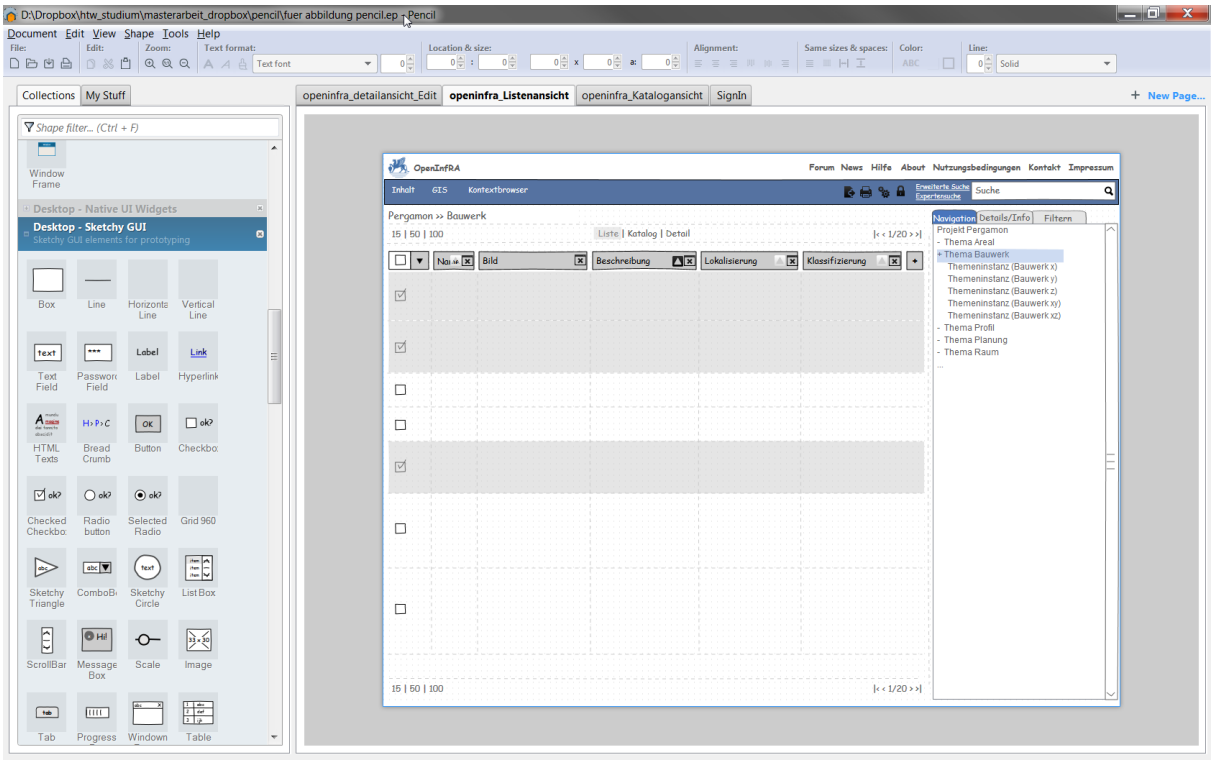


Abbildung 3: Evolus Pencil Version 2.0.5.

Evolus Pencil ist aktuell in der stabilen Version 2.0.5 verfügbar und ist für Windows, Mac OSX und Linux erhältlich. Die Oberflächen Mock-ups können über einfaches Drag and Drop aus zahlreichen vorinstalliert zur Verfügung gestellten Widgets, Icons und Formen zusammengestellt werden. Elemente können auf einzelne Oberflächen-„Seiten“ in Pencil verlinkt werden, so dass eine rudimentäre Darstellung von GUI-Arbeitsabläufen simuliert werden kann. Prototypen können als Bilder (png) oder Dokumente (html, pdf, doc) exportiert werden.

Werkzeuge für interaktive GUI-Prototypen

Auch für die Erstellung interaktiver GUI-Prototypen gibt es zahlreiche Werkzeuge, die mit unterschiedlichen Techniken arbeiten. Ein weit verbreitetes Werkzeug ist z. B. Flash von Adobe. Aber auch andere, preiswertere Programme wie z. B. AppSketcher (s. Abbildung 4) bieten die Möglichkeit Aktionen in Prototypen zu integrieren.

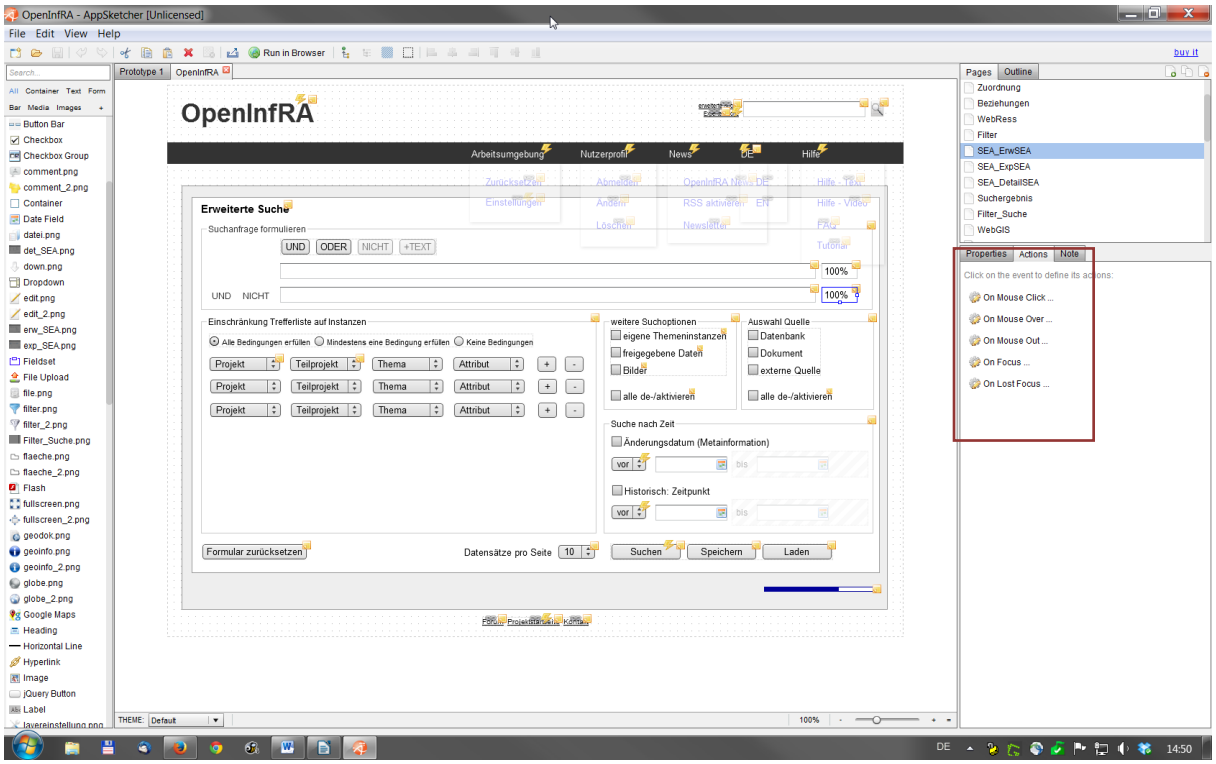


Abbildung 4: AppSketcher Version 1.5.6 mit der Möglichkeit interaktiver Aktionen in Prototypen (s. rotes Rechteck).

AppSketcher erlaubt den Nutzer ebenso wie Evolus Pencil ein einfaches Erstellen von Oberflächen via Drag and Drop von fertigen Layout-Komponenten nach dem What You See Is What You Get (WYSIWYG) Prinzip auf die Arbeitsfläche. Wireframe-Komponenten können nach bestimmten Themen ausgewählt oder frei kombiniert werden. Die Anwendung unterstützt die gebräuchlichsten CSS-Stile und erlaubt interaktive Aktionen und Links. Das Ergebnis sind interaktive Prototypen in HTML, CSS und / oder JavaScript, die in beliebigen Browsern gezeigt bzw. gehostet werden können.

Darüber hinaus können Prototypen auch direkt mit den Technologien implementiert werden, die später eventuell für die Systemimplementierung verwendet werden.

3.3 Technologien für die GUI-Prototyp-Implementierung

Für die Entwicklung von Webseiten und insbesondere Webanwendungen kommt man um die Kombination der drei Technologien HTML (zurzeit in der Version 5), CSS (zurzeit in der Version 3) und JS (zurzeit in der Version 1.8.5) nicht herum (Flanagan, 2011; Freeman, 2011; Münz & Gull, 2013; West, 2013). Sie werden im Folgenden vorgestellt. Darüber hinaus werden die wichtigsten Eigenschaften des für die Entwicklung des GUI-Prototyps verwendete JS-Frameworks Ext JS (zurzeit in der Version 4) erläutert.

3.3.1 Hypertext Markup Language

HTML ist eine Sprache zur Erstellung von Webseiten, mit der sich ein strukturierter Inhalt erstellen lässt und mit der sich Links zu anderen Seiten, aber auch multimediale Inhalte einbinden lassen (Münz & Gull, 2013).

HTML5 ist die neueste Version, die sich zurzeit noch in der Entwicklung befindet. Im Vergleich zu HTML4 ist sie syntaktisch strukturierter und erleichtert so das Einbinden von programmierten Funktionalitäten und neuer Elemente aus der Praxis (z. B. multimediale Inhalte oder die Geolokation des Nutzers). Darüber hinaus ist die Möglichkeit, Seiten nach einer semantischen Struktur aufzubauen verbessert worden. (Münz & Gull, 2013)

Darunter fallen z. B. die neuen Seitenelemente *header*, *footer*, *nav*, *aside*, *section* und *article*, die es einfach ermöglichen, neue Seitenelemente zu erstellen (Schmitt & Simpson, 2011). Mit einer Kombination dieser Elemente kann man einfache Seitenlayouts erstellen, wie z. B. in Abbildung 5 für einen Blog dargestellt.

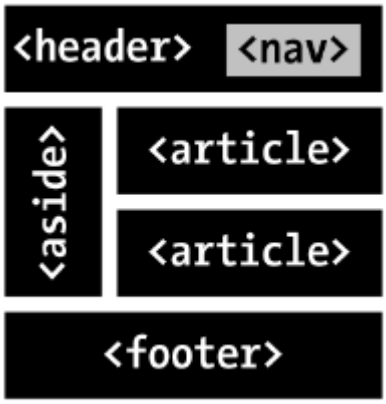


Abbildung 5: Beispiel einer einfachen Blogstruktur unter Verwendung neuer HTML5-Elementen (Schmitt & Simpson, 2011, S. 6).

Für Freeman (2011) sind die wichtigsten Neuerungen in HTML5:

- Native Unterstützung von multimedialen Inhalten (lt. Lubbers (2011) liegt der Augenmerk hier darauf, dass für viele Inhalte keine Plug-Ins mehr verwendet werden müssen)
- Bessere Unterstützung der Integration externen Quellcodes (z. B. JS im *canvas*-Element)
- Unterstützung der Vision des Semantic Web (durch die oben beschriebenen neuen HTML-Elemente, die es erlauben, eine Webseite besser in einen Kontext einzubinden)

Lubbers (2011) fügt noch hinzu, dass HTML5 ein großer Schritt weiter in die Richtung ist, Präsentation und Inhalt von Webseiten weiter zu separieren (z. B. über eine bessere Anbin-

dung von CSS). Detaillierte Listen über alle neuen Elemente (u. a. Geolocation, MathML und Application Cache) finden sich auch bei Lubbers (2011) und zahlreichen anderen Büchern.

Verschiedene Autoren loben die Neuerungen in HTML5. So z B. Matt West, der HTML5 wie folgt charakterisiert: HTML5 „enable[s] developers to create websites that are smarter, faster, and more secure then they have ever been before.“ (West, 2013, S. 1)

Neben viel Lob werden die Neuentwicklungen auch kritisch gesehen, da der „Standard“ in zwei Richtungen zerfällt, die einerseits von der Web Hypertext Application Technology Working Group (WHATWG) und vom World Wide Web Consortium (W3C) entwickelt werden. Laut Münz & Gull (2013) ist, was heute als HTML5 „bekannt“ ist, meist nicht mehr vom W3C entwickelt. Darüber hinaus trägt die Entwicklung der WHATWG keine Versionsnummer mehr, sondern repräsentiert einen lebenden Standard, der ständig weiterentwickelt wird (Schmitt & Simpson, 2011). Insgesamt wird die Zukunft von HTML5 oft als unsicher gewertet (z. B. Münz & Gull, 2013).

Genau wie mit HTML4 unterstützen aktuelle Browser HTML5 mehr oder weniger gut (Schmitt & Simpson, 2011).

3.3.2 Cascading Stylesheets

CSS sind seit HTML4 fest in HTML integriert (Münz & Gull, 2013) und sind das Werkzeug zur Spezifikation der Präsentation (Aussehen und Formatierung) für HTML-Dokumente (Freeman, 2011).

Zwei wichtige Aspekte, die eine universelle Nutzung von CSS möglich macht, sind die Möglichkeiten des Imports von CSS-Dateien in andere CSS-Dateien und die Möglichkeit Eigenschaften zu kaskadieren und zu vererben (Freeman, 2011).

Meyer ergänzt und erweitert die Liste der Eigenschaften um (Meyer, 2007):

- „Rich Styling“

  Die Designmöglichkeiten sind reichhaltiger als in HTML, so können u. a. Farben, Schriftarten, Begrenzungen, Hintergründe, Abstände und Puffer komfortabel konfiguriert werden.

- „Ease of Use“

  Ein bestimmter Style wird nur einmal definiert und kann dann wiederverwendet werden. Im Gegensatz dazu muss in HMTL jeder Tag einzeln gestaltet werden, was zu einem massiven Mehraufwand und einer Vergrößerung der zu transportierenden Dateien geführt hat.

- „Using Your Styles on Multiple Pages“

CSS-Dateien sind wiederverwendbar, z. B. um alle Webseiten einer Organisation gleich zu gestalten. Wenn sich Änderungen ergeben, muss nur eine einzige CSS-Datei modifiziert werden.

- „Cascading“

  Vererbung ermöglicht z. B. die Weitergabe einer Schriftfarbe von Überschriften auf Unterüberschriften). Zusätzlich können sie in den Unterklassen wieder beliebig überschrieben werden. Auf der Nutzerseite (z. B. im eigenen Webbrowser) können *user styles* gesetzt werden, die die von der Webseite zur Verfügung gestellten Stile überschreiben, z. B. die grundsätzliche Vergrößerung der Schrift für Sehbehinderte.

- „Compact File Size“

  Durch das kompakte einmalige Definieren des Designs wird viel unnötiger Datenverkehr vermieden und dadurch die Downloadzeit verkürzt.

- „Preparing for the Future“

  CSS unterstützt die Trennung von Inhalt und Aussehen und macht damit vieles zukunftsfähiger (z. B. die Weiterentwicklung von HTML).

Wie auch für HTML gilt, dass die Interpretation von CSS der vorhandenen Webbrowser sehr unterschiedlich ist und somit grundsätzlich die plattformunabhängige Entwicklung erschwert (Münz & Gull, 2013).

3.3.3 Java Script

Im Gegensatz zu serverseitigem Scripting z. B. über Java Server Pages (JSP), ermöglicht JavaScript clientseitiges Scripting, das im lokalen Webbrowser interpretiert und ausgeführt wird (Münz & Gull, 2013). Auf komplexe Webserverentwicklungen kann so verzichtet werden.

Flanagan bezeichnet JavaScript als „Programming language of the Web“ (Flanagan, 2011, Kapitel 1), und als „high-level, dynamic, untyped interpreted programming language that is well-suited to object-oriented and functional programming styles“ (Flanagan, 2011, Kapitel 1).

JavaScript definiert ein minimales Application Programming Interface (API), das im Webbrowser je nach Produkt unterschiedlich interpretiert wird. Darüber hinaus werden Entwickler durch Debugger unterstützt. (Flanagan, 2011)

Um die unterschiedliche JavaScript-Unterstützung der verschiedenen Browser besser überwinden zu können, nutzen viele Entwickler JS-Frameworks (Münz & Gull, 2013). Dazu zählt z. B. das für den Prototyp genutzte Framework Ext JS 4 (beschrieben in Abschnitt 3.3.4), aber auch das Dojo Toolkit (http://dojotoolkit.org/) und Bootstrap (http://getbootstrap.com/)

bzw. dessen vom DAI entwickelte Erweiterung Archeostrap (http://crazyhorse.archaeologie.uni-koeln.de/~fabian/archaeostrap/base-css.html).

Wikipedia gibt eine Tabelle mit einer Übersicht über viele JS-Frameworks und deren Eigenschaften („Comparison of JavaScript frameworks“, 2013).

3.3.4 Ext JS 4

Das JS-Framework Ext JS 4 wird für die Entwicklung des GUI-Prototyps verwendet. Eine Begründung für diese Entscheidung findet sich in Abschnitt 6.1).

Ext JS 4 ist ein plattformübergreifendes Framework für die Entwicklung von Webanwendungen. Basierend auf Web-Standards wird eine umfangreiche Bibliothek zur GUI-Entwicklung zur Verfügung gestellt, inklusive der dazugehörigen Datenverwaltung. Im Unterschied zu Ext JS 3 sind mehrere GUI-Elemente und verschiedene einfach anpassbare Designs eingeführt worden. Darüber hinaus ist eine Unterstützung des Model-View-Controller-Konzepts (MVC, Krasner & Pope, 1988) für die Entwicklung zur Verfügung gestellt worden. (Ashworth & Duncan, 2012)

Zusätzlich wird Objektorientierte Programmierung (OOP) durch die Verwendung eines Klassensystems inklusive Vererbung ermöglicht (Ashworth & Duncan, 2012).

Da die Entwicklung des GUI-Prototyps das MVC-Konzept nutzt, sei die Umsetzung in Ext JS hier kurz vorgestellt (Ashworth & Duncan, 2012):

- Ein *Model* enthält die anzuzeigenden Daten und deren Attribute. Ein Model kann sich über das *Data*-Package persistieren und gleichzeitig mit anderen Models assoziieren.
- Die Datenansicht wird über *View*-Komponenten ermöglicht. Dazu zählen alle GUI-Elemente (u. a. Tabellen, Bäume und Interaktionselemente).
- *Controller* erledigen die Kommunikation, Aktualisierung und Synchronisation zwischen Model und View und enthalten viel Integrationslogik.
- Models werden normalerweise mit entsprechenden *data stores* gekoppelt, die den Schreib- und Lesezugriff auf die Daten ermöglichen.

Weitere Details zum MVC-Konzept und deren Umsetzung im GUI-Prototyp finden sich in Abschnitt 6.2.

4 OpenInfRA

Der gesamte Inhalt des Kapitel 3 basiert auf einer Zusammenfassung des Grobkonzepts von OpenInfRA in der Version 2.2 (OpenInfRA, 2013)(Anhang C).

OpenInfRA (Open Information System for Research in Archaeology) ist ein webbasiertes Informationssystem zur Dokumentation und Publikation archäologischer Forschungsprojekte, das in Zusammenarbeit der HTW Dresden (Hochschule für Technik und Wirtschaft Dresden), der BTU Cottbus (Brandenburgische Technische Universität Cottbus – Senftenberg) und dem DAI Berlin (Deutsches Archäologisches Institut) entwickelt wird. Das Projekt wird seit Juni 2011 durch die DFG (Deutsche Forschungsgemeinschaft) gefördert und befindet sich derzeit in der zweiten (finalen) Projektphase, in der, auf Basis des in der ersten Projektphase erstellten Grobkonzepts (OpenInfRA, 2013) die Implementierung des Systems erfolgen wird. Die Anwendung soll folgende Grundfunktionen aufweisen:

- „Webbasierte Plattform für Forschungs(roh)daten
- Ausgrabungen, Survey, Bauforschung, Bohrkerne, Geophysik, Vermessung, Restaurierung, Fundanalyse, etc.
- einheitliche, modulare Datenstruktur
- differenzierte Benutzer- und Rechte-Verwaltung
- Unterstützung von Mehrsprachigkeit
- Veröffentlichung von Primärdaten und Forschungsergebnissen
- standardisierte und dokumentierte Techniken / OpenSource
- Offline-Version und zuverlässige Datensynchronisation
- webbasierte 2D- und 3D-GIS-Funktionalitäten
- 3D-Bauwerksinformationssystem und räumliche Anfragesprache
- div. Datei- und Online-Schnittstellen
- div. Flexible Such- und Abfragefunktionalitäten
- Individualisierbarer Nutzerclient für effiziente Datenerfassung
- Umfangreicher Admin-Client für einfache Anpassung an heterogene Projekte“

(OpenInfRA, 2013)

Im Sinne einer langfristigen Interoperabilität der Anwendung liegt ein Entwicklungsschwerpunkt auf der Implementierung von weitgehend standardkonformen Daten- und Dienste-schnittstellen sowohl zu DAI-internen Systemen, als auch auf im weitesten Sinne „externe“

Ressourcen. Ebenso gibt es eine Schnittstelle, die den Zugriff von externen Systemen auf freigegebene OpenInfRA-Inhalte erlaubt.

Das System soll in laufenden Projekten als zentrales Dokumentationswerkzeug und Archiv eingesetzt werden. Zusätzlich soll es die wissenschaftliche Auswertung und Analyse der gesammelten Daten durch umfangreiche Such- und Recherchefunktionen (auch angebundener externer Datenbestände) unterstützen. Gleichzeitig soll es auch zur Publikation und Diskussion digitaler, freigegebener Forschungsdaten genutzt werden können.

4.1 OpenInfRA-Systemspezifikation

Die Spezifizierung der OpenInfRA-Anwendung orientiert sich an den Richtlinien für E-Government-Anwendungen (Bundesministerium des Inneren, 2008) und entsprechend an den Konzepten des RM-ODP (Reference Model of Open Distributed Processing)(ISO/IEC, 1998). Dies spiegelt sich in der inhaltlichen Struktur des vorliegenden OpenInfRA Grobkonzepts wieder. Schwerpunkte bei der Systemspezifikation liegen auf der Anforderungserhebung und -analyse und auf dem Entwurf eines generischen Informationsmodells. Das Grobkonzept gliedert sich im Wesentlichen in die vier Sichten des RM-ODP auf das System:

- **Fachliche Sicht / Enterprise Viewpoint**

  Auf Basis der Analyse der Altsysteme iDAI.field und CISAR werden grundlegende Projektziele und Rahmenbedingungen definiert. Aus den Anwendungsfällen des Systems und der Analyse der Altsysteme werden funktionale und nicht-funktionale Anforderungen abgeleitet, dokumentiert (s. Anhang B des Grobkonzepts) und erläutert.

- **Sicht auf das Informationsmodell / Information Viewpoint**

  Aufgrund des weiten Diskursbereichs wird für die OpenInfRA-Anwendung wird ein konzeptuelles Anwendungsschema als Metamodell entwickelt. In diesem Zusammenhang wird die Umsetzung zentraler Projektthemen wie Mehrsprachigkeit, Metadaten und projektspezifische Anpassungsfähigkeit des Systems definiert. Es gibt ein Rollen-Rechte-System, das sowohl den internen Zugriff als auch den Zugriff über externe Systeme auf die OpenInfRA-Ressourcen regelt.

- **Funktionale Sicht / Computational Viewpoint**

  Das System entspricht einer klassischen Drei-Schichten-Architektur. Diese umfasst eine Präsentationsschicht mit einer grafischen Benutzerschnittstelle und Datei- und Online-Schnittstellen. Die Anwendungsschicht umfasst vor allem die Logik zur Datenerfassung und -ausgabe und zur Projekt- und System-Administration. Die Speicherung der Daten erfolgt in einer Datenbank, in der Sach- und Geometriedaten zusammengehalten werden.

- **Technologische Sicht / Technology Viewpoint**

  In diesem Kapitel werden technische Details zu Aufbau und Funktionsweise des objektrelationalen Geodatenbanksystems für OpenInfRA beschrieben. Aktuell in Bearbeitung sind Kapitel zur Umsetzung eines XML-Schemas und dessen bidirektionaler Abbildung mit der Datenbank. Des Weiteren wird die technische Umsetzung des sich aktuell in Entwicklung befindlichen 3D-WebGIS-Clients beschrieben.

Das OpenInfRA-Grobkonzept V2.2 umfasst folgende Anhänge:

- **Anhang A – Anwendungsfälle**

  Dieser Anhang enthält 21 grobgranulare Anwendungsfälle zur Administration, zur Benutzung und generelle Systemanwendungsfälle.

- **Anhang B – Anforderungskatalog**

  Der Anforderungskatalog umfasst ca. 700 Anforderungen zu den Bereichen WebGIS, Benutzung, Administration, GeoDokumentService, Konzepte, System, Schnittstellen, Sicherheit, Performanz, Suche, Usability und Zugänglichkeit (Barrierefreiheit).

