

The background of the slide features a large, faint watermark of the University of Cologne seal. The seal is circular and contains a central illustration of the Virgin Mary seated with the Christ Child on her lap. To the left of the Virgin are two standing figures, likely saints or kings, and to the right is a kneeling figure. The entire scene is set within a gothic architectural frame. The Latin text 'COLONIENSIS' is visible at the top of the seal, and 'UNIVERSITAS' is at the bottom.

ASP.NET Einführung

Laboratory Course on Development

Inhaltsübersicht

■ Rückblick

- Technische Einführung
 - Datenbank Design
 - MVC Modell
 - Angewandte Webtechnologien

Rückblick auf unser LCD

Summary

- Vorstellung unseres Projekts „GECO“
- Aufbau des Teams, Erfahrungen in der Softwareentwicklung
- Nutzung von Visual Studio ab Pflichtenheft MockUp-Erstellung



Analysephase

- **Ausarbeitung des Lastenheftes**
 - 65-seitig
 - Erstellung der Use Cases
 - Sortierung nach MoSCoW
- **Ausarbeitung des Projektplans**
 - 206-zeilig
 - Agiles Projektmanagement
 - 5 Sprints
 - Festlegung der Meilensteine
- **Abgabe: 24.04.2015**

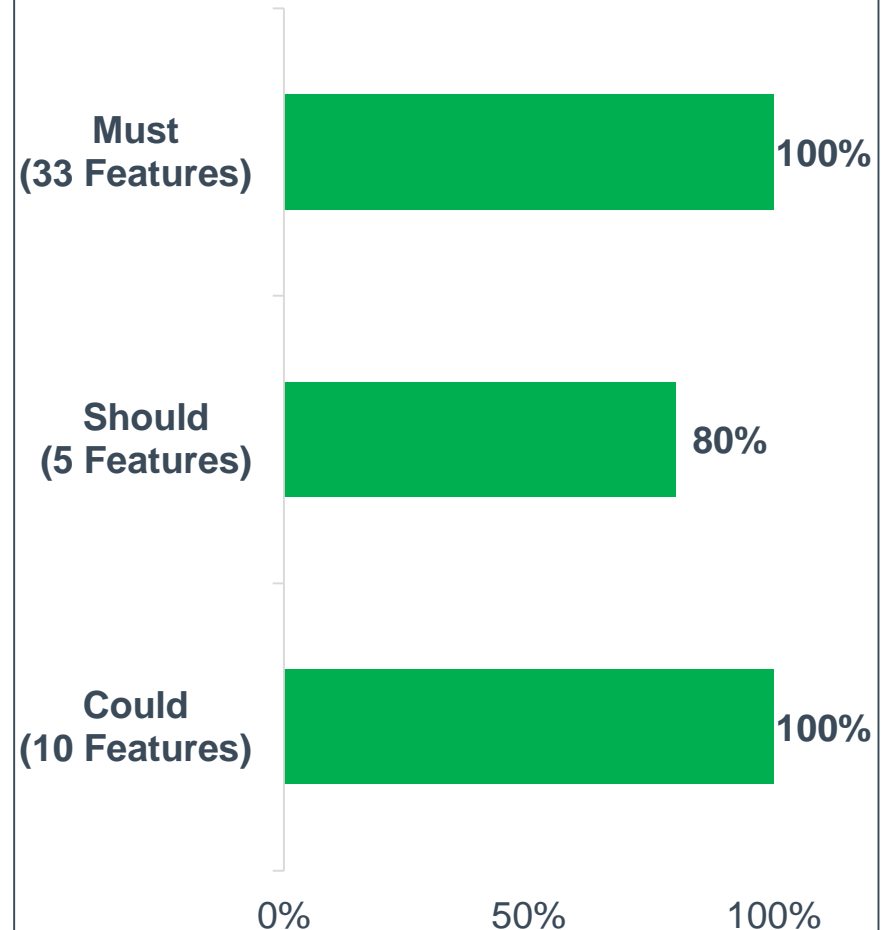
Entwurfsphase

- **Ausarbeitung des Pflichtenheftes**
 - 138-seitig
 - Festlegung der Funktionen
 - Erarbeitung der Testfälle
- **Ausarbeitung des Business Case**
 - In Kooperation mit der Stadt Köln
- **Erweiterung des Projektplans**
 - 653-zeilig
- **Abgabe : 15.05.2015**

Implementierungsphase

- **Implementierung von GECO**
 - 3491 Zeilen Code
 - Dauerhaft lauffähiges Programm
 - Durchgängige Komponententests
- **Ausarbeitung der Dokumentation**
 - 95-seitig
 - Entwickler-, Nutzerhandbuch und Installationsanleitung
- **Abgabe: 10.07.2015**