  Eine projektinterne Priorisierung legt die Wichtigkeit der Umsetzung der einzelnen Anforderungen an das System fest und wird im Ausschreibungsverfahren bei der Bewertung der Bewerber genutzt werden

- **Anhang C – Wertelisten**

  Es handelt sich um Wertelisten mit vorgegebenem vereinheitlichtem Vokabular. Diese sollen im Zuge der Implementierung noch von Fachanwendern vervollständigt werden.

- **Anhang D – Rollen-Rechte-Matrix**

  Der kontrollierte Zugriff auf Systemkomponenten und -daten wird in OpenInfRA über ein Rollen-Rechte-System geregelt. Die Rollen-Rechte-Matrix gibt einen Überblick über die Objekte, auf die Rechte vergeben werden können müssen und welche Operationen auf diese Objekte kontrolliert werden können müssen.

- **Anhang E – Ausprägungen WebGIS-Client**

  Um eine einfache Bedienung des WebGIS-Clients für Laien und Fachnutzer gleichermaßen zu gewährleisten, werden in diesem Anhang anhand der funktionalen Anforderungen beispielhaft eine Standard- und eine Expertenausprägung für den WebGIS-Client definiert.

- **Anhang F – Oberflächenprototypen**

  Es handelt sich bei diesem Anhang um eine Sammlung von Screenshots der entwickelten Oberflächenprototypen (s. Abschnitt 4.2).

- **Anhang G – Datenbank-Integritätsbedingungen**

  Die für die Datenbank definierten Integritätsbedingungen werden hier detailliert aufgeführt.

Das Grobkonzept dient im Rahmen einer Ausschreibung als Hauptbestandteil eines Lastenhefts, sowie als Grundlage und Leitfaden für die Eigenentwicklungen, die im Rahmen des Projekts geplant sind. Wie auch die Altsysteme soll OpenInfRA letztendlich über die IT-Infrastrukturen des DAI Fachnutzern für archäologische und bauforscherische Projekte zur Verfügung gestellt werden. Zusätzlich soll die OpenInfRA-Software auch als Open Source-Projekt frei zugänglich und nutzbar sein.

4.2 Altsysteme CISAR und iDAI.field

Zur Anforderungserhebung und -analyse (s. Abschnitt 4.1) werden auch die bestehenden Altsysteme CISAR und iDAI.field herangezogen, die nachfolgend kurz vorgestellt werden.

CISAR

CISAR (Cottbuser Informationssystem für Archäologie und Bauforschung) wurde in einer der Kooperation der Forschungsprojekte Stadtforschung Baalbek und der Untersuchung der Domus Severiana auf dem Palatin in Rom entwickelt. Es handelt sich um ein „projektübergreifend einsetzbares, modular aufgebautes und webbasiertes Datenbanksystem für Archäologie und Bauforschung“ (BTU Cottbus - Senftenberg, 2013). Die Lösung basiert auf standardisierten Web-Technologien, ist plattformunabhängig und garantiert eine hohe Datensicherheit und -verfügbarkeit. Zusätzlich beinhaltet CISAR ein komplexes Geoinformationssystem und ein 3D-Bauwerwksinformationssystem für die Analyse komplexer, dreidimensionaler Bauwerkstrukturen. Die Anwendung wurde ausschließlich mit Open Source Software realisiert (DAI, 2013a).

iDAI.field

Bei iDAI.field (DAI, 2013b) handelt es sich um ein modulares Dokumentationssystem für Feldforschungsprojekte, das vom IT-Referat des DAI Berlin 2009 entwickelt wurde. Es wird derzeit in mehreren DAI-Projekten mit unterschiedlichen Schwerpunkten und Fragestellungen eingesetzt. Die Anwendung entstand in enger Zusammenarbeit des DAI Berlin mit dem Archäologischen Institut der Universität zu Köln. iDAI.field wurde mit der kommerziellen Software FileMakerPro realisiert. Derzeit gibt es Bestrebungen das System auf Open Source Technologien umzuziehen, eine webbasierte Nutzeroberfläche zu erstellen und offene Schnittstellen für eine erhöhte Interoperabilität zu implementieren.

Analyse der Altsysteme

Im Rahmen der OpenInfRA-Systemspezifizierung wurde bereits eine Defizit-Analyse der Altsysteme durchgeführt, die im Detail in Abschnitt 3.2.3 des Grobkonzepts (OpenInfRA, 2013) nachgelesen werden kann. Ein wesentlicher Kritikpunkt ist bei beiden Systemen unter anderem die Benutzeroberfläche. Das Bedienkonzept von CISAR wird als uneinheitlich sowohl innerhalb der einzelnen Module als auch modulübergreifend bezeichnet und die allgemeine Nutzung des Systems wird als z.T. sehr umständlich befunden. Bei iDAI.field werden viele Fensterwechsel bemängelt und es wird als „auf Funktionalität ausgelegt“ und als eher benutzerunfreundlich angesehen. Als weiteres Manko wird die Benutzer-Hilfe identifiziert. Bei CISAR gibt es keine Online-Hilfe, sondern nur Benutzeranleitungen zu den einzelnen Modulen. In iDAI.field gibt es nur sehr allgemein gehaltene Benutzeranleitungen in Form von Word-Dateien und keine direkte Hilfe in den Oberflächen. Als erhaltenswert werden vor allem das Fachkonzept und das daraus abgeleitete Datenmodell von iDAI.field, sowie die in CISAR genutzten webbasierten Datenbanktechnologien und die auf Open Source-Standardkomponenten aufbauende Client- / Serverarchitektur genannt.

4.3 OpenInfRA-Prototypen

Während der Entwicklung der OpenInfRA-Systemspezifikation sind bisher bereits einige Oberflächen-Prototypen für OpenInfRA entstanden.

4.3.1 Oberflächenprototyp für Nutzer-GUI und Administrations-GUI

Erste Skizzen für einen GUI-Prototyp wurden mit dem OpenSource-Prototyping-Tool Pencil 1.3 von Evolus (s. Abschnitt 3.2) erstellt. In einem weiteren Arbeitsschritt wurden auf Basis der Pencil-Skizzen mit AppSketcher (s. Abschnitt 3.2) verbesserte interaktionsfähige Oberflächenprototypen unter Verwendung von HTML- und jQuery-Komponenten erzeugt (Harazim, 2012). Die entstandenen Prototypen decken die Anforderungsspezifikation in den Bereichen BN_GUI (Benutzung) (Abbildung 6) und AC_Administration (Abbildung 7) ab. Beide

Prototypen liegen in einer Version mit Verweisen auf die zu Grunde liegenden Anforderungen und in einer Version ohne Verweise vor.

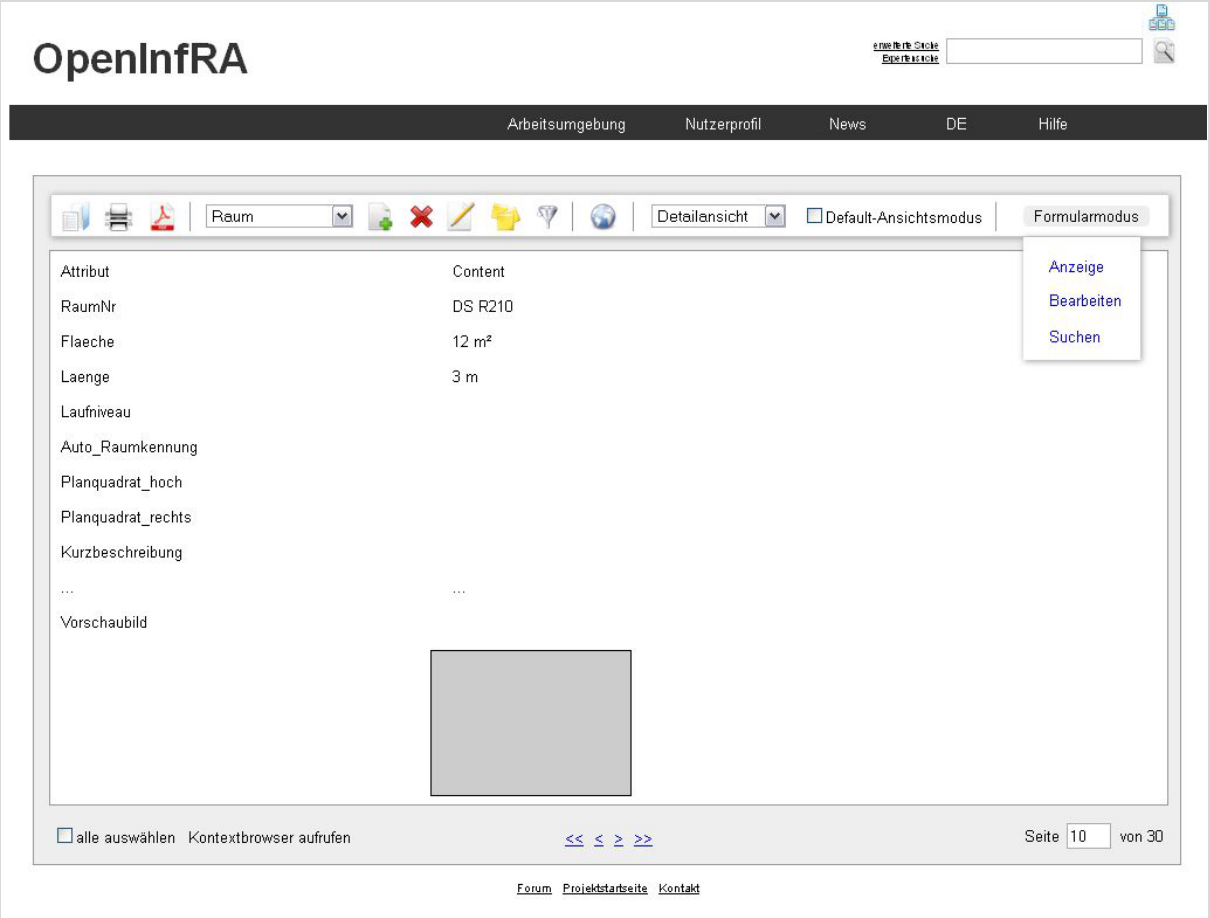


Abbildung 6: Oberflächenprototyp für die Benutzer-GUI (ohne Verweise auf die Anforderungen).

Bei beiden Prototypen handelt es sich um horizontale Oberflächenprototypen, die sich in den Bereich des explorativen Prototypings (s. Abschnitt 3.1) einordnen lassen. Das Augenmerk liegt darauf, einen visuellen Überblick über die bereits spezifizierten Funktionalitäten zu geben. Die Prototypen dienen der Qualitätssicherung der Anforderungsspezifikation und, speziell in der Version mit Verweisen, als Grundlage für die Diskussion und Verfeinerung der Anforderungen mit Projektbeteiligten und Stakeholdern.

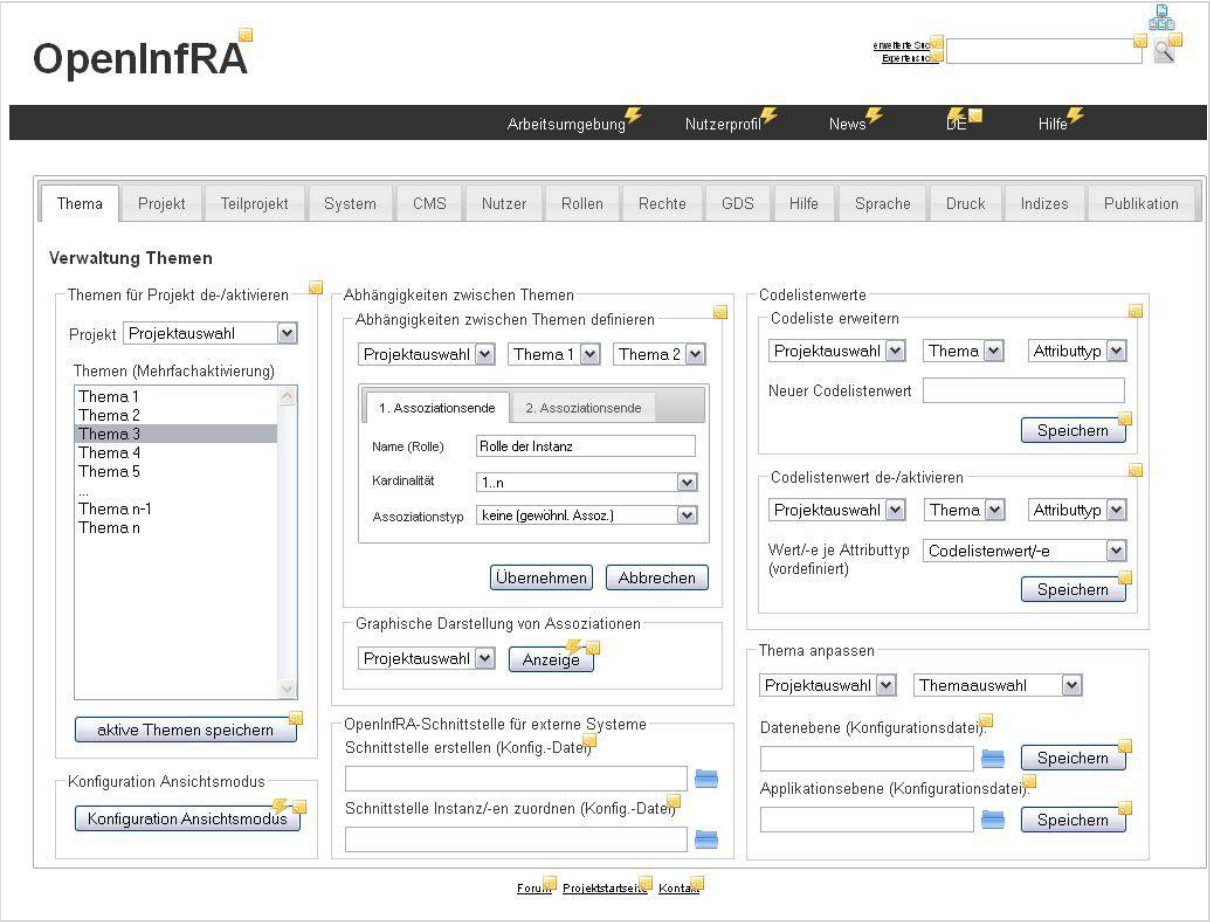


Abbildung 7: Oberflächenprototyp für die Administrations-GUI (mit Verweis auf die Anforderungen).

4.3.2 Kontextbrowser

Um auch einem „Laien“ (im Gegensatz zu einem Fachnutzer) die Exploration des Datenbestandes von OpenInfRA möglichst einfach und visuell unterstützt zu ermöglichen, soll OpenInfRA einen Kontextbrowser bieten, der die Zusammenhänge zwischen verschiedenen Datensätzen graphisch darstellt (Harazim, 2012). Der Kontextbrowser soll intuitiv bedienbar sein und sowohl thematische als auch geographische Kontexte visualisieren. Bei der Umsetzung dieses Prototyps wurden Grundlagen und Richtlinien zur visuellen Suche, zur Farbwahrnehmung, zu Farbfehlsichtigkeiten und zur Schriftgestaltung berücksichtigt. Die Kontextarten in OpenInfRA wurden analysiert und liegen dem entworfenen Kontextbrowser als Hintergrundlogik zu Grunde. Der Kontextbrowser wurde in das Layout der bereits vorhandenen Oberflächenprototypen (s. Abschnitt 4.3.1) integriert und kann über diese aufgerufen werden.

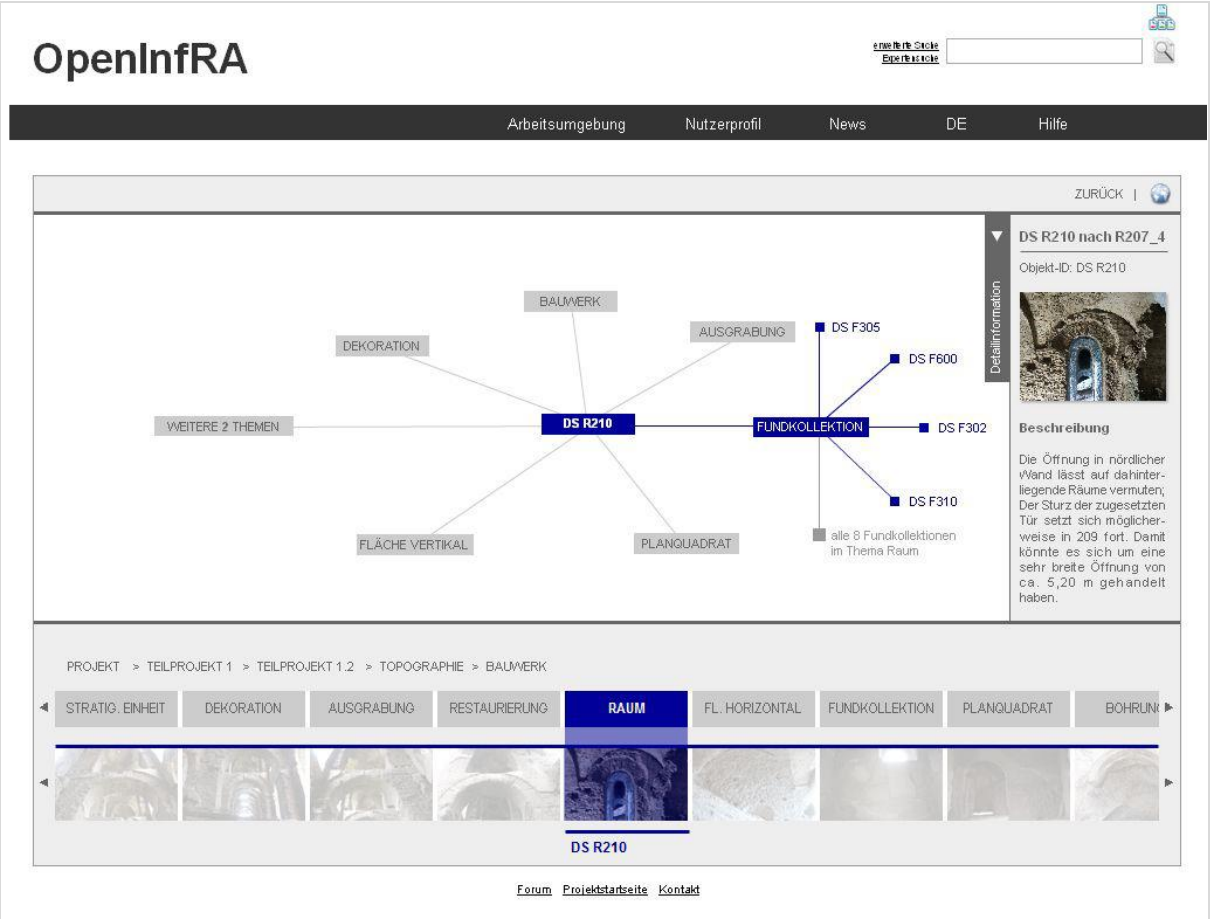


Abbildung 8: Oberflächenprototyp für einen Kontextbrowser.

4.3.3 Prototyp mit Schwerpunkt Barrierefreiheit

Im Rahmen einer Bachelorarbeit an der HTW Dresden (Harazim, 2013) wurden die Anforderungen und -bedingungen der Barrierefreien-Informationstechnik-Verordnung (BITV) 2.0 vom 12. September 2011 (BITV 2.0, 2011) auf das OpenInfRA-Projekt abgebildet. Relevante Anforderungen wurden identifiziert und feinspezifiziert. Als Ergebnis dieser Arbeit liegt die Anforderungskategorie ZLG (Zugänglichkeit) im OpenInfRA-Anforderungskatalog (Anhang B, OpenInfRA, 2013) mit ca. 80 Anforderungen zur Barrierefreiheit vor. Zusätzlich wurde ein barrierefreier Prototyp für die Erweiterte Suche von OpenInfRA entwickelt (s. Abbildung 9).

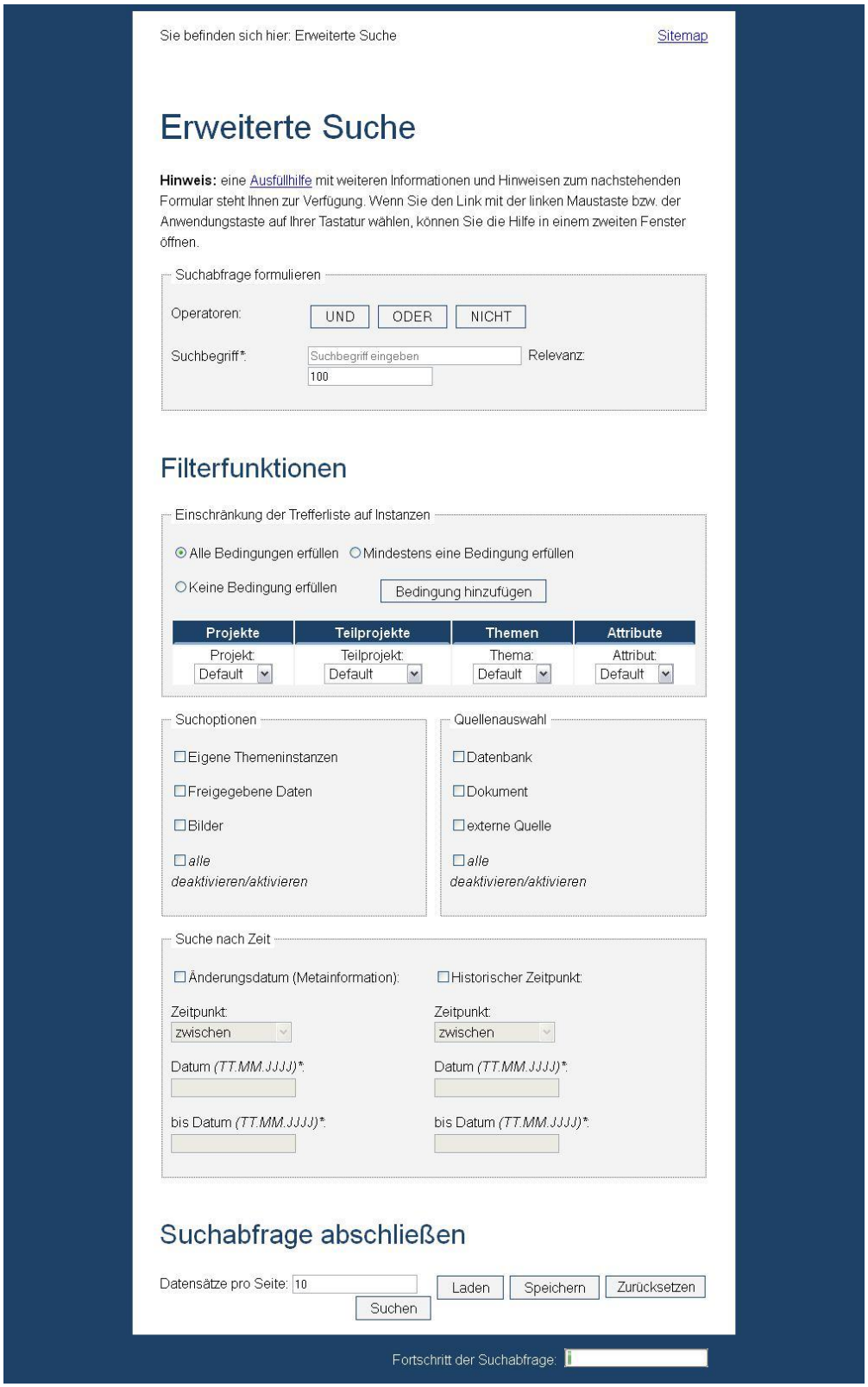


Abbildung 9: Barrierefreier Oberflächenprototyp für die Erweiterte Suche.

Weitere Screenshots der vorgestellten Prototypen finden sich im Anhang F des OpenInfRA Grobkonzept V2.2 (OpenInfRA, 2013). Online besteht Zugriff auf alle Prototypen unter der URL http://geoinformatik.htw-dresden.de/openinfra/.

5 Entwurf

Dieses Kapitel beschreibt die Auswahl des Diskursbereichs und analysiert die Vorgaben für den Entwurf durch Altsysteme, existierende Anforderungen, das DAI und durch die Nutzer.

5.1 Auswahl des Diskursbereichs

Bei der geplanten OpenInfRA-Anwendung handelt es sich um ein umfangreiches System, was sich auch am Umfang des aktuellen Grobkonzepts und des Anforderungskatalogs (ca. 700 Anforderungen) widerspiegelt. Daher ist es im Rahmen dieser Arbeit nicht möglich, einen Oberflächenprototyp für alle Bereiche des Systems zu entwickeln und zu implementieren.

Um eine möglichst repräsentative Auswahl für die Implementierung des Oberflächenprototyps zu treffen, wurden die Anwendungsfälle der OpenInfRA-Spezifikation herangezogen, da sie die eigentlichen Arbeitsaufläufe des späteren Systems spezifizieren. Hier zeigt sich, dass die Anwendungsfälle zur Benutzung des Projekt-Clients eine zentrale Rolle spielen und eine Brückenfunktion darstellen, zwischen weiteren Komponenten und Modulen des Systems, wie z. B. dem WebGIS-Client, dem Kontextbrowser, dem Administrationsclient oder den Suche-Modulen. Für die Implementierung des Oberflächenprototyps wird daher die zentrale Benutzer-GUI des Projekt-Clients ausgewählt.

Die nicht umgesetzten Komponenten / Module der OpenInfRA-Anwendung werden allerdings im Oberflächenentwurf berücksichtigt und könnten ohne großen Aufwand in das entwickelte Layout integriert werden. Abbildung 13 zeigt z. B. eine mögliche Integration des WebGIS-Clients.

5.2 Vorgaben für den Entwurf

5.2.1 Durch Altsysteme

Wie bereits in Abschnitt 4.2 angesprochen, wurden die Altsysteme im Zuge der OpenInfRA-Systemspezifikation bereits hinsichtlich ihres Weiterverwendungspotentials untersucht. Eine Reihe von Anforderungen hinsichtlich des Aufbaus und der verwendeten Technik des Systems wurden extrahiert und in den Anforderungskatalog von OpenInfRA übernommen. Sowohl CISAR als auch iDAI.field weisen bzgl. ihrer Oberflächengestaltung und Benutzerführung allerdings wesentliche Defizite auf. Das Bedienkonzept von CISAR wird als uneinheitlich und umständlich bezeichnet und bei iDAI.field fallen die vielen Fensterwechsel und die generelle Benutzerunfreundlichkeit auf. Eine Hilfe bzw. Dokumentation für den Benutzer ist in beiden Systemen nur rudimentär vorhanden. In Abstimmung mit dem OpenInfRA-Team

wird daher keine eigene Evaluierung der Altsysteme durchgeführt und es fließen wissentlich keine Eigenschaften der Oberflächen der Altsysteme in die GUI-Gestaltung von OpenInfRA ein.

5.2.2 Durch Spezifikation

Wie bereits in Abschnitt 3.3 beschrieben, existiert für OpenInfRA eine umfangreiche Systemspezifikation, deren Herzstück der Anforderungskatalog darstellt (Anhang B, OpenInfRA, 2013). Diese Sammlung funktionaler und nicht-funktionaler Anforderungen ist das ausschlaggebende Dokument für die anstehende Implementierung des GUI-Prototyps. Der Projekt-Client, der für die prototypische Implementierung der GUI ausgewählt wurde, ist definiert durch die Anforderungen der Kategorie BN (Benutzung) des Anforderungskatalogs. Zusätzlich sollen die bereits spezifizierten Anforderungen zur Usability (s. Anhang C – Anforderungskatalog Prototyp V2) berücksichtigt werden. Eine weitere wichtige Vorgabe, die für die Umsetzung der Benutzungsschnittstelle eine Rolle spielt, ist das für OpenInfRA geltende Rollen-Rechte-System, das als Rollen bzw. Nutzergruppen Administratoren (System- und Projekt-Administrator), Projekt-Bearbeiter, eingeschränkte Projekt-Bearbeiter und Projekt-Gast vorsieht, mit in dieser Reihenfolge abnehmenden Rechten bzgl. Funktionalität und Inhalt.

Auf die Umsetzung der Barrierefreiheit wird in weiten Teilen verzichtet, da diese durch die Bachelorarbeit "Untersuchung zur barrierefreien Präsentation von Webinhalten und praktische Umsetzung am Beispiel des Projektes OpenInfRA" (Harazim, 2013) bereits für OpenInfRA umgesetzt wurde, und der Fokus der vorliegenden Arbeit auf den Komponenten Layout, Design und Benutzerführung liegen soll.

Eine detaillierte Liste der bei der Umsetzung des Prototyps berücksichtigten relevanten funktionalen und nicht funktionalen Anforderungen findet sich im Anhang C.

5.2.3 Durch das Deutsche Archäologische Institut

Da die OpenInfRA-Anwendung langfristig in die IT-Infrastruktur des DAI eingebunden wird (s. Abschnitt 4.1), soll sie sich in ihrer Gestaltung an dem Corporate Design des DAI orientieren.

Der maßgebliche DAI-Styleguide (s. auch Anhang C) umfasst 52 Seiten, unterteilt in 5 Kapitel (s. Tabelle 3), die Vorgaben zum Design, zum Layout und zur Typographie definieren.

Tabelle 3: Inhalt des DAI-Styleguides.

00 Styleguide	
01 Adaptive Layout	Sizes and Breakpoints Flexible Solutions

	Map Layout
02 Fundamentals	Grid Details The Baseline Colors Main Navigation Second Navigation Backgrounds and Pattern Widgets Footer Map
03 Typographie	Sorce Sans Headlines and Copy
04 Main Content	Links Content Module Overview (Small and Medium Size) Content Module Overview (Large Size)
05 Thanks	Prototyp: https://nonkonform.hotgloo.com/wf/eabebe30 Map Prototyp (map 2+3): http://gunnarfriedrich.info/map/

Während der DAI-Styleguide zu den oben genannten Bereichen teilweise sehr detaillierte An- und Vorgaben macht, liegen dem Guide auch Links zu zwei Prototypen bei. Des Weiteren gibt es mehrere DAI-Projekte die für ihre Webseiten (teilweise zunächst als Oberflächenprototypen) die Vorgaben des Styleguides bereits umgesetzt haben. Diese werden im weiteren Verlauf als Referenzimplementierungen bezeichnet:

- ZENON (= Bibliothekskatalog)

  http://testopac.dainst.org/ (dargestellt in Abbildung 10)

- iDAI.gazetteer (= Ortsdatenverwaltung)

  http://virginiaplain06.klassarchaeologie.uni-koeln.de:8080/gazetteer/

- ARACHNE (= Objektdatenbanken)

  http://crazyhorse.archaeologie.uni-koeln.de/~kummer/arachne4/

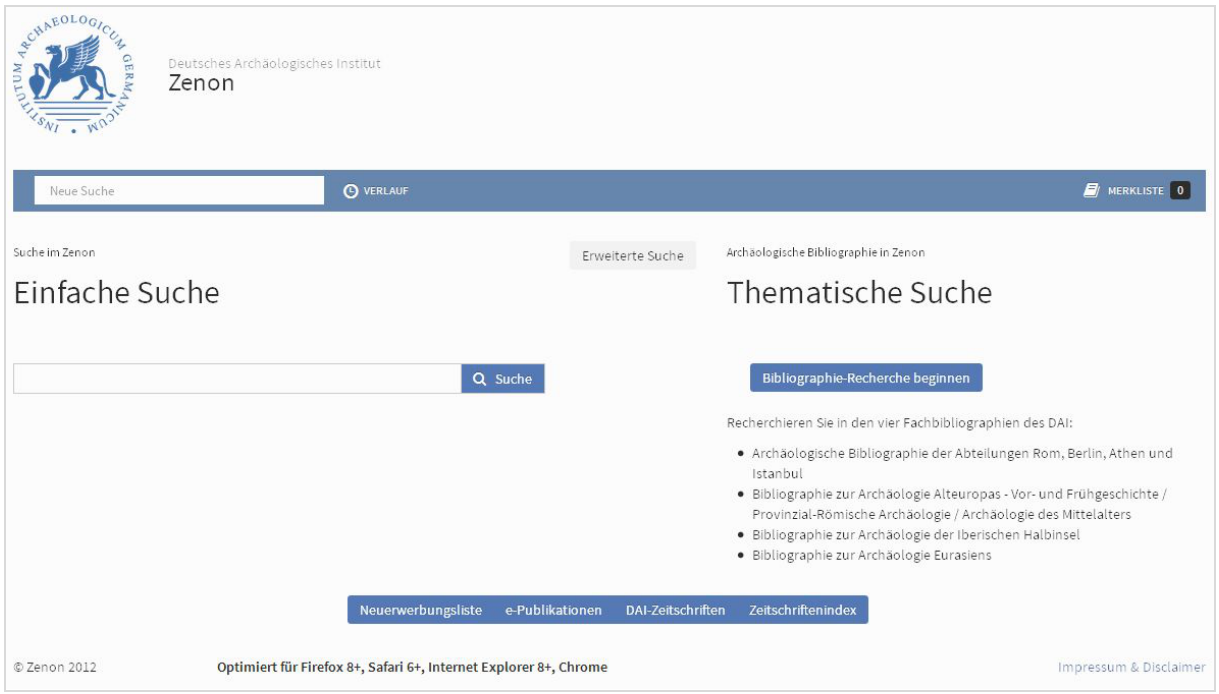


Abbildung 10: Webseite von Zenon nach dem DAI-Styleguide.

Von besonderer Bedeutung für das Corporate Design des DAI sind der Header mit dem DAI-Logo und dem projektspezifischen Webseitentitel, das verwendete Farbschema und der verwendete Font bzw. die Font-Größe. Diese Vorgaben werden beim Entwurf des GUI-Prototyps daher primär beachtet. Weiteren Vorgaben zur Strukturierung der DAI-Webseiten werden auch durch die DAI-Referenzimplementierungen weniger stringent umgesetzt und flexibel den jeweiligen Inhalten und Funktionalitäten angepasst.

5.2.4 Durch Nutzer

Die aktuelle OpenInfRA-Systemspezifikation (OpenInfRA, 2013) basiert zum einen auf einer umfangreichen Analyse der Altsysteme (s. Abschnitt 4.2) und einer Extraktion und Spezifikation der erhaltenswerten Systemeigenschaften und Fachkonzeption. In die weitere OpenInfRA-Spezifikation sind Entwickler der Altsysteme CISAR und iDAI.field und Fachnutzer aus den Bereichen Bauforschung und Archäologie involviert. Durch diese Fachkompetenz wird eine umfangreiche, detaillierte und gleichzeitig benutzerzentrierte Spezifikation möglich. Es wird daher im Vorfeld der Prototypentwicklung auf eine Nutzerumfrage zu Arbeitsabläufen verzichtet, da diese in den Anwendungsfällen bereits ausreichend und erfahrungsbasiert spezifiziert sind. Auch die funktionalen Anforderungen sind bereits umfassend durch Nutzer verifiziert.

5.3 GUI-Entwurf

In diesem Kapitel werden die Überlegungen zum, und die Umsetzung des Entwurfs des OpenInfRA-GUI-Prototyps beschrieben. Es werden Entwurfsentscheidungen und Probleme erläutert.

5.3.1 Layout und Gestaltung

Wie bereits in Abschnitt 2.3.1 beschrieben, sind Kenntnisse über die Nutzer, ihre Aufgaben und Ziele der erste Schritt beim benutzerorientierten Entwurf einer Anwendung. Im Rahmen der umfangreichen OpenInfRA-Spezifikation (Abschnitt 4.1) wurden verschiedene Nutzergruppen mit unterschiedlichen Berechtigungen bzgl. Funktionalität und Inhalt für die Benutzerschnittstelle identifiziert. Um ein möglichst breites repräsentatives Spektrum an Funktionen zeigen zu können, orientiert sich der Entwurf (und auch die Implementierung, s. Kapitel 6) an dem maximalen Umfang an Funktionen und Inhalten, die der Rolle „Projekt-Bearbeiter“ zur Verfügung stehen.

Die Fachterminologie (s. Abschnitt 2.3.3) ist durch die Beteiligung von Entwicklern der Altsysteme und von Fachanwendern an der Anforderungs- und Spezifikationsphase (s. Abschnitt 5.2.4) umfangreich in das Grobkonzept und dessen einzelnen Teildokumente eingeflossen und spiegelt sich in diesen wieder. Daher können Begriffe und Bezeichnungen aus dem Grobkonzept qualitätsgesichert für den Entwurf übernommen werden. Aus dem Anforderungskatalog stammen z. B. das Konzept und die Bezeichnung der unterschiedlichen Sichten (Listen-, Katalog und Detailansicht) für Daten.

Obwohl die Benutzerschnittstelle von OpenInfRA losgelöst von der Oberflächengestaltung der Altsysteme (s. Abschnitt 5.2.1) entworfen werden soll, soll für den Nutzer dennoch eine möglichst vertraute Arbeitsumgebung mit gewohnten Funktionalitäten und Bedienverhalten geschaffen werden. Erlernzeiten und die kognitive Belastung (s. Abschnitt 2.3.2) werden dadurch minimiert und es treten weniger Bedienfehler auf. Daher orientiert sich der Entwurf des GUI-Prototyps an dem Look-and-Feel einer Desktopanwendung. Es wird ein bildschirmfüllendes Layout vorgesehen. Die Strukturierung der Seite verbindet die Anordnung bekannter Desktopelemente, wie z. B. eine *Sidebar* mit Navigation, mit der erwartungsgerechten Anordnung von Bildschirmelementen nach Thesmann (2010), siehe Abbildung 11.

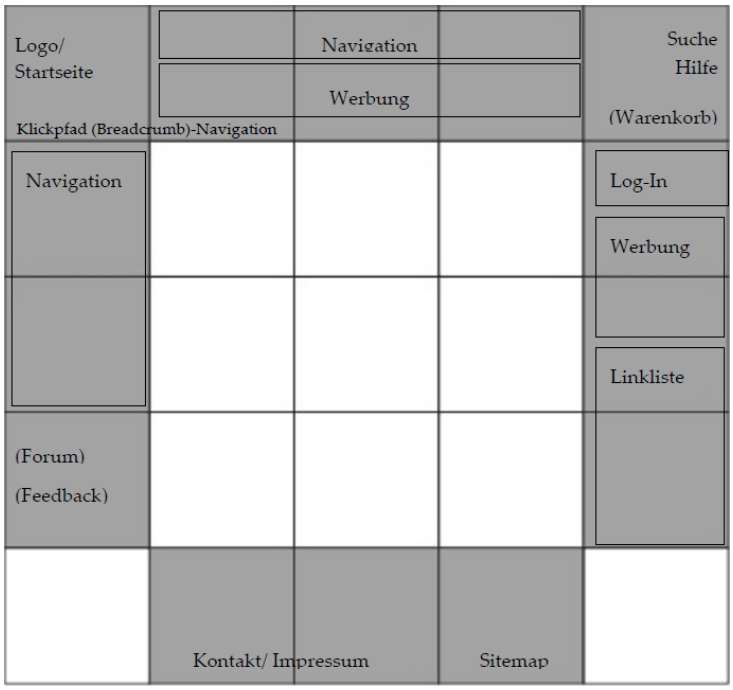


Abbildung 11: Erwartungsgerechte Anordnung von Bildschirmelementen (Thesmann, 2010, S. 297)

Das OpenInfRA-Logo (mit Link zur Startseite) befindet sich links oben im Header und ist im Gegensatz zu den DAI-Vorgaben kleiner gehalten, um mehr Platz für die eigentliche Anwendung zu lassen. Navigationselemente, die immer (unabhängig von Seitenzustand, Ansicht oder Modus) zur Verfügung stehen sollen und vermutlich nicht sehr häufig genutzt werden (Forum, News, Hilfe, Impressum, usw.) werden rechts oben im Header angeordnet.

Die Hauptmenüleiste enthält häufig genutzte Funktionalitäten, wie eine Inhaltsübersicht, die verschiedenen Sichten auf die Daten, und neben Listen-, Katalog- und Detailansicht auch die Kartenansicht und den Kontextbrowser. Es finden sich hier räumlich getrennt von den Sichten weitere häufig genutzte Funktionen wie Drucken, Exportieren, System- und Nutzereinstellungen und die verschiedenen Suchen.

Unter der Hauptmenüleiste befinden sich *bread crumbs*, eine Statusanzeige, die dem Nutzer seine aktuelle Position im Datenbestand anzeigt. An dieser Position werden auch Angaben zu gesetzten Filter oder zu Parametern einer durchgeführten Suchanfrage platziert (dann ist keine eindeutige Position im Datenbestand verfügbar).

Unter den *bread crumbs* ist das Hauptinformationsfenster angeordnet (s. Abbildung 12). Hier werden alle Sichten, Modi, Trefferlisten, usw. angezeigt. In der Detailansicht im Bearbeitungsmodus befinden sich links oben im Hauptfenster die sinnvoll gruppierten Funktionen des Bearbeitungsmodus, wie Daten importieren, Dateien anhängen, Speichern, Detailsuche, Veröffentlichen, Löschen, Abbrechen usw. Rechts oben im Hautinformationsfenster werden Buttons angeordnet, mit denen alle angezeigten Attributtypgruppen geöffnet bzw. kollabiert werden können.

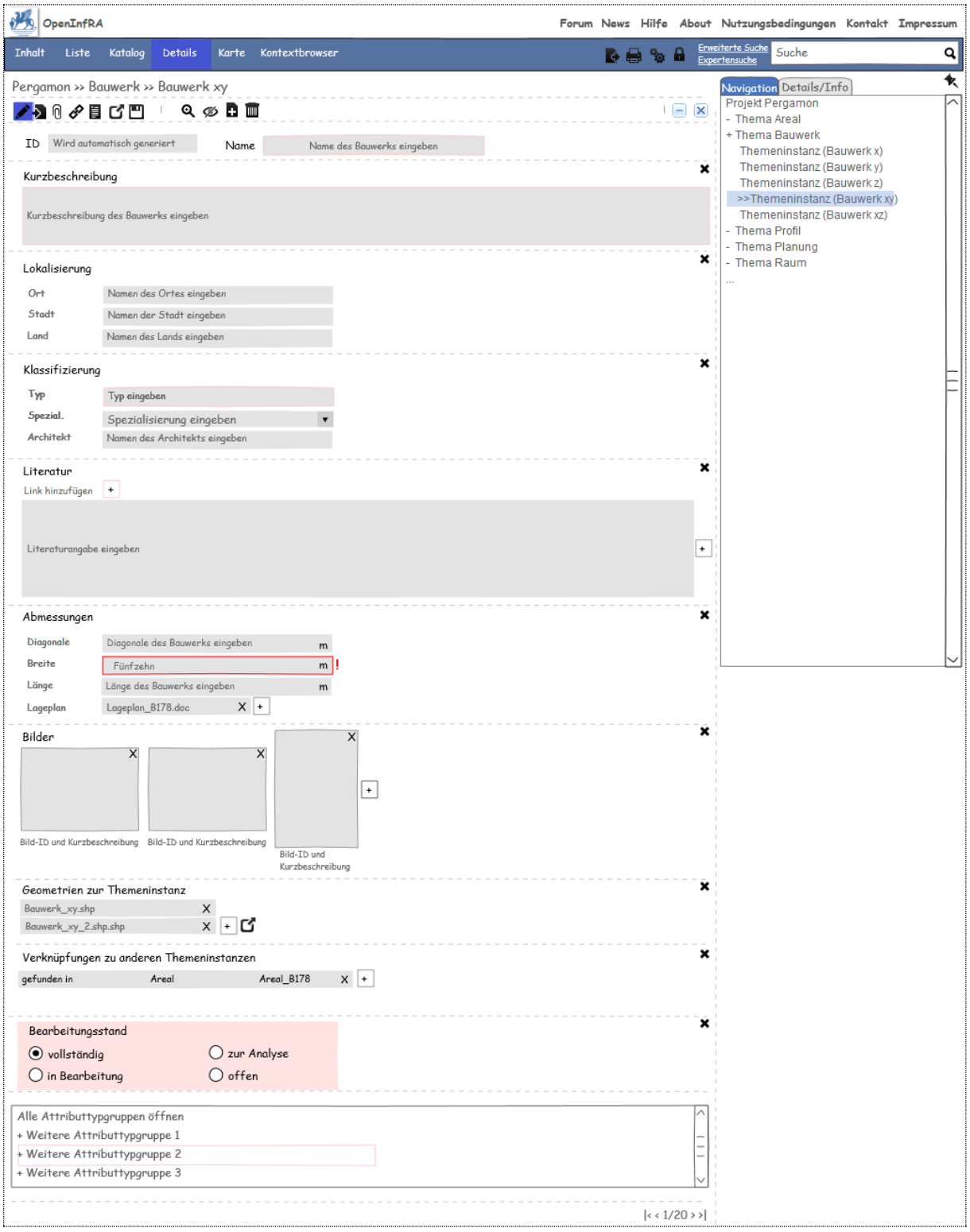


Abbildung 12: Entwurf OpenInfRA-GUI-Prototyp im Bearbeitungsmodus.

Unter den Editierfunktionen sind im Bearbeitungsmodus die Formularfelder für die Dateneingabe nach Attributtypgruppen sortiert und entsprechend der Gestaltgesetze (Abschnitt 2.5) angeordnet. Gerade im Bearbeitungsmodus kommen viele der in Abschnitt 2.3 beschriebenen Entwurfsprinzipen zum Einsatz. Es soll eine Validierung der Nutzereingaben stattfinden,

Fehleingaben sollen durch die Verwendung von Indexlisten und feste Feldformate vermieden werden (Abschnitt 2.3.11). Der Nutzer soll sprechende Rückmeldungen bekommen, wenn Eingaben fehlen oder falsch sind. Die Eingabe von Daten soll rückgängig gemacht und abgebrochen werden können (Abschnitt 2.3.8). Der Bearbeitungsmodus ist so konzipiert, dass vom Ansichtsmodus durch ein Klicken auf angezeigte Daten direkt in den Bearbeitungsmodus gewechselt werden kann.

Ganz unten auf der Seite befindet sich eine Paginierung, die dem Nutzer die Navigation durch den Datenbestand erlaubt. Diese ist inaktiv, wenn nicht geblättert werden kann, weil keine Daten verfügbar sind.

Auf der rechten Seite befindet sich eine *Sidebar*, die primär einen hierarchischen Themenbaum zur schnellen Navigation durch den Datenbestand enthält. Über Tabs oder Ziehharmonikas können hier weitere Funktionalitäten, wie Sortieren, Filtern, Kurzinformationen / Hilfe oder evtl. auch temporäre Kontextmenüs angezeigt werden. Liegt der Fokus des Benutzers auf dem Hauptinformationsfenster kann die Breite der Sidebar angepasst oder diese vollständig an den Bildschirmrand kollabiert werden. Angestrebt wird eine beliebige Platzierbarkeit der Sidebar durch den Nutzer nach rechts oder nach links und eine entsprechende Anpassung des Hauptinformationsfensters.

Das generelle Layout der Anwendung soll aber nicht nur für den gewählten Diskursbereich (s. Abschnitt 5.1), also den Projekt-Client, sondern auch für die Integration weiterer OpenInfRA-Module wie den WebGIS-Client, den Kontextbrowser und die Suche-Module (Einfache Suche, Erweiterte Suche, Expertensuche, Detailsuche, Trefferlisten, Filter) geeignet sein. Die Anordnung dieser Komponenten ist im Hauptinformationsfenster vorgesehen, während deren explizite Funktionalitäten in die *Sidebar* übernommen werden können. Dies ist in Abbildung 13 an Beispiel des 2D-WebGIS-Client dargestellt.

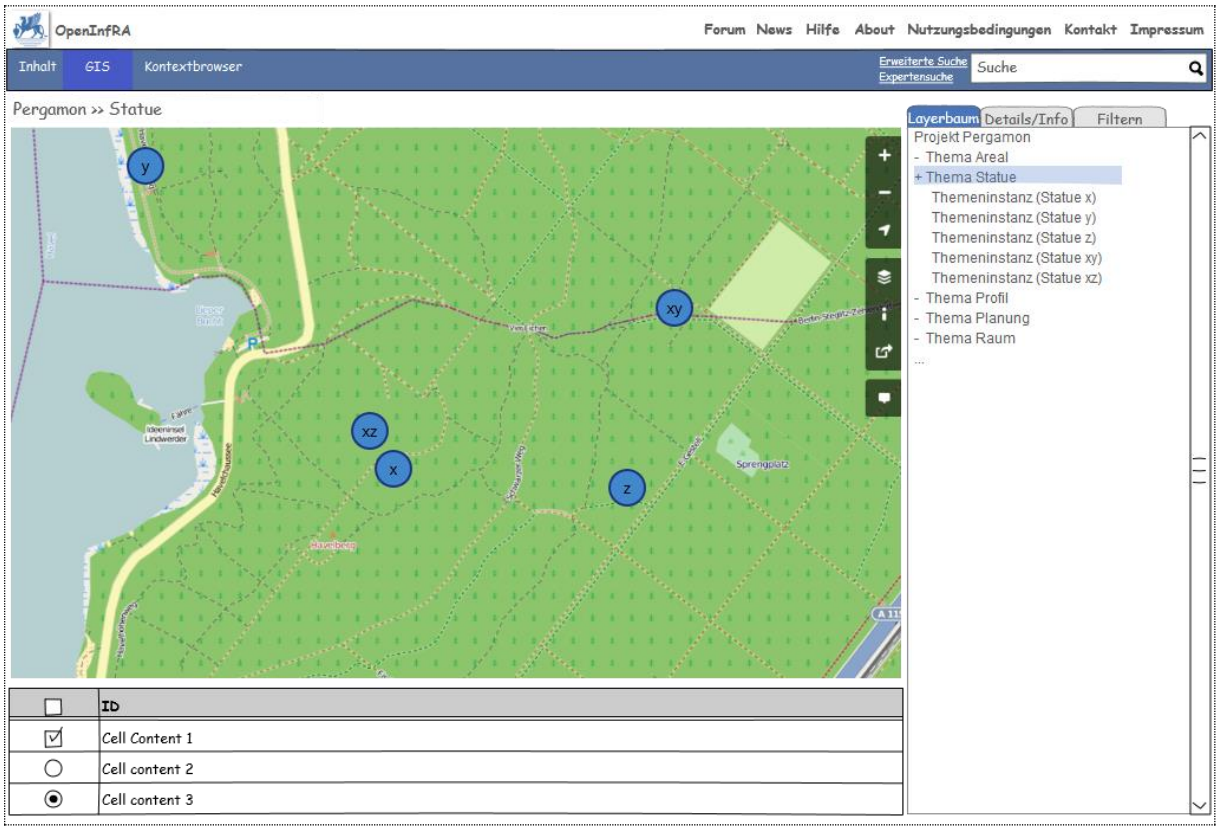


Abbildung 13: OpenInfRA-GUI-Layout mit eingebetteter WebGIS-Komponente.

Es wird eine hohe Adaptierbarkeit (s. Abschnitt 2.3.12) der Benutzerschnittstelle angestrebt. Bei OpenInfRA handelt es sich nicht um eine Webseite mit oft recht kurzer Verweildauer und schnell wechselnder Klientel, sondern um eine webbasierte Anwendung, die zum Teil von Fachanwendern über lange Zeiträume hinweg sehr häufig benutzt wird, zum einen zur Datenerfassung und -dokumentation, zum anderen aber auch in Ansätzen für erste Analysen (Suche, Filter, Anzeige und Suche räumlicher Daten) (s. Abschnitt 5.2). Ein weiterer erfahrener Nutzer von OpenInfRA ist der Administrator. Da die OpenInfRA-Anwendung für möglichst viele auch sehr inhomogene Projekte verwendet werden kann, liegt ihr ein sehr generisches Datenmodell zu Grunde. Das führt dazu, dass die Anpassungsmöglichkeiten an Projekte sehr umfangreich sind und ein entsprechend komplexer und hoher Administrationsaufwand besteht (s. Abschnitt 5.2). Gleichzeitig soll OpenInfRA allerdings auch direkt als Online-Publikationsplattform für die Veröffentlichung der erfassten Fachdaten genutzt werden. Das bedeutet, dass auch unerfahrene Nutzer die Anwendung nutzen werden. Es wird deutlich, dass es sich hierbei bzgl. ihrer Ziele, Aufgaben und Kenntnisse um sehr unterschiedliche Nutzergruppen handelt. Die Benutzerschnittstelle soll daher zum einen bzgl. ihrer Wahrnehmungsfähigkeit anpassbar sein (z. B. bzgl. Schriftgröße, Farben). Da die unterschiedlichen Benutzergruppen mit unterschiedlichen Berechtigungen (Abschnitt 4.1) auf die Benutzerschnittstelle zugreifen, muss zum anderen auch die Ausprägung der GUI bzgl. angebotener Funktionen und Inhalte anpassbar sein. Da die Anwendung in sehr unterschiedlichen Umge-

bungen zum Einsatz kommen kann, ist eine Anpassungsfähigkeit an unterschiedliche Endgeräte wünschenswert, wie sie in Abschnitt 2.3.12 beschrieben wird.

Vorgaben des DAI-Styleguides bzw. der Referenzimplementierungen bzgl. Farben, Fonts, und Font-Größen werden berücksichtigt.

5.3.2 Probleme

Wie bereits im vorhergehenden Abschnitt erwähnt, handelt es sich bei OpenInfRA nicht um eine Webseite sondern eine Webapplikation, die Nutzergruppen mit unterschiedlichen Berechtigungen bedient. Daher variiert die GUI durch die entsprechend vorhandenen oder nicht vorhandenen Funktionalitäten stark. Dennoch soll sich die Haupt-Struktur der Applikation nicht verändern, um dem Nutzer eine konsistente Erfahrung (Abschnitt 2.3.9) zu bieten. Es ist nicht einfach ein Konzept / Layout zu finden, dass dem gerecht wird.

Funktionen, die auf Grund ihrer inhaltlichen Zusammengehörigkeit eigentlich gruppiert und nah beieinander angeordnet werden sollten, wie z. B. der Import und Export von Daten, können auf Grund der starken Verschachtelung von unterschiedlichen Berechtigungen und Ansichten nicht zusammen auf der GUI verortet werden. Ein Import von Daten ist nur in der Detailansicht im Bearbeitungsmodus zulässig und nur für einen berechtigten Bearbeiter möglich. Ein Export von Daten ist hingegen, wenn diese freigegen sind, für jeden Nutzer ohne besondere Berechtigung in jedem Ansichtsmodus (Liste, Katalog, Detail) möglich.

In der Detailansicht sind unterschiedliche Formularmodi verfügbar: Anzeigen, Bearbeiten und Suchen. Es ist vorgesehen, aus dem Anzeigemodus über ein Klicken in den Formularbereich direkt in den Bearbeitungsmodus wechseln zu können, so dass in beiden Modi eine Möglichst gleichbleibende Sicht auf eine Themeninstanz besteht. Dadurch, dass im Bearbeitungsmodus jedoch Funktionalitäten besser nicht oder zumindest nicht nur über die Funktionsbuttons in der Editiertoolbar zur Verfügung stehen sollten, sondern auch direkt an Formularfeldern (z. B. beim Hinzufügen weiterer Links auf externe Ressourcen oder von Bildern und Geometrien zur Themeninstanz) gestaltet sich dieses Vorhaben schwierig. In einer weiteren Ausbaustufe des Prototyps sollte dieser Ansatz nicht mehr verfolgt werden. Die Modi Anzeigen und Bearbeiten unterscheiden sich doch zu stark.

5.3.3 Mapping GUI-Entwurf / MVC-Implementierung

Da die Implementierung nach dem MVC-Konzept (Krasner & Pope, 1988) erfolgen soll, muss bereits im Entwurf eine Strukturierung und Zuordnung der zu entwickelnden GUI-Komponenten auf die Bereiche Model, View und Controller erfolgen. In Abschnitt 3.3.4 wird die Implementierung von MVC in Ext JS erläutert.

Als Models werden Projekt und Nutzer definiert. Auf Basis der für die GUI-Implementierung relevanten datengebundenen Komponenten (s. Abbildung 14) werden die Data-Stores Projekt und Nutzer identifiziert.

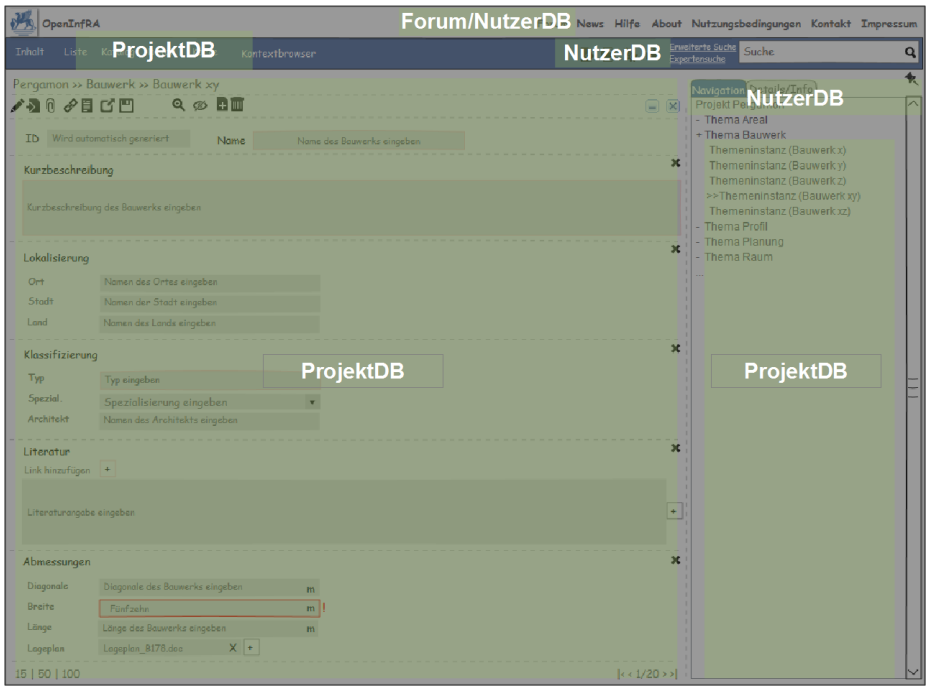


Abbildung 14: Relevante datengebundene Komponenten des GUI-Protoyp-Entwurfs.

Die für den Entwurf definierten Views sind in Abbildung 15 dargestellt.

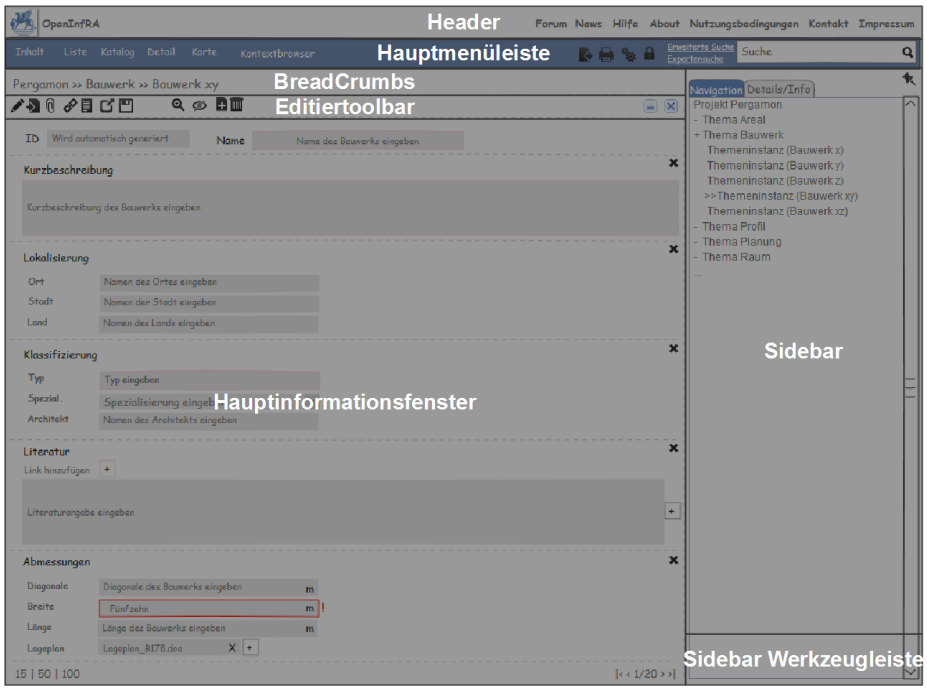


Abbildung 15: Views (nach MVC-Konzept von Ext JS) des GUI-Prototyp-Entwurfs.

Die im Entwurf identifizierten und zugeordneten Controller zeigt Abbildung 16.

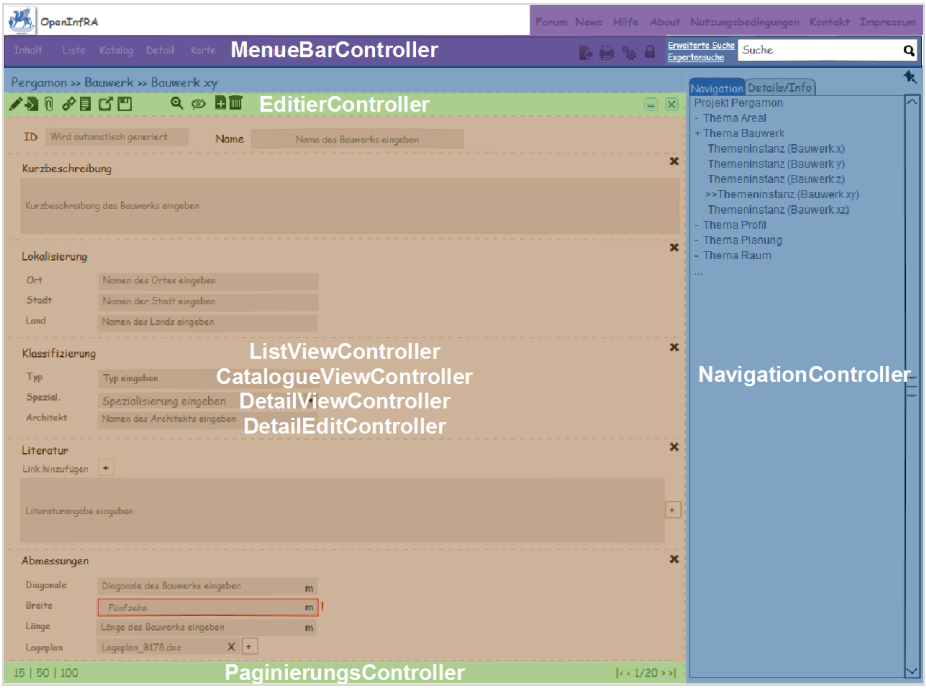


Abbildung 16: Controller (nach MVC-Konzept von Ext JS) des GUI-Prototyp-Entwurfs.

6 GUI-Implementierung

Abbildung 17 zeigt die Umsetzung des GUI-Prototyps basierend auf dem in Abschnitt 5.3 beschriebenen Entwurfs.

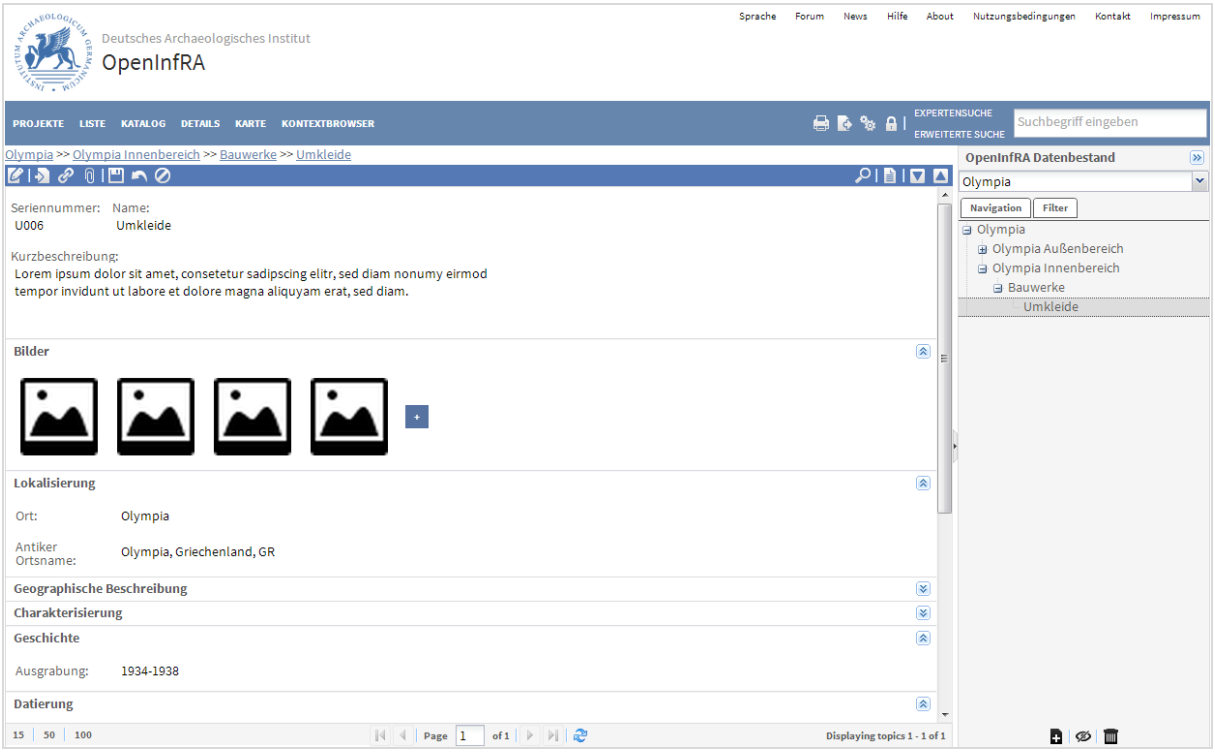


Abbildung 17: Der entwickelte GUI-Prototyp zeigt die Detailansicht einer Themeninstanz.

Die folgenden Funktionalitäten wurden aus dem Entwurf ausgewählt und umgesetzt, und bilden den Fokus in diesem Kapitel (siehe auch Abbildung 17):

- Navigation durch den Datenbestand über den Baum auf der rechten Seite und die *bread crumb* Leiste unter dem blauen Hauptmenü
- Dynamische Erzeugung verschiedener Sichten auf denselben OpenInfRA-Datenbestand:
    - Listenansicht (dargestellt) in Tabellenform
    - Katalogansicht mit Kurzinformationen
    - Detailansicht mit allen Informationen zu den einzelnen Objekten
- Verknüpfung zwischen Projektauswahl in den Navigationsbereichen und den drei Sichten auf den Datenbestand
- Verknüpfung zwischen den verschiedenen Sichten auf den OpenInfRA-Datenbestand untereinander

- Editiermodus mit einfacher Feldvalidierung über Formulare

Darüber hinaus werden die verwendeten Technologien begründet und beschrieben (Abschnitt 6.1), die Umsetzung des DAI-Styleguides und den Referenzimplementierungen erläutert (Abschnitt 6.3), sowie die Evaluierung anhand der, durch den Prototypen umgesetzten, funktionalen Anforderungen und Anforderungen an die Usability (Abschnitt 6.4) diskutiert. Das Kapitel endet mit einer Beschreibung der verschiedenen Entwicklungsphasen und den dazugehörigen Nutzerbefragungen (Abschnitt 6.5).

Eine im Internet verfügbare Version mit dem Stand der Abgabe der Arbeit befindet sich hier:

- http://magdalinski.eu/mscmag_v2/
- Nutzer: openinfra
- Passwort: guiprot!

Um die Hauptfunktionalitäten kennenzulernen kann die Kurzanleitung aus Anhang A verwendet werden. Die in der Kurzanleitung gezeigten Funktionalitäten dienen gleichzeitig als Abgrenzung zu Funktionalitäten, die nicht oder nur teilweise umgesetzt werden. Auf eine entsprechende Abgrenzung in der Implementierung z. B. durch Popups oder Tooltips wird deshalb verzichtet.

Der vollständige dazugehörige in UTF-8 codierte und kommentierte Quellcode befindet sich in Anhang C. Kleinere Codeauszüge werden im Text diskutiert. Die für einige Buttons verwendete Icon-Bibliothek von gentleface (http://gentleface.com/free_icon_set.html) unterliegt der Creative Commons Lizenz CC BY-NC 3.0 (http://creativecommons.org/licenses/by-nc/3.0/). Dieser Hinweis muss in das Impressum des GUI-Prototyps aufgenommen werden.

6.1 Verwendete Technologien

Die in Abschnitt 3.2 beschriebenen Werkzeuge zur GUI-Prototyp-Entwicklung werden für die Implementierung nicht verwendet, da ein evolutionärer Prototyp (s. Abschnitt 3.1) mit einer realen Codebasis entwickelt wird, um die Möglichkeit zu erhalten, dass das in der nächsten Phase von OpenInfRA entwickelte Produktivsystem auf dem Prototypen aufsetzen kann. Generell ist es so auch möglich, JS-Frameworks und HTML5 besser kennenzulernen und im Detail besser konfigurieren zu können.

Den Prototypen ausschließlich auf HTML5-Basis zu entwickeln ist ausgeschlossen worden. Zum Zeitpunkt der Implementierung ist HTML5 noch nicht fertig entwickelt und ausgereift. Die Verwendung eines JS-Frameworks wurde vorgezogen, da es auf eine große Basis an bereits implementierten GUI-Elementen zugreifen kann und durch die Verwendung von einzelnen HTML5-Elementen trotzdem eine plattformunabhängige Darstellung in den verschiedenen Browsern mögliche macht. Ein späterer Umstieg auf HTML5 oder eine verstärkte Nut-

zung von HTML5 ist so trotzdem möglich – ohne auf die etablierten Funktionalitäten eines JS-Frameworks zu verzichten.

Die JS-Frameworks Dojo und Ext JS sind ähnlich ausgereift und gut nutzbar. Beide unterstützen sowohl HTML5 und die OO-Programmierung (durch Verwendung von Klassen im Sinne der OO), als auch direkt eingebaute MVC-Unterstützung, die die Trennung nach Model, View und Controller erlaubt. Die Entscheidung, Ext JS zu nutzen, ist hauptsächlich durch das GeoExt Framework bedingt, das evtl. für die Entwicklung des OpenInfRA-WebGIS-Clients genutzt werden soll. Da GeoExt auf Ext JS basiert und der WebGIS-Client in die OpenInfRA-Entwicklungen genauso integriert werden soll, wie der GUI-Prototyp, ist Grundlage für die Entscheidung für Ext JS. Ext JS wird in der Version 4.2.0 verwendet und unterliegt der GNU-GPL-Lizenz 3.0 (http://www.gnu.org/copyleft/gpl.html) mit einigen möglichen Ausnahmen (Details auf http://www.sencha.com/products/extjs/license/).

Das für die DAI-Referenzimplementierungen verwendete auf dem Framework Bootstrap basierte Archaeostrap konnte für die Entwicklung des GUI-Prototyps nicht mehr berücksichtig werden, da die Entwicklungen mit Ext JS schon zu weit fortgeschritten waren. Nichtsdestotrotz wurden das Archaeostrap-CSS und dich wichtigsten Designelemente zum großen Teil mit Ext JS umgesetzt (weitere Details in Abschnitt 6.3). Für eine verbesserte Integration kann in Zukunft über eine Umsetzung mit Bootstrap nachgedacht werden, besonders dann, wenn für den WebGIS-Client doch Alternativen zu GeoExt verwendet werden. Durch eine Nutzung von Archeostrap kann eine nahtlosere Einbindung des OpenInfRA-Clients in die Seiten und das zukünftige Corporate Design des DAI erfolgen.

Um die Funktionsweise des GUI-Prototyps zu zeigen, ist eine Anbindung an die volle OpenInfRA-Datenbasis nicht erforderlich. Es reicht aus, eine repräsentative Untermenge an Datensätzen zu bilden und für den Prototyp zur Verfügung zu stellen. Zur Integration dieser Datensätze in den Prototypen werden Dateien in der JavaScript Object Notation (JSON) verwendet. Sie bieten einen Kompromiss aus geringer Komplexität aber dennoch ausreichend formalisierter Datenstruktur, um z. B. GUI-Elemente dynamisch erzeugen zu können. Die generell gute Unterstützung von JSON in JavaScript im Generellen und in Ext JS im Speziellen führt dazu, dass die Anbindung so effizient und mit angemessenem Aufwand umgesetzt werden kann.

Für das Design der GUI-Elemente und des Inhalts wird sowohl in HTML als auch in jeglichen JS-Frameworks CSS verwendet. Das Anpassen des Designs des GUI-Prototypen anhand des DAI-Styleguides und den Referenzimplementierungen ist deshalb ebenso mit CSS erfolgt (Details in Abschnitt 6.3).

Obwohl durch die Verwendung von Ext JS und CSS generell browserunabhängig, sind feine Unterschiede in der Darstellung möglich, die die Funktionalität nicht einschränken. Deshalb wird Google Chrome zur Ansicht empfohlen. Eine Nutzung im Internet Explorer und in Firefox ist generell aber auch möglich.

Auf der Serverseite wird Lighttpd für das Hosting des online verfügbaren Prototyps (siehe Einleitung zu Kapitel 6) verwendet. Generell können aber auch alternative HTTP-Server eingesetzt werden, wie z. B. der ebenfalls leichtgewichtige Nginx (http://nginx.org) oder der schwergewichtigere Apache-HTTP-Server (http://httpd.apache.org). Besondere Anforderungen an den zu verwendenden Webserver gibt es nicht.

Für die Entwicklung wird die Entwicklungsumgebung WebStorm (http://www.jetbrains.com/webstorm/) genutzt.

6.2 Umsetzung des Model-View-Controller-Konzepts

Entsprechend des MVC-Konzepts von Ext JS (s. Abschnitt 3.3.4) und des GUI-Entwurfs (s. Abschnitt 5.3.3) wird eine Projektstruktur angelegt (Abbildung 18).

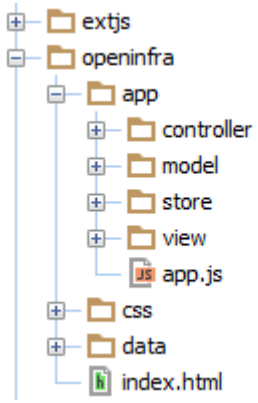


Abbildung 18: Übersicht über die Projektstruktur des GUI-Prototyps, bestehend aus Ordner für die Ext JS Bibliothek, der OpenInfRA-App (weiter untergliedert nach Model, View, Controller und einem Datastore), den CSS und einem Ordner für die JSON-Files (data).

Neben den Ordnern für die Ext JS-Bibliothek (*extjs*), für die CSS und die Datenhaltung (*data*), ist das Projekt für den eigentlichen GUI-Prototyp weiter in Model, View und Controller untergliedert. Die Klassen zur Haltung der JSON-Files befinden sich im Ordner *store*. Der Prototyp wird über die *index.html* gestartet.

Abbildung 19 zeigt die komplette Verzeichnisstruktur mit der Ext-JS-Bibliothek, allen selbst entwickelten Klassen im Ordner *app*, den JSON-Dateien, CSS und der *index.html*.

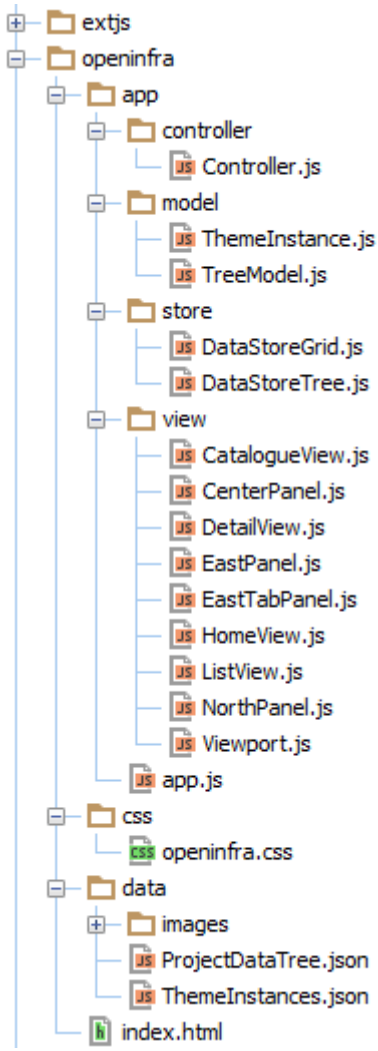


Abbildung 19: Übersicht über die Projektstruktur des GUI-Prototyps inklusive aller erzeugten Klassen.

Um die Klassen gemäß dem MVC-Konzept nutzen zu können, werden sie in der *app.js* konfiguriert (siehe Codeauszug in Abbildung 20). Wenn der Viewport in einer *Viewport.js* definiert und erzeugt wird, erfolgt der weitere GUI-Aufbau automatisch. Eine Aufzählung der View-Klassen ist dann nicht erforderlich.

```
Ext.application({
    name: 'Openinfra',

    // Automatically call Viewport.js as main class / container for the app.
    autoCreateViewport: true,

    // Declaration of model, store and controller classes.
    models: ['ThemeInstance', 'TreeModel'],
    stores: ['DataStoreGrid', 'DataStoreTree'],
    controllers: ['Controller'],
```

Abbildung 20: Deklaration der Klassen für Model, View und Controller in der *app.js* (Codeauszug).

Die wichtigsten Funktionalitäten, Einzelheiten zur konkreten Umsetzung und das Zusammenspiel der in MVC unterteilten Klassen werden in den folgenden Kapiteln beschrieben.

6.2.1 Model-Klassen und dazugehörige Datastores

In den Model-Klassen werden die Datenstrukturen für die verwendeten Elemente im GUI-Prototyp festgelegt. Dazu zählen hier einerseits die einzelnen Themeninstanzen als auch die Struktur für einige GUI-Elemente. Die eigentlichen Daten werden in JSON-Dateien vorgehalten (s. Abschnitt 6.1), die über *Datastores* angebunden werden. Diese regeln den direkten Lese- und Schreibzugriff auf die JSON-Dateien und kapseln von der in der Implementierung verwendeten Datenstruktur im Model. Obwohl die Datenstruktur der JSON-Dateien den realen Daten nahe kommt, ist der Inhalt rein fiktiv.

Ursprünglich waren für das Model Klassen zu allen in OpenInfRA an dieser Stelle relevanten Daten vorgesehen (u. a. *Project*, *PartProject*, *Theme*, *Theme Instance*, *AttributeTypeGroup* und *AttributeType*), die inklusive Ihrer Kardinalitäten und Hierarchie entsprechend untereinander verknüpft waren. Das Klassenmodell von Ext JS unterstützt Vererbung, Abhängigkeiten in Kompositionen und Aggregationen und entsprechende Kardinalitäten. In Abbildung 21 werden die Deklaration von Vererbung, Abhängigkeiten, Attributdefinition und der Kardinalität und Komposition zu einem *PartProject* exemplarisch dargestellt. Die Abbildung der weiteren Hierarchie der relevanten Daten wäre analog erfolgt.

```
Ext.define("Openinfra.model.Project", {

    // Inheritance and dependencies
    extend: 'Ext.data.Model',
    requires: 'Openinfra.model.PartProject',

    // Attributes
    fields: [
        {
            name: 'text',
            type: 'string'
        },
        {
            name: 'id',
            type: 'string'
        }
    ],

    // Composite attributes and respective cardinalities.
    hasMany: {
        model: 'PartProject',
        name: 'Teilprojekte'
    }
});
```

Abbildung 21: Definition der Datenstruktur *Project* zur Darstellung der umsetzbaren Aspekte der OO-Programmierung.

Diese gemäß OO-Paradigmen sinnvolle Klassenaufteilung wurde aus zwei Gründen wieder verworfen. Erstens, kann die Syntax von JSON ein hierarchisches Datenmodell zwar abbilden, aber für eine Suche nach Objekten mit einer bestimmten ID und für eine gleichzeitige Nutzung zum Aufbau von GUI-Elementen (z. B. dem Navigationsbaum) müssen die Elemente des Baums immer den gleichen Namen bzw. Typ haben und die Blätter müssen eindeutig mit dem Attribut *leaf* versehen werden (dies sieht das OpenInfRA-Datenmodell nicht vor). Für die Darstellung von Themeninstanzen ist es notwendig, dass Blätter des Baums verschiedene Namen haben können müssen (z. B. Attributtypgruppen, dann Attributtypen, usw.). Mindestens eine zweite Datenstruktur für den Navigationsbaum ist also erforderlich. Zweitens können dynamische Datenstrukturen, die das OpenInfRA-Datenmodell vorsieht nicht abgebildet werden. Zum Definitionszeitpunkt muss die Struktur der Modelklassen genau mit der JSON-Struktur übereinstimmen – eine nachgelagerte Konfiguration bzw. Änderung zur Laufzeit kann nicht mehr berücksichtigt werden. Der zweite Grund ist für den Prototyp vernachlässigbar. Der erste Grund ist wird über eine implizite Verknüpfung der Daten über das Attribut *directparent* gelöst, das die ID des Elternknoten enthält (siehe Abbildung 22). In der Datenstruktur *TreeModel* besitzen alle Zweige den gleichen Namen und die Blätter sind entsprechend als *leaf* markiert. Dennoch werden dieselben IDs für die Zuordnung zu Projekte, Teilprojekten und Themen verwendet. Dieser Aufbau (Abbildung 23) ermöglicht den Aufbau des Projektbaums.

Die Kombination und Integration beider Datenmodelle ermöglicht einerseits eine syntaktisch lose und flache Hierarchie für die Themeninstanzen (Zweige können unterschiedliche Namen haben), als auch das erforderliche Mapping auf die Projekte, Teilprojekte und Themen, die im *TreeModel* abgelegt sind.

```
"Themeninstanzen": [
    {
        "text": "Tempel des Traianus",
        "Seriennummer": "TdT001",
        "Kurzbeschreibung": "Lorem ipsum dolor sit amet.",
        "id": "themeinstance01",
        "directparent": "theme01",
        "Attributtypgruppen": [
            {
                "name": "Lokalisierung",
                "Attributtypen": [
                    {
                        "Ort": "Bergama",
                        "Antiker Ortsname": "Pergamon, Türkei, TR"
                    }
                ]
            },
```

Abbildung 22: Codeauszug aus der *ThemeInstances.json*, der einen Teil der OpenInfRA-Themeninstanz *Tempel des Traianus* zeigt.

```
{
    "children": [
        {
            "text": "Pergamon",
            "id": "project01",
            "directparent": null,
            "children": [
                {
                    "text": "Pergamon Ost",
                    "id": "partproject01",
                    "directparent:": "project01",
                    "children": [
                        {
                            "text": "Bauwerke",
                            "id": "theme01",
                            "directparent": "partproject01",
                            "children": [
                                {
                                    "text": "Tempel des Traianus",
                                    "id": "themeinstance01",
                                    "directparent": "theme01",
                                    "leaf": true
                                },
```

Abbildung 23: Codeauszug aus der *TreeModel.json*, der neben der Projekthierarchie die OpenInfRA-Themeninstanz *Tempel des Traianus* zeigt.

Die beiden beschriebenen Klassen *ThemeInstance* und *TreeModel* bilden somit das Model des GUI-Prototypen pragmatisch aber dennoch vollständig ab (s. Abbildung 24). In einem späteren Produktivsystem können die wieder entfernten Datenstrukturen für Projekte, Teilprojekte, Themen, Attributtypgruppen und Attributtypen wieder ergänzt werden, die dann direkt aus der zukünftigen OpenInfRA-Datenbank befüllt werden können. Die Verwendung von statischen JSON-Dateien ist dann vermutlich nicht mehr erforderlich.

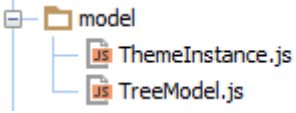


Abbildung 24: Klassen des Models.

Darüber hinaus wäre für die Umsetzung des Rollen- und Rechtesystems und die davon abhängige Ausprägung der Benutzeroberfläche (z. B. Verbergung von GUI-Elementen zur Dateneingabe) noch eine Datenstruktur für den Nutzer erforderlich. Dies wird hier nicht implementiert, da sich dabei die Darstellungsebene mit der Anwendungslogik vermischt, so dass selbst eine prototypische Umsetzung zu aufwändig wäre.

Um vom Model auf die JSON-Dateien zuzugreifen, muss jeweils ein *DataStore* für beide Modelklassen gebildet werden (Abbildung 25) der als ein *Proxy* (Gamma, Helm, Johnson, &

Vlissides, 1995) fungiert. Die Klasse *DataStoreGrid* dient der Anbindung an die Themeninstanzen und *DataStoreTree* versorgt den Projektbaum mit Daten.

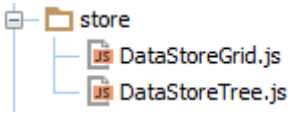


Abbildung 25: *DataStore* Klassen.

Ein *DataStore* enthält die Verbindungsparameter im Attribut *proxy* und optionale Filterfunktionen, die z. B. für die Anzeige eines bestimmten Zweiges des Projektbaums notwendig sind (Abbildung 26). Da für die Datenhaltung von Bäumen andere *DataStores* notwendig sind (s. o. Diskussion über das Attribut *leaf*), erbt die *DataStoreTree* von *Ext.data.TreeStore*, anstatt von *Ext.data.Store*, der für die Themeninstanzen verwendet wird.

```
/*
 * JSON connector to provide access from JS to the JSON files (here the file holding the data for the project explorer).
 * Also contains a filter to only provide the theme instances from the selected project.
 * */
Ext.define('Openinfra.store.DataStoreTree', {
    extend: 'Ext.data.TreeStore',
    requires: 'Openinfra.model.TreeModel',
    model: 'Openinfra.model.TreeModel',
    proxy: {
        type: 'ajax',
        url: 'data/ProjectDataTree.json',
        reader: {
            type: 'json',
            root: 'children'
        }
    },
    filterBy: function (fn,scope){...},
    clearFilter: function () {...}
});
```

Abbildung 26: *DataStoreGrid* als Proxy für den Projektbaum aus der *ProjectDataTree.json* (gekürzt).

Da Änderungen an der OpenInfRA-Datenbasis nicht persistent vorgenommen werden sollen, wird für beide *DataStores* nur der Lesezugriff benötigt.

6.2.2 View-Klassen

Abgeleitet aus dem GUI-Entwurf in Abschnitt 5.3, ergeben sich die in Abbildung 27 gezeigten Klassen für den View.

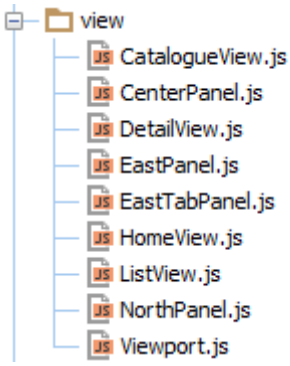


Abbildung 27: View-Klassen.

Die *View*-Hierarchie der in teilt sich wie folgt auf:

- *Viewport*
    - *NorthPanel*
    - *CenterPanel*
        - Bread crumbs
        - *HomeView*
        - *ListView*
        - *CatalogueView*
        - *DetailView*
    - *EastPanel*
        - *EastTabPanel*

Die einzelnen Elemente werden im Folgenden näher erläutert.

Viewport

Der *Viewport* definiert die in Abbildung 15 in Abschnitt 5.3.3 gezeigte Dreigliederung der GUI in *NorthPanel* (Header und Hauptmenüleiste), *EastPanel* (Sidebar) und *CenterPanel* (Hauptinformationsfenster). Diese werden im *Viewport* erzeugt, der als *Composite* (Gamma u. a., 1995) fungiert, und der Startpunkt für den weiteren automatischen Aufbau eine MVC-Ext-JS-GUI ist. Alle weiteren GUI-Elemente können ebenso wieder als *Composite* fungieren, z. B. kann eine Toolbar ein *Composite* für die darauf abgelegten Buttons sein.

NorthPanel

Das *NorthPanel* enthält neben Seitenheader und Toolbar für Impressum, Hilfe, etc., die Haupt-Toolbar, mit der man in OpenInfRA die verschiedenen Sichten auf die Daten auswählen kann (Liste, Katalog, Details, Karte und Kontextbrowser) und Werkzeuge zum Drucken, Exportieren, Einstellungen speichern, Einloggen und Suchen im Datenbestand. Für den Pro-

totyp sind nur Liste, Katalog und Details mit Funktionen hinterlegt. Das *NorthPanel* ist in Abbildung 28 dargestellt.

Abbildung 28: *NorthPanel* mit Seitenheader und Haupt-Toolbar (blau).

CenterPanel

Das *CenterPanel* ist für die Navigation im Datenbestand (über *bread crumbs*), für die Startseite, Listen-, Katalog und Detailansicht, und für das in die Detailansicht integrierte Formular zur Dateneingabe verantwortlich. An der unteren Kante befindet sich eine nicht mit Funktionen hinterlegte Werkzeugleiste zur Paginierung, für den Fall, dass die angezeigten Daten nicht auf einer Seite dargestellt werden können. Auf die verschiedenen Ansichten wird weiter unten eingegangen (*HomeView*, *ListView*, *CatalogueView* und *DetailView*).

EastPanel und EastTabPanel

Im *EastPanel* wird das anzuzeigende Projekt über ein Dropdown-Menü ausgewählt. Die Struktur bis auf Themeninstanzebene wird für dieses Projekt in einem Baum dargestellt, der zur Navigation, Filtern der Auswahl der Objekte genutzt werden kann. Über die Werkzeugleiste an der unteren Kante können neue Datensätze erzeugt, veröffentlicht und gelöscht werden (wobei die letzten beiden nicht mit Funktionalität versehen sind). Eine Darstellung des *EastPanel* findet sich in Abbildung 29.

Abbildung 29: Navigation im OpenInfRA-Projektbaum (mit Werkzeugen zum Datensatz anlegen, veröffentlichen und löschen an der unteren Kante).

Je nachdem was im Projektbaum selektiert wird, ändert sich die Ansicht im *CenterPanel*. Wenn Projekte, Teilprojekte oder Themen ausgewählt sind, wird im *CenterPanel* die Listenansicht mit allen enthaltenen Themeninstanzen angezeigt. Beim Auswählen einer Themeninstanz im Projektbaum, wird die Detailansicht der entsprechenden Themeninstanz im *CenterPanel* angezeigt.

Der *EastPanel* kann beliebig eingeklappt werden. Dadurch vergrößert sich das *CenterPanel* und es können dort mehr Informationen gleichzeitig angezeigt werden. Wird das *EastPanel* wieder ausgeklappt verkleinert sich das *CenterPanel.*

Bread crumbs

Die *bread crumbs* (siehe Abbildung 30) sind an der oberen Kante ins *CenterPanel* integriert. Sie bilden zusammen mit dem Projektbaum des *EastPanel* die Hauptnavigationselemente und zeigen zusätzlich die aktuelle Position im Datenbestand an. Ext JS bietet von sich aus (erstaunlicherweise) keine vorkonfigurierten Elemente für *bread crumbs* an, deshalb wird dieses Element komplett selbst implementiert, inklusive der notwendigen Synchronisation mit dem Projektbaum. Dies macht eine etwas aufwändigere Controller-Implementierung notwendig (siehe Abschnitt 6.2.3).

Abbildung 30: *Bread-crumb*-Leiste (in der Abbildung direkt unter der blauen Menüleiste).

HomeView (Startseite)

Der *HomeView* (dargestellt in Abbildung 31) definiert die Startseite der OpenInfRA-GUI. Sie kann später mit beliebigen Inhalten gefüllt werden. Die Buttons für Listen-, Katalog und Detailansicht werden erst sichtbar, wenn rechts ein Projekt ausgewählt ist.

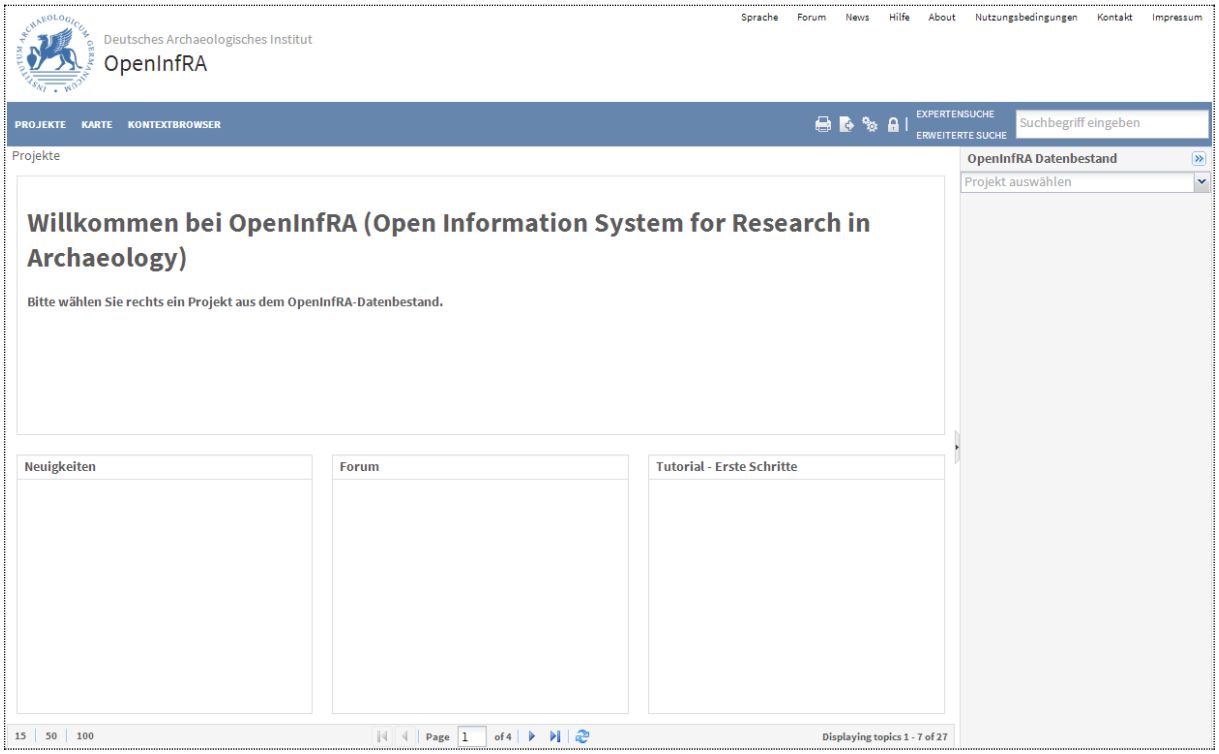


Abbildung 31: Startseite des GUI-Prototyps ohne ausgewähltes Projekt.

ListView (Listenansicht)

Im *ListView* wird die Listenansicht auf eine Auswahl des OpenInfRA-Datenbestands definiert (Abbildung 32) und mit einem *Ext.grid.Panel* umgesetzt. Die Auswahl definiert sich über die selektierte Ebene im Projektbaum und durch Navigation über die *bread crumbs*.

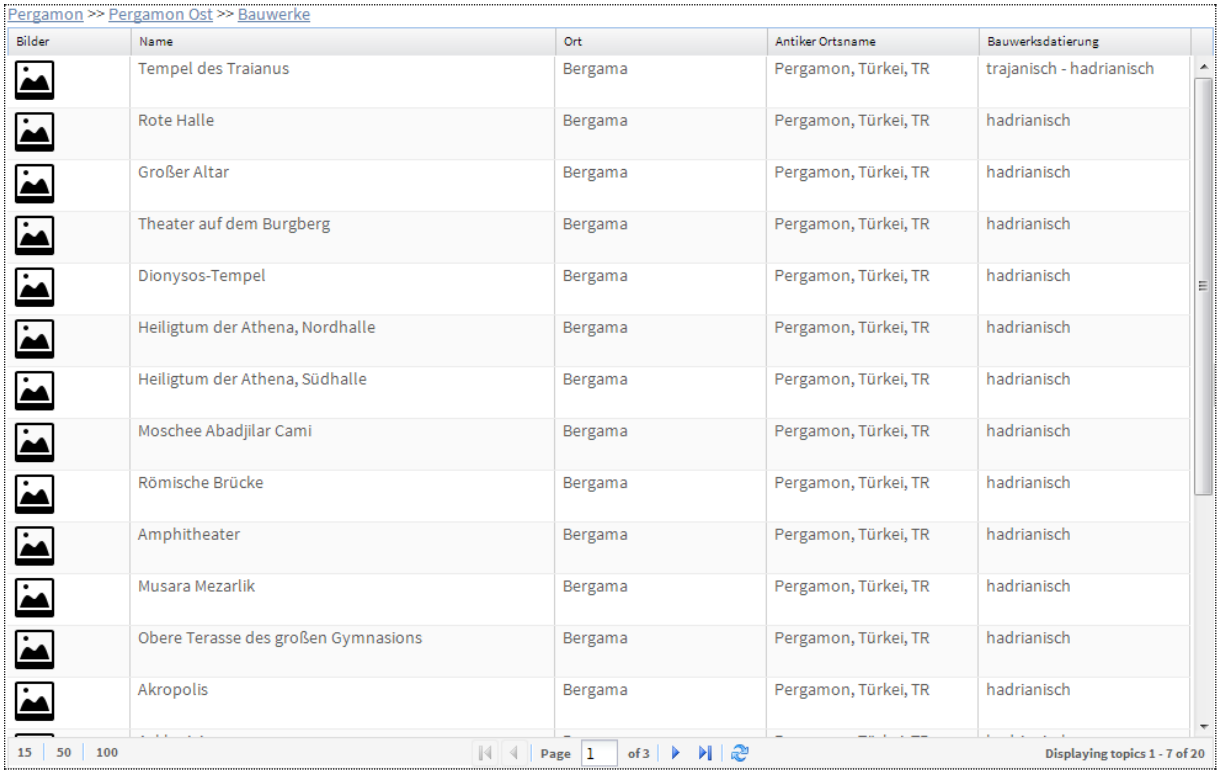


Abbildung 32: Listenansicht aller Bauwerke im Teilprojekt Pergamon Ost.

Die angezeigten Spalten mit den verschiedenen Attributen können beliebig aus- und eingeblendet werden, und die Liste kann nach beliebigen Attributen auf- und absteigend sortiert werden (Abbildung 33).

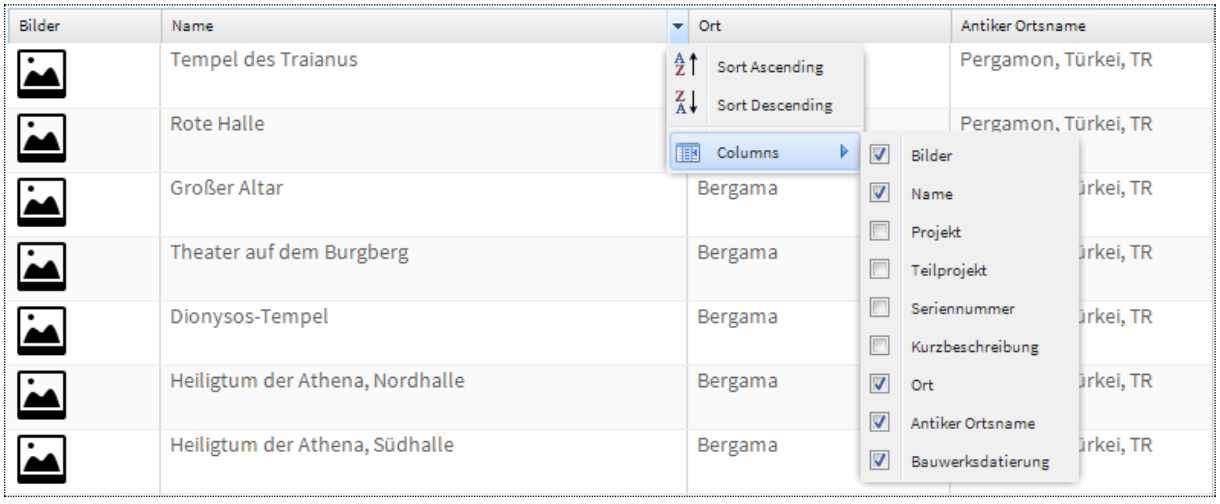


Abbildung 33: Konfigurationsmöglichkeiten in der Listenansicht.

In der Paginierungs-Toolbar (siehe unterer Rand von Abbildung 32) kann die Anzahl der dargestellten Themeninstanzen pro Seite (links) und die Seitennavigation (Mitte) ausgewählt werden. Am rechten Rand wird die Anzahl der aktuell angezeigten Themeninstanzen im Verhältnis zu allen in dem Thema, Teilprojekt oder Projekt vorhandenen Themen (je nachdem was im Projektbau ausgewählt ist) dargestellt. Die Paginierungstoolbar wird nur eingeschränkt mit Funktionalität hinterlegt.

CatalogueView (Katalogansicht)

Die Katalogansicht (Abbildung 34) ist in ihren Anforderungen speziell und kann nicht mit Ext-JS-COTS umgesetzt werden. Sie basiert auf einem angepassten einspaltigen *Ext.grid.Panel*.

Abbildung 34: Katalogansicht aller Bauwerke im Teilprojekt Pergamon Ost.

Die Auswahl der angezeigten Attribute pro Zeile ist nur im Quellcode konfigurierbar (entsprechender Codeauszug in Abbildung 35).

```
columns: [
    {
        flex: 1,
        renderer: function (val, meta, record) {
            var output = '<h3>' + record.raw.text + ' (' + record.raw.Seriennummer + ')' + '</h3>';
            output += '<h4>' + detectProjectForGrid(record.raw.directparent) + ', '
                + detectPartProjectForGrid(record.raw.directparent) + '</h4>';
            output += '<p/ ><b>Bilder</b><br>';
            output += '<img height=100px src=data/images/image_32.png> ';
            output += '<img height=100px src=data/images/image_32.png> ';
            output += '<img height=100px src=data/images/image_32.png>';
            output += '<p/ ><b>Lokalisierung</b><br>';
            output += '<i>Ort:</i> ' + record.raw.Attributtypgruppen[0].Attributtypen[0].Ort;
            output += '<br><i>Antiker Ort:</i> '
                + record.raw.Attributtypgruppen[0].Attributtypen[0].Ort;
            output += '<p/ ><b>Geographische Beschreibung</b><br>';
            output += '<i>Beschreibung:</i> '
                + record.raw.Attributtypgruppen[1].Attributtypen[0].Beschreibung;
            output += '<br><i>Antike Landschaft:</i> '
                + record.raw.Attributtypgruppen[1].Attributtypen[0]['Antike Landschaft'];
            output += '<br><i>Römische Provinz:</i> '
                + record.raw.Attributtypgruppen[1].Attributtypen[0]['Römische Provinz'];
            output += '<br><i>Kulturepoche:</i> '
                + record.raw.Attributtypgruppen[1].Attributtypen[0].Kulturepoche;
            return output;
        }

    }
]
```

Abbildung 35: Im Quellcode verankerte Konfiguration der angezeigten Attribute in der Katalogansicht (Codeauszug aus *CatalogueView.js*).

Die Auswahl der angezeigten Themeninstanzen definiert sich über die selektierte Ebene im Projektbaum und durch Navigation über die *bread crumbs*. Die Paginierungs-Toolbar ist im *ListView* bereits beschrieben.

DetailView (Detailansicht und Dateneingabe)

Die Detailansicht von OpenInfRA-Themeninstanzen erfüllt im GUI-Prototyp zwei Funktionen: Die Ansicht von allen Attributen einer einzelnen Themeninstanz (Abbildung 36), und das Editieren und Hinzufügen von neuen Themeninstanzen über ein Eingabeformular (Abbildung 37 und Abbildung 38). Die Detailansicht existiert deshalb in zwei Ausprägungen, die sich äußerlich zwar ähneln aber unterschiedliche Aufgaben erfüllen.

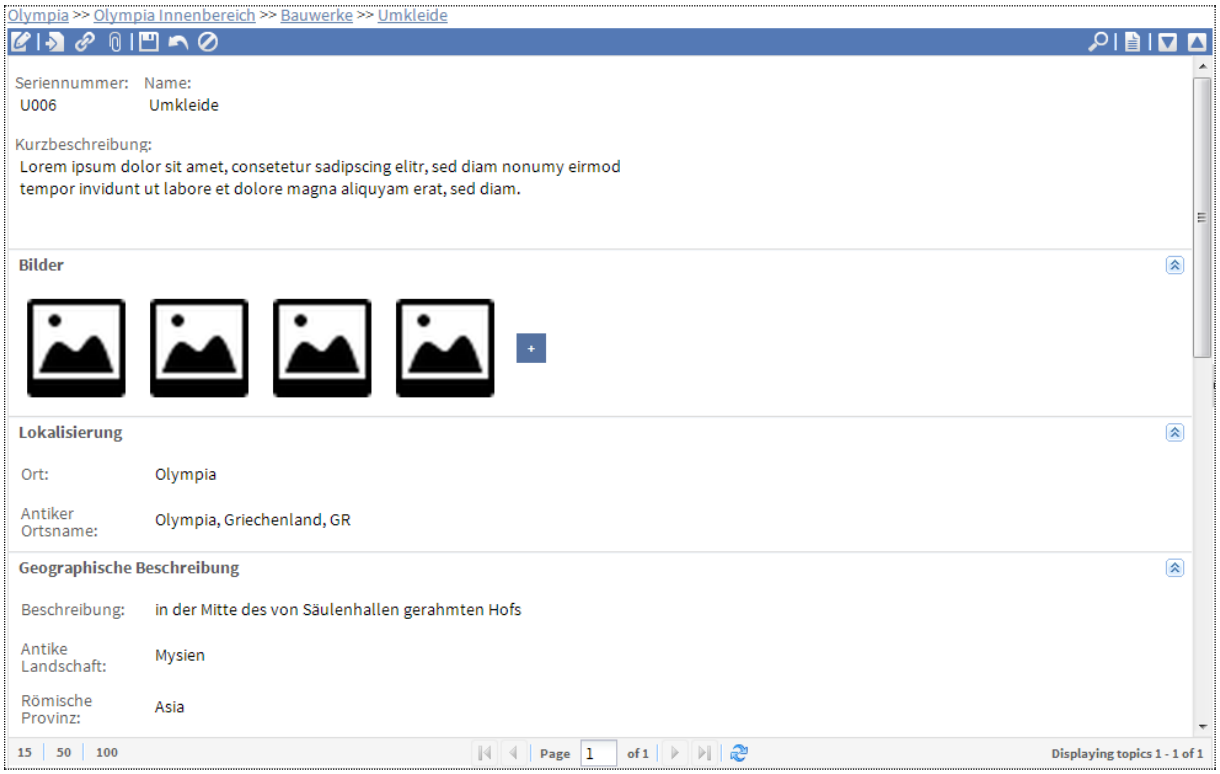


Abbildung 36: Detailansicht der Umkleide im Olympia Innenbereich.

Die einzelnen Attributtypgruppen (in Abbildung 36 sind „Bilder“, „Lokalisierung“ und „Geographische Beschreibung“ sichtbar) können einzeln über die Buttons am rechten Rand in der blauen Toolbar ein- und ausgeklappt werden. Über die beiden Pfeil-Buttons oben rechts können alle Attributtypgruppen auf einmal geöffnet bzw. geschlossen werden.

Mit einem beliebigen Klick ins Formular wird der Editiermodus (Abbildung 37) gestartet. Es werden diejenigen Felder rot markiert, die nicht oder nicht korrekt ausgefüllt werden. Zurzeit wird nur abgeprüft ob das Feld gefüllt ist oder nicht; der eingegebene Inhalt wird nicht validiert. Weitere Bilder oder Literaturquellen können über die jeweiligen blauen „Plusbuttons“ hinzugefügt werden (nicht mit Funktionalität hinterlegt). Nach dem Beenden des Editierens kann der Zustand über das Diskettensymbol in der oberen blauen Toolbar gespeichert werden. Der Zustand des Formulars wechselt dann wieder in die o. a. Detailansicht.

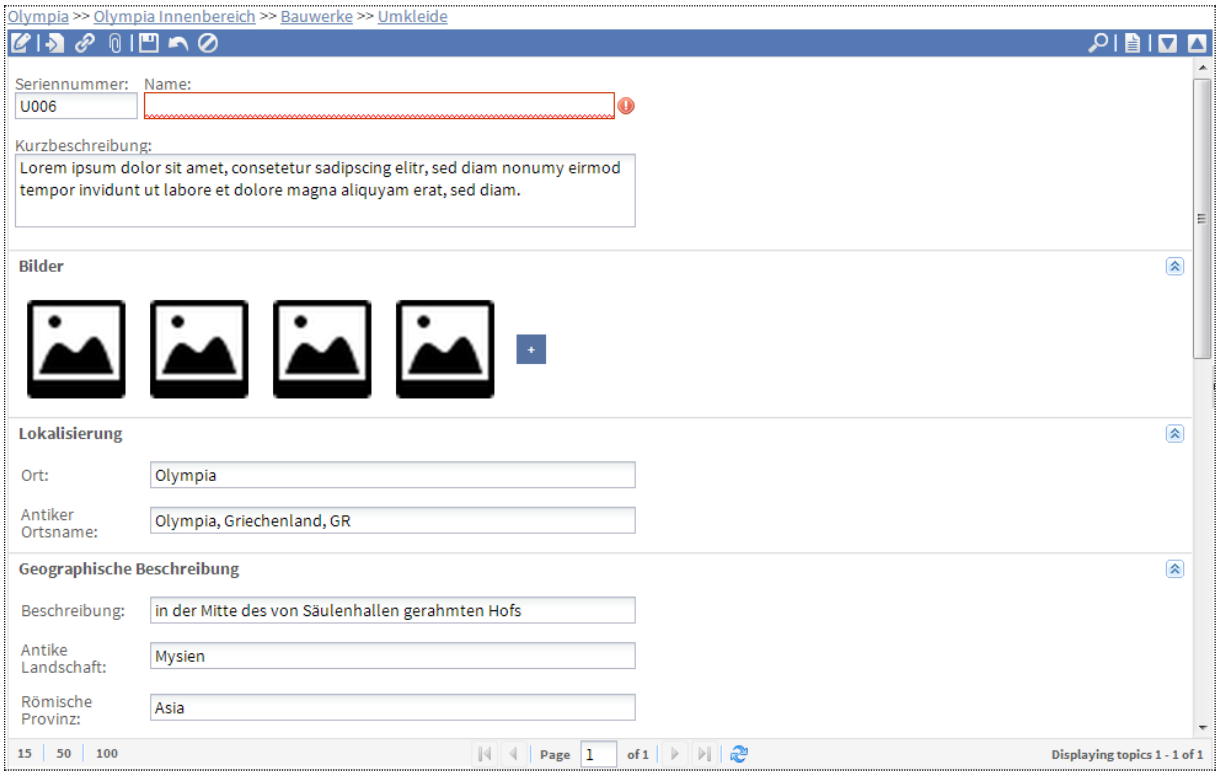


Abbildung 37: Editieransicht der Umkleide im Olympia Innenbereich (die rote Markierung zeigt an, dass das Pflichtfeld „Name“ noch nicht ausgefüllt ist).

Über den linken unteren Button im *EastPanel* können neue Themeninstanzen für das im Projektbaum ausgewählte Thema erzeugt werden. Es öffnet sich ein leeres Formular in der Detailansicht mit rot markierten Pflichteingabefeldern (Abbildung 38). Die Interaktion mit dem Formular erfolgt analog zum oben beschriebenen Editiermodus.

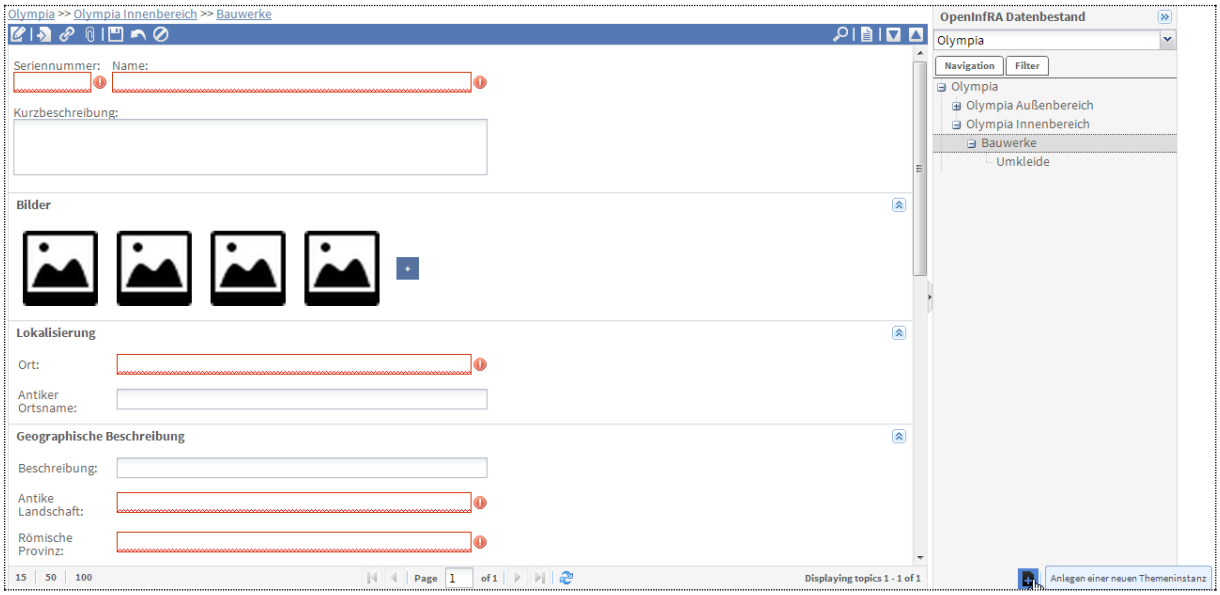


Abbildung 38: Anlegen eines neuen Bauwerks im Olympia Innenbereich über den linken unteren Button im *EastPanel* (die roten Markierungen zeigen an, welche Pflichtfelder noch ausgefüllt werden müssen).

6.2.3 Controller-Klasse

Im Entwurf (siehe Abschnitt 5.3) werden folgende Controller vorgesehen, die korrespondierende Bereiche der GUI-Oberfläche abdecken (Abbildung 16):

- *ListViewController*
- *CatalogueViewController*
- *DetailViewController*
- *DetailEditController*
- *NavigationController*
- *BrowseController*
- *MenuBarController*
- *PaginierungsController*
- *EditierController*

Viele dieser Controller werden bereits implizit durch die verwendeten GUI-Elemente der Ext-JS-Bibliothek zur Verfügung gestellt. Deshalb wird für die Implementierung nur noch eine *Controller*-Klasse als *Observer* (Gamma u. a., 1995) geschrieben (Abbildung 39).

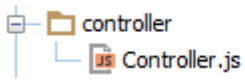


Abbildung 39: *Controller*-Klasse.

Besonders für die Teile der GUI, die nicht auf vordefinierten Ext-JS-Elementen aufgebaut sind, z. B. den *bread crumbs* und dem Editiermodus in der Detailansicht werden größere Teile des Quellcodes im *Controller* eingefügt.

```
/*
* Fill bread crumb in Center Panel (below main menu)
* with content according to the selected Item in the East Panel (project selector)
* which is given as 'record'.
* */
function fillBreadcrumb(record) {

    breadcrumbArray.push(record);

    if (record.parentNode.data.text != "Root") {
        fillBreadcrumb(record.parentNode);
    } else {
        breadcrumbArray = breadcrumbArray.reverse();
        Ext.each(breadcrumbArray, function(item) {
            breadcrumbTextNew += '<a href=\"#\" onclick=\"onClickBreadcrumb(\''
                + item.internalId + '\')\">' + item.data.text + '</a>  &gt;&gt;  ';
        });

        breadcrumbTextNew = breadcrumbTextNew.substring(0, breadcrumbTextNew.length - 12);
        Ext.getCmp('breadCrumbNew').body.update(breadcrumbTextNew);
    }
}

// When item in bread crumb is clicked select respective item in the East Panel (project selector).
function onClickBreadcrumb(record_id) {
    var record = Ext.getCmp('treepanel').getStore().getNodeById(record_id);
    Ext.getCmp('treepanel').getSelectionModel().select(record);
}
```

Abbildung 40: Codeauszug aus der *Controller*-Klasse zur Synchronisation der *bread crumbs* mit dem Projektbaum.

Der Codeauszug in Abbildung 40 zeigt zwei Methoden, die aufgerufen werden, sollte der Benutzer die Auswahl im Projektbaum im *EastPanel* ändern oder einen *bread crumb* anklicken. Die obere Methode *fillBreadcrumb* wird aufgerufen, wenn sich die Auswahl im Projektbaum ändert und somit die *bread crumbs* neu erzeugt werden müssen, um die aktuelle Auswahl ebenso korrekt anzuzeigen. Dies ist über eine Rekursion gelöst, also eine Methode, die sich so lange selber aufruft, bis eine Abbruchbedingung zutrifft. In diesem Fall wird ausgehend von der Auswahl im Projektbaum die Ebenen nach oben gegangen und damit Objekte in das *bread crumb* Array gespeichert (*if*-Bedingung), bis das Wurzelelement (das eigentliche Projekt) erreicht ist (*else*-Bedingung). Wenn das Wurzelelement erreicht ist, wird das gefüllte Array in der Reihenfolge komplett umgedreht (*breadcrumbArray.reverse()*), so dass das Pro-

jekt nun an erster Stelle steht und das eigentlich angewählte Objekt am Ende. Das Array wird dann über *Ext.each(breadcrumbArray, […])* iteriert und die Namen der Objekte in die *bread crumbs* geschrieben. Dieses Vorgehen ist nötig, da Ext JS dafür keine GUI-Elemente bereitstellt.

Die Methode *onClickBreadcrumb* ist das Gegenstück zur gerade beschriebenen Methode. Sie registriert den Klick im *bread crumb* und wählt das entsprechende Projekt im Projektbaum aus. Da der Projektbaum durch die Ext-JS-Bibliothek bereitgestellt wird, ist der Aufruf sehr viel weniger komplex.

Der restliche *Controller* besteht aus Hilfsmethoden um aus der flachen Datenstruktur der Themeninstanzen (beschrieben in Abschnitt 6.2.1) über das Attribut *directparent* die jeweiligen Teilprojekt und Projekte zu identifizieren und Methoden die zum (initialen) Befüllen und Leeren der Dateneingabeformulare notwendig sind (*fillDetailView* und *emptyDetailView)*. Ebenso werden für alle Formularfelder ein *Listener* hinzugefügt. Dies ist notwendig um Klicks in der Detailansicht zu registrieren, die das Formularfeld in den Editiermodus schalten.

6.3 Umsetzung des Styleguides des Deutschen Archäologischen Instituts und Berücksichtigung von Referenzimplementierungen

Für die Entwicklung des eigentlichen Designs stehen zwei Quellen zur Verfügung: Der DAI-Styleguide (Anhang C) und die bisher auf einer Weiterentwicklung der Bootstrap-Bibliothek (Archaeostrap) basierten DAI-Referenzimplementierungen.

Zwischen beiden Quellen bestehen teilweise erhebliche Diskrepanzen im Design, da der Styleguide auf das Design einer normalen Webseite abzielt und nicht auf komplexere Anwendungen wie z. B. OpenInfRA oder andere DAI-Produkte. Eine Entscheidung für eine der beiden Quellen ist also erforderlich.

Da die GUI-Implementierung schon zu weit fortgeschritten ist, kann Archeostrap als Bibliothek nicht mehr berücksichtigt werden. Um eine spätere Umsetzung mit Archeostrap zu unterstützen, orientiert sich die GUI im Design an den Referenzimplementierungen.

Diese beachten ihrerseits hauptsächlich die DAI-Styleguide-Vorgaben zu Farben, Fonts und einigen Standardinteraktionswerkzeugen. Für den GUI-Prototyp sind diese Aspekte also ebenso relevant und fließen als Teil des DAI-Styleguides mit in die Entwicklung ein.

Der Hauptunterschied zwischen den DAI-Referenzimplementierungen und dem GUI-Prototyp liegt in der Art und Weise des Aufbaus der Seiten. Ext JS, und somit auch der Prototyp, ist so konzipiert, dass Clients mehr oder weniger bildschirmfüllend implementiert werden. Sollte der Seiteninhalt nicht auf den vorhandenen Platz passen, wird das entsprechende Panel scroll-bar geschaltet. Die Referenzimplementierungen sind so implementiert, dass die ganze

Seite gescrollt wird; dabei bleibt die Menüleiste am oberen Bildschirmrand angedockt. Eine Umsetzung des Verhaltens der Referenzimplementierungen kann deshalb für den GUI-Prototyp mit Ext JS nicht umgesetzt werden. Im Vergleich zur Referenzimplementierung wird der Seitenheader zusätzlich für die GUI entsprechend kleiner dargestellt, damit die nicht für die Darstellung nutzbare Bildschirmfläche minimiert werden kann (siehe Designvergleich in Abbildung 43 und Abbildung 44).

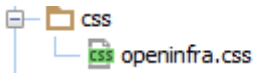


Abbildung 41: Die für den GUI-Prototyp angepassten CSS finden sich in der *openinfra.css*.

Die weiteren Anpassungen der Ext-JS-CSS finden sich in der *openinfra.css* (Abbildung 41). Der Unterschied zwischen den Ext-JS- und den anpassten CSS kann man im Vergleich der Abbildung 45 mit der Abbildung 46 erkennen (Abschnitt 6.5). Die Umsetzung von ausschlaggebenden Designelementen erfordert oft sehr verschachtelte CSS-Anpassungen.

Bei den Anpassungen ist aufgefallen, dass die vom DAI-Styleguide und den Referenzimplementierungen verwendete Schriftart „Source Sans Pro“ beispielsweise auf Linux-Betriebssystemen nicht vorhanden ist. Um dennoch ein möglichst ähnliches Aussehen zu generieren mussten alternative Schriftarten gesucht und konfiguriert werden (siehe Codeauszug in Abbildung 42). Für alle weiteren CSS-Anpassungen sei auf den Quellcode (Anhang C) verwiesen.

```
/* Change font family to 'Source Sans Pro' for all text and provide back fall for systems w/o this font. */
* {
    font-family: 'Source Sans Pro', 'Open Sans', sans-serif !important;
}
```

Abbildung 42: Konfiguration der Basisschriftart für die GUI.

Abschließend kann durch einen Designvergleich der Seitenheader zwischen GUI-Prototyp (Abbildung 43) und der Referenzimplementierung Arachne (Abbildung 44) erkannt werden, dass sich das Design nur in kleineren Einzelheiten unterscheidet, ein Nachbau der Referenzimplementierungs-CSS also grundsätzlich möglich ist, aber einen hohen Aufwand erfordert.

Abbildung 43: Header der OpenInfRA-GUI.

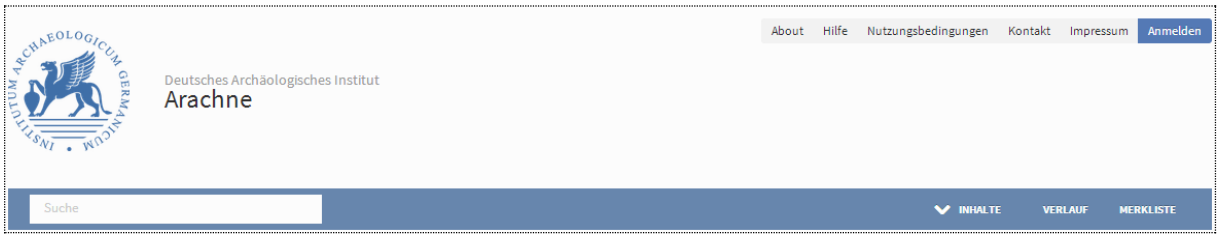


Abbildung 44: Header der DAI-Referenzimplementierung für das System Arachne.

6.4 OpenInfRA-Anforderungen im Prototyp

Um einen Überblick über die im Prototyp umgesetzten Funktionalitäten zu erhalten, wurden die für den Diskursbereich relevanten Anforderungen bzgl. ihrer Verwendung im Prototyp gekennzeichnet. Sie sind in den Bereichen BN, SEA und UAB des erweiterten Anforderungskatalogs (s. Anhang C – Anforderungskatalog Prototyp V2) in den Spalten „Umgesetzt in GUI-Prototyp V2“ und „Art der Umsetzung in GUI-Prototyp V2“ zu finden.

Explizite Anforderungen zur Usability finden sich in der Anforderungskategorie UAB. Neben den BITV-Anforderungen, die in dieser Arbeit nicht betrachtet wurden, sind nur wenige so allgemein formulierte Anforderungen zu finden, dass sie für die Umsetzung im Prototyp nur sehr unscharf bewertet werden können. Allerdings ist zu erwähnen, dass viele Vorgaben zur Usability oft implizit in der Beschreibung oder der Bemerkung zu einer funktionalen Anforderung enthalten sind. Diese konnten im Rahmen dieser Arbeit allerdings nicht berücksichtigt werden.

6.5 Entwicklungsphasen und Nutzerbefragung

Die Entwicklung des Prototyps fand in zwei Entwicklungsphasen statt, mit einer informellen Nutzerbefragung innerhalb des OpenInfRA-Projektteams nach der Fertigstellung der ersten Prototyp-Version (die Zugangsdaten finden sich in Anhang A und B).

In der ersten Phase lag der Fokus auf der Entwicklung des Layouts der Benutzerschnittstelle und der, für die unterschiedlichen Datenansichten benötigten Basisfunktionalitäten. Entsprechend des Entwurfs wurden Elemente, Icons und Menüs im Layout platziert und die Logik

der Datenansichten aufgebaut. Die Entwicklungsarbeiten fanden unter Verwendung des Standard-CSS von Ext JS statt.

Nach Abschluss der ersten Entwicklungsphase (http://magdalinski.eu/mscmag) wurde eine informelle Nutzerbefragung (s. Anhang B) zum GUI-Prototyp innerhalb des OpenInfRA-Teams durchgeführt. Hier ein Auszug aus den Kommentaren und Verbesserungsvorschlägen des Feedbacks zur Umfrage:

- Mauszeiger in Pointer verwandeln bei Mouse-Over in Listen- und Katalogansicht (für verbesserte Visualisierung der Interaktionsmöglichkeit)
- Pfeilsymbole für Öffnen und Schließen der Attributtypgruppen in Detailansicht (für verbesserte intuitive Verständlichkeit) verwenden
- Mauszeiger in Pointer verwandeln bei Mouse-Over über vorhandene Attributwerte in der Detailansicht zum Starten des Editiermodus (für verbesserte Visualisierung der Interaktionsmöglichkeit)
- Editiermodus sollte auch über Klick außerhalb der Formularfelder wieder beendet werden können. Absichern durch Abfrage, ob Ändern gespeichert werden soll.
- Fehlende Funktion für das Abbrechen von Editiervorgängen hinzufügen
- Kategorisierung und Gruppierung der Werkzeuge der Editiertoolbar verbessern
- Werkzeugleiste der Sidebar günstiger oben platzieren, evtl. in Editiertoolbar integrieren
- Links im Header nicht ästhetisch, evtl. Suche in den Header integrieren
- Hauptmenüleiste nicht ästhetisch, zu starke Mischung von Schriften, Symbolen und Funktionen, evtl. Suche in den Header integrieren
- Bearbeitungsmodus in der Detailansicht evtl. mehrspaltig konzeptionieren
- In allen Sichten Fotos und Zeichnungen markanter oben platzieren
- Der Umfang und die Platzierung von Funktionalitäten sind in der Detailansicht im Modus Anzeigen, Suchen und Bearbeiten so unterschiedlich, dass diese auch konzeptionell von der Darstellung her getrennt werden sollten
- Funktionalitäten wie Suchen und Bearbeiten in der Detailansicht im Bearbeitungsmodus in jede Attributtypgruppe integrieren

In der zweiten Entwicklungsphase des Prototyps (http://magdalinski.eu/mscmag_v2) stand die Umsetzung der Design- und Gestaltungsvorgaben des DAI-Styleguides und der DAI-Referenzimplementierungen im Mittelpunkt. Dafür wurden umfangreiche Anpassungen am CSS des GUI-Prototyps durchgeführt. Auf Basis des Nutzerfeedbacks zur ersten Version des

GUI-Prototyps wurden Entwurfskonzepte (z. B. zur Ausprägung und zum Verhalten des Bearbeitungsmodus in der Detailansicht) grundlegend überdacht (s. Kapitel 7) und folgende Änderungen am Prototyp durchgeführt:

- Pfeilsymbole wurden für Öffnen und Schließen der Attributtypgruppen in der Detailansicht eingeführt
- Buttons für das Abbrechen und „Rückgängig machen“ von Editiervorgängen wurden hinzugefügt
- Kategorisierung und Gruppierung der Werkzeuge der Editiertoolbar wurden verbessert
- Hauptmenüleiste wurde nach ästhetisch Gesichtspunkten und den DAI-Vorgaben angepasst
- In allen Sichten wurden Fotos und Zeichnungen markanter oben platziert

Die Evolution des Prototyps ist auch in Abbildung 45 (1. Phase) und Abbildung 46 (2. Phase) gut zu erkennen.

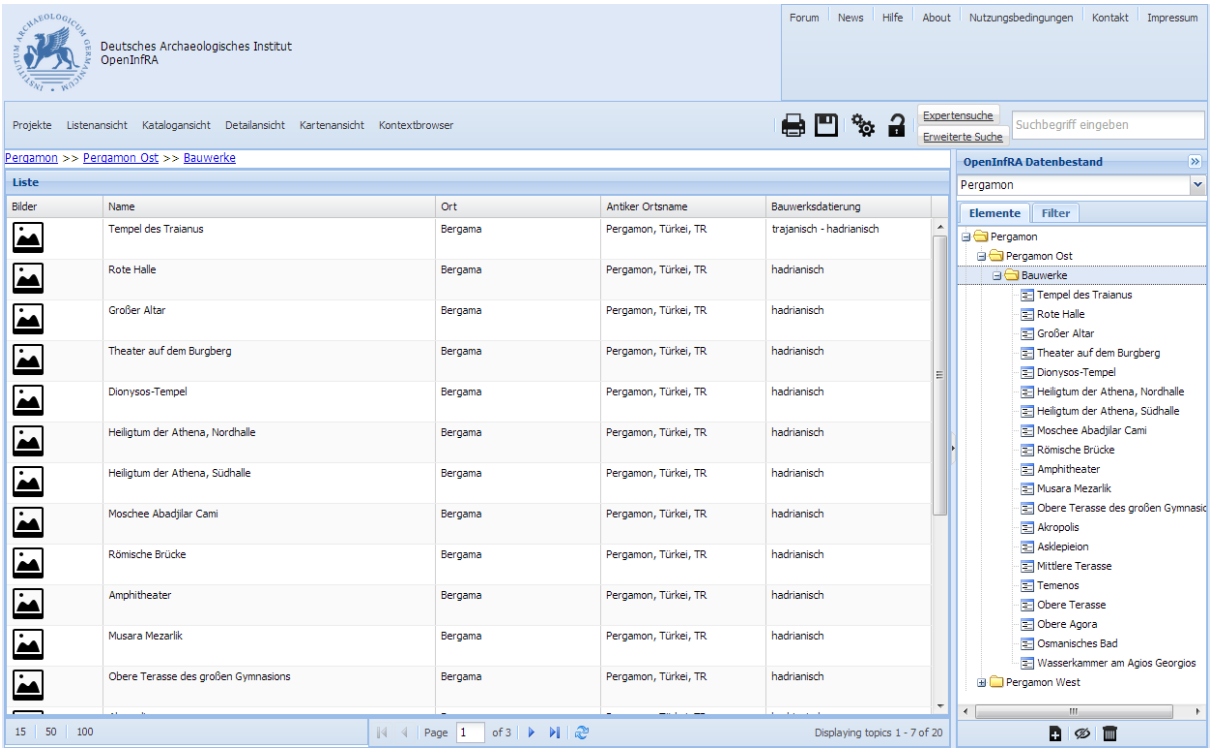


Abbildung 45: Der erste GUI-Prototyp mit der Listenansicht aller Bauwerke im Teilprojekt Pergamon Ost.

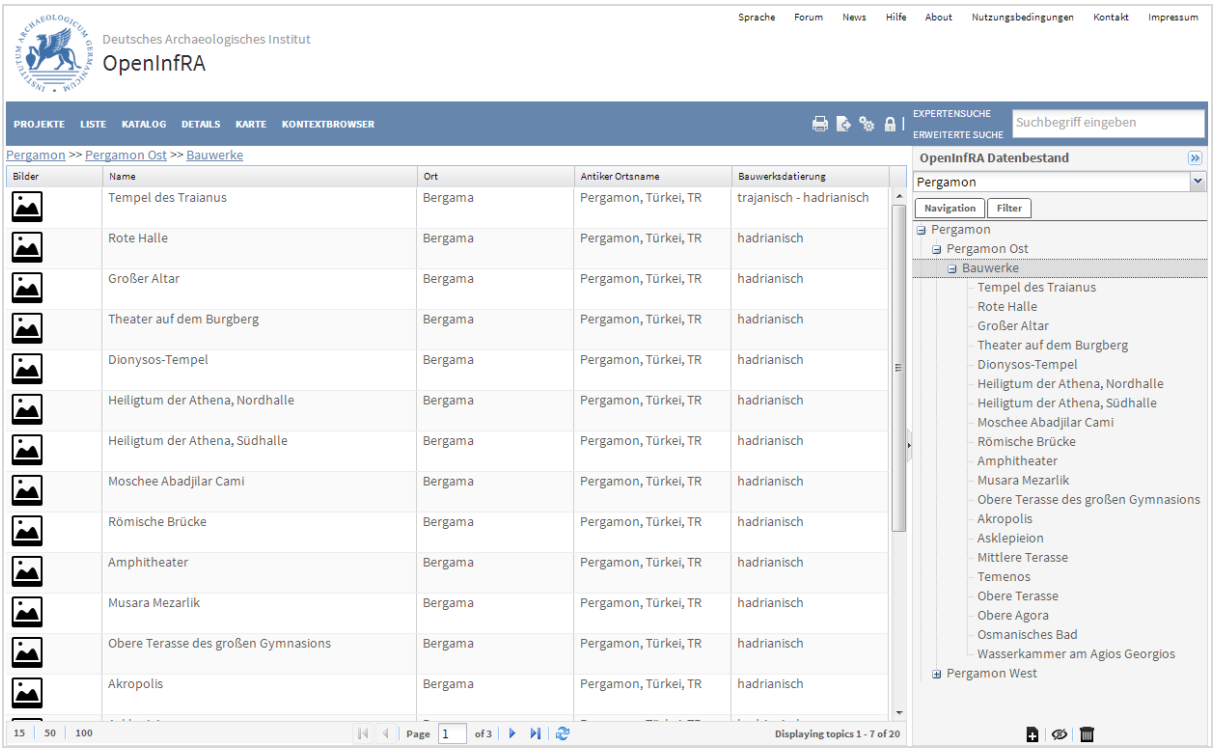


Abbildung 46: Der zweite GUI-Prototyp mit der Listenansicht aller Bauwerke im Teilprojekt Pergamon Ost.

7 Zusammenfassung und Ausblick

Da für das Projekt OpenInfRA zwar schon einige isolierte Prototypen zur Verfügung stehen, ein integrierender Prototyp aber noch fehlt, ist in der vorliegenden Arbeit ein entsprechender GUI-Prototyp entworfen und implementiert worden. Integrierend bedeutet dabei, dass er sowohl die wichtigsten funktionalen Anforderungen eines archäologischen Informationssystems (z. B. verschiedene Sichten auf die Daten und Dateneingabe), als auch Anforderungen an die Usability und das Aussehen der Nutzeroberfläche umsetzt. Die Implementierung der vollen Funktionalität ist dabei nicht erforderlich, es wurden wichtige Teilbereiche ausgewählt. Darüber hinaus ist ein Seitenlayout erarbeitet worden, dass sich auf die Vorgaben des DAI stützt und das Corporate Design des DAI umsetzt. Der aktuelle Stand von Wissenschaft und Technik in den Bereichen Usability von Webanwendungen, Prototyping und Webdesign dient als Grundlage für den Entwurf und die Entwicklung des Prototyps. Dieser ist online verfügbar, liegt im kommentierten Quellcode vor und kann anhand des beigefügten Beispiel-Workflows exploriert werden. Er basiert auf den Technologien HTML, CSS und JavaScript und ist mit dem Framework Ext JS umgesetzt worden. Der Prototyp ist durch eine informelle Befragung im Projekt nach der ersten von zwei Entwicklungsphasen getestet worden. Verschiedene Vorschläge des Feedbacks sind umgesetzt worden.

Bei der Analyse der Grundlagen zu Usability und Webdesign und der im Grobkonzept OpenInfRA (OpenInfRA, 2013) spezifizierten Anforderungen sind einige Aspekte aufgefallen, die für den weiteren Projektverlauf relevant sind und beachtet werden sollten. Zum einen könnte Usability gerade im Anforderungskatalog sehr viel umfangreicher berücksichtigt werden als bisher im Grobkonzept geschehen. Schmidtke & Jastrzebska-Fraczek (2013) liefern eine Vielzahl von sehr gut aufbereiteten und übersichtlichen Anforderungen an die Usability. Vieles davon erscheint selbstverständlich und ist vielleicht deshalb bisher nicht explizit berücksichtigt worden. Die „gute“ Gebrauchstauglichkeit und Bedienbarkeit der Anwendung ist jedoch für die Akzeptanz durch die Nutzer essentiell – und sollte daher auf jeden Fall sichergestellt sein. Es erscheint allerdings nicht sinnvoll im großen Umfang feingranulare Usability-Anforderungen zu formulieren und im Anforderungskatalog zu ergänzen. Zum einen existieren diese bereits als Soll-Vorgaben von Schmidtke & Jastrzebska-Fraczek (2013b) und zum anderen würde diese Menge an Anforderungen (s. Abschnitt 2.4) den Anforderungskatalog „sprengen“. Die Forderung nach der Erfüllung dieser feingranularen Anforderung bei der Implementierung soll durch die Aufnahme weiterer relevanter Normen in den Anforderungskatalog sichergestellt werden, da in diesen die Anforderungen zum Teil implizit aber auch explizit enthalten sind. Aus den vorhandenen feingranularen Soll-Vorgaben (Schmidtke & Jastrzebska-Fraczek, 2013b) können vielmehr Checklisten extrahiert und für die Erstellung von konkreten Testfällen genutzt werden, um die Entwicklung der Benutzerschnittstelle zu unterstützen und letztendlich abzunehmen.

Der DAI-Styleguide und die DAI-Referenzimplementierungen (s. Abschnitt 5.2.3) liefern umfangreiche Vorgaben bezüglich des Layouts und der Gestaltung für den GUI-Prototyp und später auch für das OpenInfRA-Produktivsystem. Im Sinne eines Corporate Designs des Deutschen Archäologischen Instituts ist dies sicher sehr sinnvoll. Für die Umsetzung spezieller Anforderungen bzgl. Bedienbarkeit, Funktionsumfang und Präsentation von Informationen eines webbasierten Informationssystems können sie allerdings auch eine Einschränkung darstellen und die optimale Umsetzung der Anforderungen evtl. erschweren, sowohl konzeptionell als auch technologisch (falls man an die Technik der Referenzimplementierungen gebunden ist). Zum Teil werden Einzelfallentscheidungen notwendig sein, um das Verhältnis zwischen vorgegebenem Design und Layout und notwendigen Usability-Anforderungen zu evaluieren.

Was die Gestaltung eines GUI-Prototyps angeht, gibt es keine genau richtige oder genau falsche Darstellung (Preim & Dachselt, 2010). Unterschiedliche Benutzergruppen haben unterschiedliche Ansprüche an eine Anwendung und andere Vorstellungen von deren Ausprägung. Selbst innerhalb von Nutzergruppen variieren die individuellen Erwartungen. Daher stellt der im Rahmen dieser Arbeit entwickelte Prototyp nur einen ersten Schritt dar. Es ist sinnvoll alternative Prototypen zu erstellen und diese mit Nutzern und Experten zu evaluieren und weiter zu entwickeln. Da sich im Zusammenhang mit der ausstehenden Ausschreibung des OpenInfRA-Projektes einige markante Änderungen der Anforderungen gerade auch bzgl. des Projekt-Clients ergeben haben, sollte dieser in einem nächsten Schritt auf den aktuellen Stand der Spezifikation erweitert werden. Außerdem sollten Nutzerevaluierungen zu aktuellen und zukünftigen Versionen des Prototyps durchgeführt werden.

Bei der Implementierung des Prototyps ist die Verwendung des Ext-JS-Frameworks (oder alternativer JS-Frameworks) grundsätzlich der reinen HTML-Entwicklung vorzuziehen, da viele Konzepte wie beispielsweise das MVC-Konzept sehr einfach umgesetzt werden können. Darüber hinaus kann in Ext JS aus einer Fülle von optimierten GUI-Elementen gewählt werden, die, wenn sie keine individuellen Erweiterungen benötigen, ohne großen Aufwand zusammengesetzt werden können. Oft werden bereits alle notwendigen Listener und CSS mitgeliefert. Die Verwendung von vorkonfigurierten Elementen funktioniert sehr gut, hat aber gelegentlich Nachteile. Teilweise sind z. B. Beschriftungen nur auf Englisch verfügbar, und die Anpassung von Icons an alternative Farbschemata ist nur schwer möglich. Sobald die durch Ext JS vorkonfigurierten Pfade verlassen werden, werden Entwicklungen schnell komplex (siehe Beschreibung der *bread crumbs* inklusive ihrer Synchronisation mit dem Projektbaum in Abschnitt 6.2.3).

Die Eingabe von Attributwerten soll in OpenInfRA weitgehend über vordefinierte Indexlisten und Drop-Down-Menüs erfolgen. Dies beschleunigt die Dateneingabe und verhindert Fehleingaben. Diese Funktionalität ist nicht umgesetzt worden. Die Validierung der Formularfelder nach Syntax und Inhalt ist in diesem Sinne ein ebenso wichtiges Merkmal und unbedingt

für die Weiterentwicklung zu berücksichtigen. Zurzeit wird nur geprüft, ob Pflichteingabefelder gefüllt sind, ohne die Eingabe weiter zu analysieren. Details dazu finden sich in Abschnitt 6.2.2.

Es hat sich gezeigt, dass die Datenhaltung über JSON-Dateien auf der einen Seite syntaktisch sehr einfach, auf der anderen Seite aber nur mäßig für die Nutzung eines komplexeren Datenmodells geeignet ist. Hierarchien können zwar abgebildet werden, sind allerdings nur schwer nutzbar, da bestimmte GUI-Elemente, wie z. B. der Projektbaum daraus nicht erzeugt werden können. Eine Suche nach Objekten mit einer bestimmten ID ist nur dann möglich, wenn die unterschiedlichen Hierarchieebenen gleich benannt sind. Hier wurden pragmatische Anpassungen durchgeführt, die für einen Produktiveinsatz nicht empfohlen werden können. Details dazu finden sich in Abschnitt 6.2.1.

Wenn man den Arbeitsaufwand zwischen Model-, View- und Controller-Klassen vergleicht, fällt auf, dass die Anpassung der Model-Klassen am meisten Zeit gekostet hat. Vom Aufwand her folgen danach Implementierungen abseits der vordefinierten Ext-JS-Elemente inklusive zugehöriger Controller. Der Aufbau und die Konfiguration der GUI konnten dank vordefinierter Ext-JS-Elemente vergleichsweise schnell umgesetzt werden. Die Anpassung an das CSS der DAI-Referenzimplementierungen war zwar mühsam, aber im Vergleich nicht sehr zeitaufwändig.

Unterschiede zwischen den DAI-Referenzimplementierungen und dem GUI-Prototyp liegen zum Teil an der unterschiedlichen grundsätzlichen Konzeption von Archeostrap und Ext JS (siehe Abschnitt 6.3). Manche Konzepte der Referenzimplementierungen sind deshalb nicht ohne weiteres mit Ext JS umsetzbar. Grundsätzlich ist aufgefallen, dass der DAI-Styleguide und damit auch die Referenzimplementierungen auf einer Schriftart basieren, die z. B. auf Linuxsystemen nicht notwendigerweise vorhanden ist und somit das einheitliche Design verletzt - wenn auch nur geringfügig (siehe Abschnitt 6.3).

Für den vorliegenden Prototyp waren viele Entwicklungen schon zu weit fortgeschritten, um nachträglich noch eine vollständige Anpassung an den DAI-Styleguide vorzunehmen oder den GUI-Prototyp mit Archeostrap umzusetzen, um das gleiche Framework der Referenzimplementierungen zu verwenden. Grundsätzlich wird empfohlen, über eine nachträgliche Umsetzung mit Archeostrap nachzudenken, da dies die DAI-Implementierungen vereinheitlichen kann. Dabei kann der Aufwand für den Aufbau der GUI mit Archeostrap nicht abgeschätzt werden, da nicht bekannt ist, wie gut die in Archeostrap vordefinierten GUI-Elemente zusammengebaut werden können. Dies funktioniert, wie bereits erwähnt, in Ext JS sehr gut. Zusätzlich bietet Ext JS eine sehr gute Dokumentation mit vielen verständlichen Beispielen und eine sehr hilfsbereite Nutzergemeinschaft. Auch diese Faktoren sollten bei einer Entscheidung für oder gegen Archeostrap oder ein alternatives Framework berücksichtigt werden.

Glossar

Benutzermodell	Annahmen und Angaben über die Nutzer einer Software in der Software-Entwicklung. („Benutzermodell“, 2013)
Corporate Design	Eindeutig zuordenbares grafisches Erscheinungsbild eines Unternehmens (Corporate Design = Unternehmensidentität), geprägt durch eindeutige Wortzeichen (Firmenschriftzug) und Bildzeichen (Logo) und deren Kombination. Verwendung auf Geschäftspapieren, Werbemitteln, Verpackungen, Internetauftritten und in der Produktgestaltung. („Corporate Design“, 2013)
Gestaltung	Hier verstanden als Synonym für Design im Kontext von Webseiten.
Layout	Im Kontext von Webseiten, umfasst Layout hier den inhaltlichen und strukturellen Aufbau inkl. des Designs.
Stakeholder	Person oder Gruppe, die ein berechtigtes Interesse am Verlauf oder Ergebnis eines Prozesses oder Projektes hat. („Stakeholder“, 2013)
User Experience	Auch als Nutzererlebnis bezeichnet, d. h. beschreibt wie der Nutzer eine Webseite erlebt.
Wireframe (Webseite)	Auch Seitenschema oder Bildschirm-Blueprint. Es handelt sich um den groben Aufbau einer Webseite bzw. die grobe Anordnung von Elementen auf der Seite für einen bestimmten Zweck. („Website wireframe“, 2013)

Quellenverzeichnis

Ashworth, S., & Duncan, A. (2012). *Ext JS 4 Web application development cookbook*. Birmingham, UK: Packt Pub.

Balzert, H. (2008). *Software-Management: Lehrbuch der Software-Technik*. Heidelberg: Spektrum Akademischer Verlag.

Benutzermodell. (2013). In *Wikipedia*. Abgerufen von http://de.wikipedia.org/w/index.php?title=Benutzermodell&oldid=115287946 (10.12.2013)

BGG. (2002). Gesetz zur Gleichstellung behinderter Menschen (Behindertengleichstellungsgesetz - BGG). (BGBl. I S. 1467, 1468). Abgerufen von http://www.gesetze-im-internet.de/bgg/inhalts_bersicht.html (10.12.2013)

BildscharbV. (1996). Verordnung über Sicherheit und Gesundheitsschutz bei der Arbeit an Bildschirmgeräten (Bildschirmarbeitsverordnung - BildscharbV). (BGBl. I S. 1843). Abgerufen von http://www.gesetze-im-internet.de/bildscharbv/index.html#BJNR184300996BJNE000100000 (10.12.2013)

BITV 2.0. (2011). Barrierefreie Informationstechnik-Verordnung (BITV-2.0) - Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz vom 12.09.2011 (BGBl. I S. 1843). Abgerufen von http://www.gesetze-im-internet.de/bitv_2_0/BJNR184300011.html (10.12.2013)

BTU Cottbus - Senftenberg. (2013, Juli 28). CISAR - ein modulares Informationssystem für Archäologie und Bauforschung. Abgerufen von www.tu-cottbus.de/cisar/cisar/beschreibung.html (8.12.2013)

Bundesministerium des Inneren (Hrsg.). (2008). SAGA (Version 4.0) - Standards und Architekturen für E-Government-Anwendungen. Abgerufen von http://www.cio.bund.de/SharedDocs/Publikationen/DE/Architekturen-und-Standards/SAGA/saga_4_0_download.pdf?__blob=publicationFile (8.12.2013)

Comparison of JavaScript frameworks. (2013). In *Wikipedia, the free encyclopedia*. Abgerufen von https://en.wikipedia.org/w/index.php?title=Comparison_of_JavaScript_frameworks&oldid=586176545 (15.12.2013)

Corporate Design. (2013). In *Wikipedia*. Abgerufen von https://de.wikipedia.org/w/index.php?title=Corporate_Design&oldid=124930745 (12.12.2013)

DAI. (2013a). CISAR. Abgerufen von http://www.dainst.org/de/project/cisar?ft=all (10.12.2013)

DAI. (2013b). iDAI.field. Abgerufen von http://www.dainst.org/de/project/idaifield (10.12.2013)

Flanagan, D. (2011). *JavaScript: the definitive guide*. Beijing: O'Reilly.

Freeman, A. (2011). *The definitive guide to HTML5*. Berkeley, CA; New York: Apress.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns - Elements of Reusable Object-Oriented Software*. Boston: Addison-Wesley.

Harazim, S. (2012). Betriebspraktikum - Thema: „Konzeption und Implementierung eines Kontextbrowsers für eine Web-Anwendung". HTW Dresden.

Harazim, S. (2013). Bachelorarbeit - „Untersuchung zur barrierefreien Präsentation von Webinhalten und praktische Umsetzung am Beispiel des Projektes OpenInfRA". HTW Dresden.

International Ergonomics Association. (2013). What is Ergonomics. Abgerufen von http://www.iea.cc/whats/index.html (16.12.2013)

ISO. (1998a). Ergonomie der Mensch-System-Interaktion – Teil 12: Informationsdarstellung (ISO 9241-110:2006); (ISO 9241-12:1998).

ISO. (1998b). Ergonomie der Mensch-System-Interaktion Teil 11: Anforderungen an die Gebrauchstauglichkeit - Leitsätze. (EN ISO 9241 -11 :1998).

ISO. (2002a). Software ergonomics for multimedia user interfaces -- Part 1: Design principles and framework. (ISO 14915-1:2002).

ISO. (2002b). Software ergonomics for multimedia user interfaces -- Part 3: Media selection and combination. (ISO 14915-3:2002).

ISO. (2003). Software ergonomics for multimedia user interfaces -- Part 2: Multimedia navigation and control. (ISO 14915-2:2003).

ISO/IEC. (1998). Information technology - Open Distributed Processing - Reference model: Overview. (ISO/IEC 10746-1:1998(E)).

ISO/IEC. (2005). Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE. (ISO/IEC 25000:2005(E)).

ISO/IEC. (2006). Software-Engineering – Softwareproduktbewertung – Qualitätsanforderungen an kommerzielle serienmäßig produzierte Softwareprodukte (COTS) und Prüfanweisungen. (ISO/IEC 25051:2006).

ISO/IEC 25000. (2013). In *Wikipedia*. Abgerufen von http://de.wikipedia.org/w/index.php?title=ISO/IEC_25000&oldid=123384703 (16.12.2013)

ISO/TC 159. (2011). Ergonomie der Mensch-System-Interaktion - Teil 210: Prozess zur Gestaltung gebrauchstauglicher interaktiver Systeme (ISO 9241-210:2010).

Krasner, G. E., & Pope, S. T. (1988). A Cookbook for Using the Model-view Controller User Interface Paradigm in Smalltalk-80. *J. Object Oriented Program.*, *1*(3), 26–49.

Lubbers, P. (2011). *Pro HTML5 Programming*. (B. Albers & F. Salim, Hrsg.) (Second Edition.). Berkeley CA: Apress,. Abgerufen von http://dx.doi.org/10.1007/978-1-4302-3865-2 (15.12.2013)

Meyer, E. A. (2007). *CSS the definitive guide*. Beijing; Sebastopol, CA: O'Reilly.

Mock-up. (2013). In *Wikipedia*. Abgerufen von http://de.wikipedia.org/w/index.php?title=Mock-up&oldid=122800089 (10.12.2013)

Münz, S., & Gull, C. (2013). *HTML5-Handbuch*. Haar bei München: Franzis.

Nielsen, J., & Loranger, H. (2006). *Prioritizing web usability*. Berkeley California: New Riders.

Nielsen, J., & Pernice, K. (2010). *Eyetracking web usability*. Berkeley, CA: New Riders.

OpenInfRA. (2013). *OpenInfRA - Grobkonzept für ein webbasiertes Informationssystem zur Dokumentation archäologischer Forschungsprojekte V2.2*. HTW Dresden, BTU Cottbus - Senftenberg, DAI Berlin. Nicht veröffentlicht.

Pomberger, G., & Blaschek, G. (1996). *Software-Engineering: Prototyping und objektorientierte Software-Entwicklung*. München; Wien: Hanser.

Preim, B., & Dachselt, R. (2010). *Interaktive Systeme. Band 1: Grundlagen, Graphical User Interfaces, Informationsvisualisierung.* Heidelberg: Springer.

Prototyping (Softwareentwicklung). (2013). In *Wikipedia*. Abgerufen von http://de.wikipedia.org/w/index.php?title=Prototyping_(Softwareentwicklung)&oldid=122126123 (10.12.2013)

Rampl, H. (2007). Handbuch Usability. Abgerufen von http://www.handbuch-usability.de/ (16.12.2013)

REFA Bundesverband e. V. (2011). Ergonomie. Abgerufen von http://www.refa-lexikon.de/artikel/143/ergonomie (10.12.2013)

Rohles, B. (2013). *Grundkurs gutes Webdesign alles, was Sie über Gestaltung im Web wissen sollten*. Bonn: Galileo Press.

Sarodnick, F., & Brau, H. (2006). *Methoden der Usability Evaluation: wissenschaftliche Grundlagen und praktische Anwendung*. Bern: Huber.

Schmidtke, H., & Jastrzebska-Fraczek, I. (2013a). EKIDES – Ergonomics Knowledge and Intelligent Design System. Abgerufen von http://www.lfe.mw.tum.de/forschung/methoden-und-laboreinrichtungen/ekides/ (8.12.2013)

Schmidtke, H., & Jastrzebska-Fraczek, I. (2013b). *Ergonomie. Daten zur Systemgestaltung und Begriffsbestimmungen*. München: Hanser Verlag,.

Schmitt, C., & Simpson, K. (2011). *HTML5 cookbook*. Beijing; Cambridge, Mass.: O'Reilly.

Stakeholder. (2013). In *Wikipedia*. Abgerufen von http://de.wikipedia.org/w/index.php?title=Stakeholder&oldid=123787449 (10.12.2013)

Stapelkamp, T. (2010). *Interaction- und Interfacedesign. Web-, Game-, Produkt- und Servicedesign Usability und Interface als Corporate Identity*. Berlin Heidelberg: Springer Berlin Heidelberg.

Thesmann, S. (2010). *Einführung in das Design multimedialer Webanwendungen*. Wiesbaden: Vieweg+Teubner.

Usability. (2013). In *Wikipedia, the free encyclopedia*. Abgerufen von https://en.wikipedia.org/w/index.php?title=Usability&oldid=579736057 (12.12.2013)

VDI. (1990). Bürokommunikation - Software-Ergonomie in der Bürokommunikation. (VDI 5005:1990-10). Richtlinie zurückgezogen.

Website wireframe. (2013). In *Wikipedia, the free encyclopedia*. Abgerufen von https://en.wikipedia.org/w/index.php?title=Website_wireframe&oldid=584603582 (10.12.2013)

West, M. (2013). *HTML5 foundations*. Chichester, West Sussex, U.K.: John Wiley and Sons.

Abbildungsverzeichnis

Tabellenverzeichnis

Anlagenverzeichnis

Anhang A – Workflow GUI-Prototyp V2

GUI – Prototyp OpenInfRA (Stand mit Abgabe der Arbeit am 18.12.2013)

Der Prototyp ist hier zu erreichen:

www.magdalinski.eu/mscmag_v2

Nutzer: openinfra

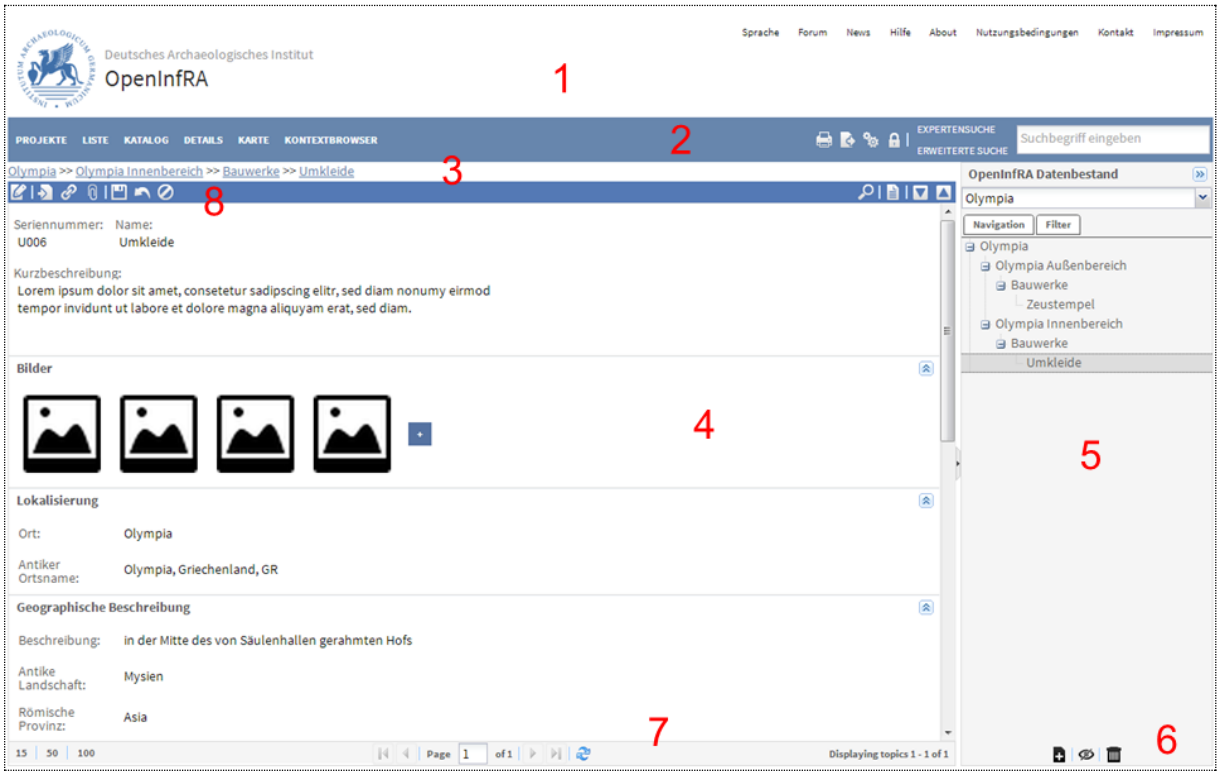
Passwort: guiprot!

Browser: Chrome

Wichtig:

- Die Unterscheidung, was im Prototyp funktioniert und was (noch) nicht, ist nicht explizit im Prototyp markiert. Der unten beschriebene Workflow zeigt, was möglich ist und dokumentiert so die verfügbare Funktionalität.

Grundlegendes Layout:



[*In eckigen Klammern jeweils die dazugehörige GUI-Komponenten wie in Kapitel 6 beschrieben*]

1 Header mit Logo und verschiedenen Links [*NorthPanel*]

2 Hauptmenü mit Link zur Projektübersicht und den verschiedenen Sichten auf die Daten (Liste, Katalog, Details, Karte und Kontext) und weiteren allgemeinen Werkzeugen (Druck, Export, Nutzereinstellungen, Login/Logout und verschiedene Arten der Suche) [*NorthPanel*]

3 *Bread crumbs*, die die aktuelle Position im Datenbestand anzeigen (synchronisiert mit dem Projektbaum rechts) [*CenterPanel*]

4 Container zur Detailansicht (sind Liste oder Katalog ausgewählt, dann wird der Inhalt hier ebenfalls dargestellt)

5 Ausblendbare Sidebar mit Tabs zur hierarchischen Navigation (Projektbaum) mit Filterkategorien [*EastPanel* / *EastTabPanel*]

6 Toolbar zum Erzeugen, Veröffentlichen und Löschen von Themeninstanzen [*EastPanel* / *EastTabPanel*]

7 Paginierungstoolbar mit Seitennavigation und der Auswahlmöglichkeit, wie viele Objekte pro Seite angezeigt werden sollen. [*CenterPanel*]

8 Editier-Toolbar zur Dateneingabe und –editierung (nur in der Detailansicht sichtbar) [*CenterPanel*]

Beispielworkflow:

1. Schritt: Projekt „Pergamon“ auswählen im Projektbaum (5)
    - Sichten (Liste, Katalog und Details) für Daten werden im Hauptmenü (2) sichtbar
    - Im *CenterPanel* (4) werden als Default-Ansicht alle Instanzen des Projektes „Pergamon“ in der Listenansicht gezeigt
    - Maus über den Spaltentitel „Name“ (ein Dropdown-Button wird sichtbar) -> klicken (4)
    - Auf den Eintrag „Columns“ klicken und einen Haken bei Seriennummer setzen (4)
    - Das Attribut „Seriennummer“ wird als zusätzliche Spalte eingeblendet (4)
    - Über die beiden „Sort“ Buttons kann die Sortierrichtung beliebig verändert werden (4)

2. Schritt: Navigation in hierarchischer Datenstruktur des Projektbaums: „Pergamon“ → „Pergamon-Ost“ → „Bauwerke“: „Bauwerke“ markieren (5)
    - Menge der angezeigten Themeninstanzen in *CenterPanel* (4) wird auf die Auswahl eingeschränkt und *bread crumbs* (3) werden angepasst
    - Klick auf „Pergamon Ost“ (die Ansicht in (4) und der Projektbaum (5) werden angepasst)

3. Schritt: Wechsel auf Katalogansicht im Hauptmenü (2)
    - Menge der Themeninstanzen wird in der Katalogansicht angezeigt (4)

4. Schritt: Auswahl „Rote Halle“ mit Doppelklick auf den Abschnitt in der Katalogansicht (4)
    - Themeninstanz wird in der Detailsicht angezeigt (4)

5. Schritt: Schließen aller Attributtypgruppen über Button (Pfeil nach oben) in der Editier-Toolbar (8)
    - Nur die „Grunddaten“ (Seriennummer, Name, Kurzbeschreibung) der Themeninstanz werden angezeigt (4)

6. Schritt: Attributtypgruppe „Charakterisierung“ über Button mit Doppelpfeil nach unten öffnen (4)

7. Schritt: „Tempel“ anklicken (4)

- Editiermodus wird gestartet
- Einträge in allen Formularfeldern können beliebig geändert werden (4)
- Den Eintrag in „Gebäudetyp“ bitte entfernen (4) -> daraufhin erscheint das Formularfeld mit einem roten Rand, da eine Pflichtangabe vergessen wurde
- Gebäudetyp in Halle ändern (4)

8. Schritt: Themeninstanz speichern über Disketten-Button in Editier-Toolbar (8)

- Beendet den Editiermodus und zeigt wieder die Detailansicht ohne Bearbeitungsmodus an (4)

9. Schritt: Neue Themeninstanz erzeugen über Plus-Button unter dem Projektbaum (6)

- Öffnet leeres Themeninstanz-Formular für das Thema „Bauwerk“ in Pergamon Ost (4)
- Alle Attributtypgruppen über Button mit dem Pfeil nach unten in der Editier-Toolbar (8) öffnen
- Projektbaum (5) über Button mit Doppelpfeil nach rechts weg- und wieder einklappen
- Größe kann über linke Grenze von (5) beliebig verbreitert oder schmaler gemacht werden

Anhang B – Workflow GUI-Prototyp- Informelle Umfrage

GUI – Prototyp OpenInfRA (Arbeitsstand vom 20.8.2013)

Ich würde mich freuen, wenn ihr euch ein wenig Zeit nehmt und den Beispiel-Workflow im Prototypen durchgeht.

Alle Fragen, Anmerkungen, Anregungen und Kritik sind willkommen und helfen mir den Prototypen zu verbessern.

Der Prototyp ist hier zu erreichen:

www.magdalinski.eu/mscmag

Nutzer: openinfra

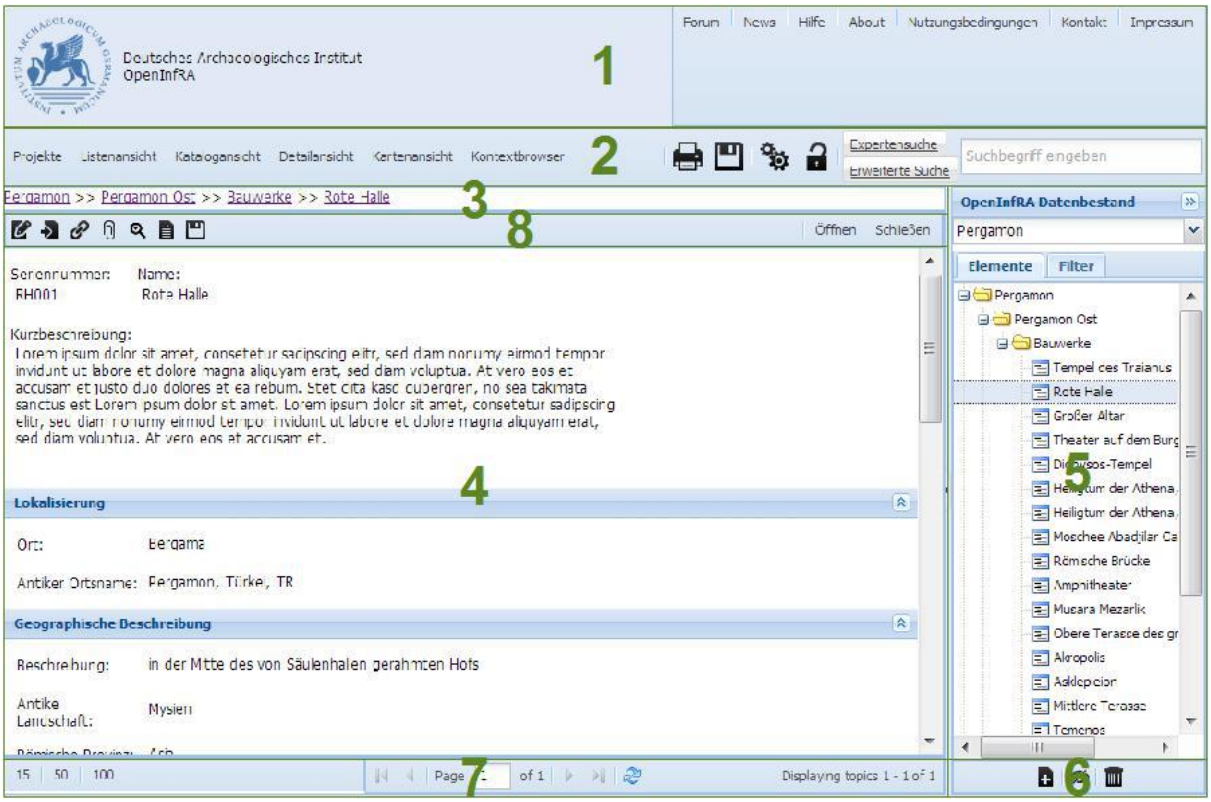
Passwort: guiprot!

Browser: Chrome

Wichtig:

- Dem Prototypen liegt noch kein „personalisiertes“ CSS zu Grunde (generelle Anpassungen und Anpassungen an den DAI-Styleguide und werden noch durchgeführt).
- Außer dem nachfolgenden Workflow (der hauptsächlich Listen-, Katalog- und Detailansicht enthält) sind noch keine weiteren Funktionen implementiert (Grenzen des Prototypen sind nicht markiert).

Grundlegendes Layout:



1 Header mit Logo und Links

2 Hauptmenü mit Link zur Projektübersicht und den verschiedenen Sichten auf die Daten (Liste, Katalog, Detail, Karte und Kontext)

3 BreadCrumb, der die aktuelle Position im Datenbestand anzeigt (später auch Such-, Filter-, Sortierangaben)

4 Datenfenster

5 Ausblendbare Sidebar mit Tabs zur hierarchischen Navigation, mit Filterkategorien (später evtl. auch Kontextmenüs)

6 Side-Toolbar mit Erzeugen, Veröffentlichen und Löschen von Daten

7 Bottom-Bar mit Paginierung und Mengenauswahl

8 Editier-Toolbar in der Detailansicht

Beispielworkflow:

1. Schritt: Projekt „Pergamon“ auswählen in Sidebar
   - Sichten (Liste, Katalog und Detail) für Daten werden im Hauptmenü sichtbar
   - Als Default-Ansicht werden alle Instanzen des Projektes „Pergamon“ in der Listenansicht gezeigt

2. Schritt: Navigation in hierarchischer Datenstruktur der Sidebar: „Pergamon“ → „Pergamon-Ost“ → „Bauwerke“: „Bauwerke“ markieren
   - Menge der angezeigten Themeninstanzen wird auf die Auswahl eingeschränkt

3. Schritt: Wechsel auf Katalogansicht im Hauptmenü
   - Menge der Themeninstanzen wird in der Katalogansicht angezeigt

4. Schritt: Auswahl „Rote Halle“ mit Doppelklick in Ansicht
   - Themeninstanz wird in der Detailsicht angezeigt

5. Schritt: Schließen aller Attributtypgruppen über Button in Editier-Toolbar
   - Nur die „Grunddaten“ (Seriennummer, Name, Kurzbeschreibung) der Themeninstanz werden angezeigt

6. Schritt: Attributtypgruppe „Charakterisierung“ öffnen

7. Schritt: „Tempel“ anklicken
   - Editiermodus wird gestartet
   - Einträge können geändert werden

8. Schritt: Themeninstanz speichern über Button in Editier-Toolbar
   - Beendet den Editiermodus

9. Schritt: Neue Themeninstanz erzeugen über Button in Side-Toolbar

- Öffnet leeres Themeninstanz-Formular für das Thema „Bauwerk“

10. Alle Attributtypgruppen über Button „Öffnen“ in der Editier-Toolbar öffnen

Anhang C – Digitale Anlagen

Der digitale Anhang befindet sich auf der beigefügten CD-ROM und enthält:

- Masterarbeit Magdalinski
- Anforderungskatalog Prototyp V2
- GUI-Prototyp-Quellcode
- Weitere Anlagen:
    - DAI-Styleguide
    - Grobkonzept OpenInfRA V2.2 vom 10.9.2013
        - Grobkonzept_OpenInfRA
        - Anhang A_Anwendungsfälle
        - Anhang B_Anforderungskatalog
        - Anhang C_Wertelisten
        - Anhang D_Rollen-Rechte-Matrix
        - Anhang E_Ausprägungen WebGIS-Client
        - Anhang F_Oberflächenprototypen
        - Anhang G_Datenbank-Integritätsbedingungen
        - Relationen v10

Erklärung über die eigenständige Erstellung der Arbeit

Hiermit erkläre ich, dass ich die vorgelegte Arbeit mit dem Titel

„Bereitstellung eines GUI-Prototyps für ein webbasiertes archäologisches Informationssystem unter besonderer Berücksichtigung von Usability-Aspekten“

selbständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie alle wörtlich oder sinngemäß übernommenen Stellen in der Arbeit als solche und durch Angabe der Quelle gekennzeichnet habe / haben. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet.

Mir ist bewusst, dass die Hochschule für Technik und Wirtschaft Dresden Prüfungsarbeiten stichprobenartig mittels der Verwendung von Software zur Erkennung von Plagiaten überprüft.

Dresden, 18.12.2013