MoSCoW

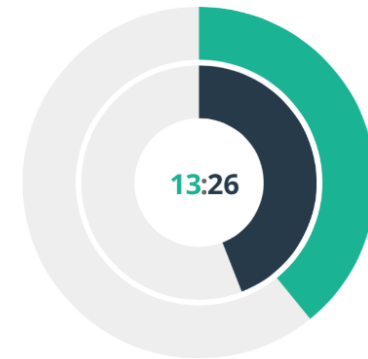


Unser Projekt



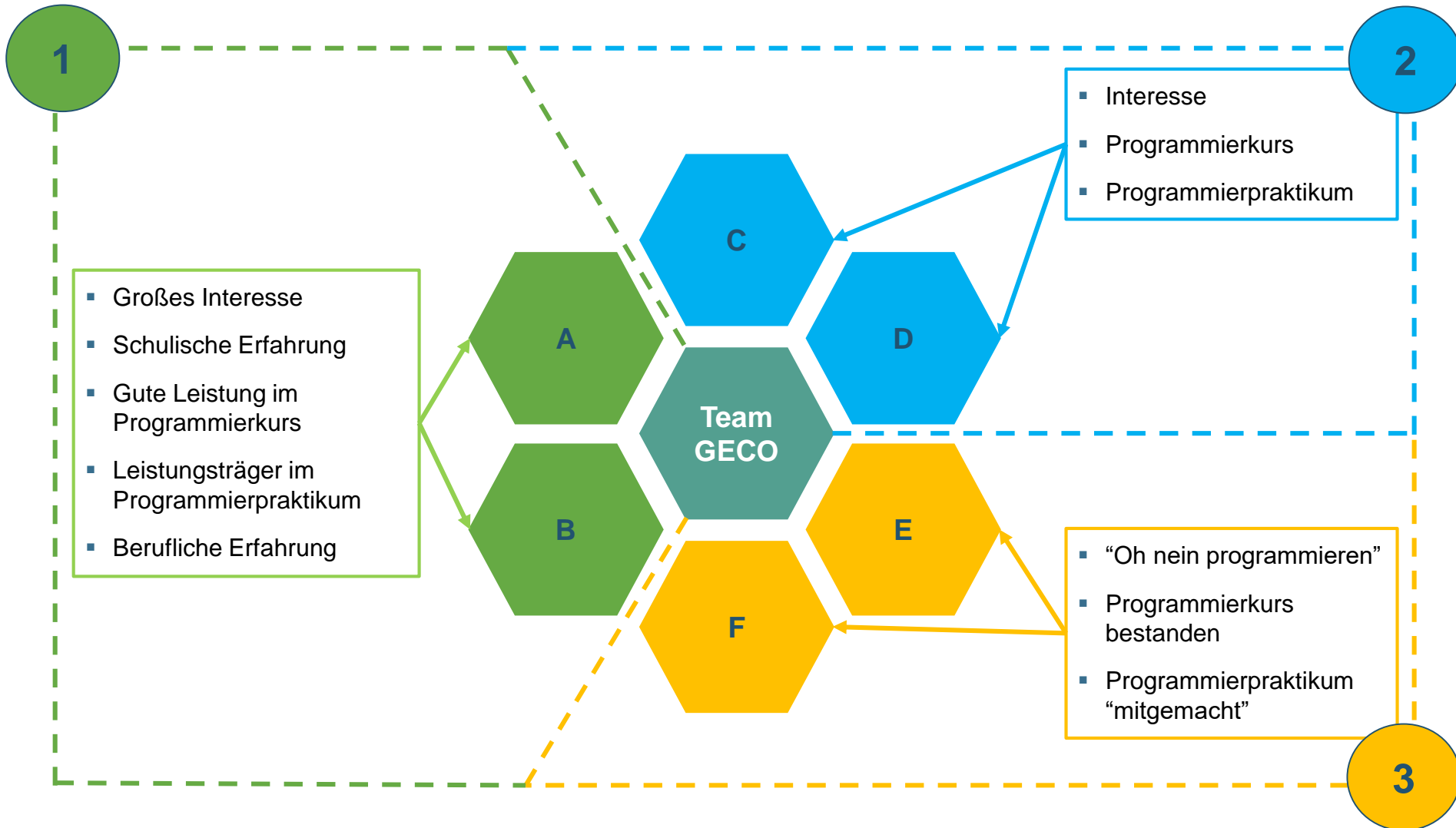
Die Parkraumverwaltung innovativ
und einfach gestalten durch **GECO**:

- Einfache bargeldlose Bezahlung
von Parktickets
- Vereinfachung der
Parkplatzsuche durch
Visualisierung der
Parkraumauslastung



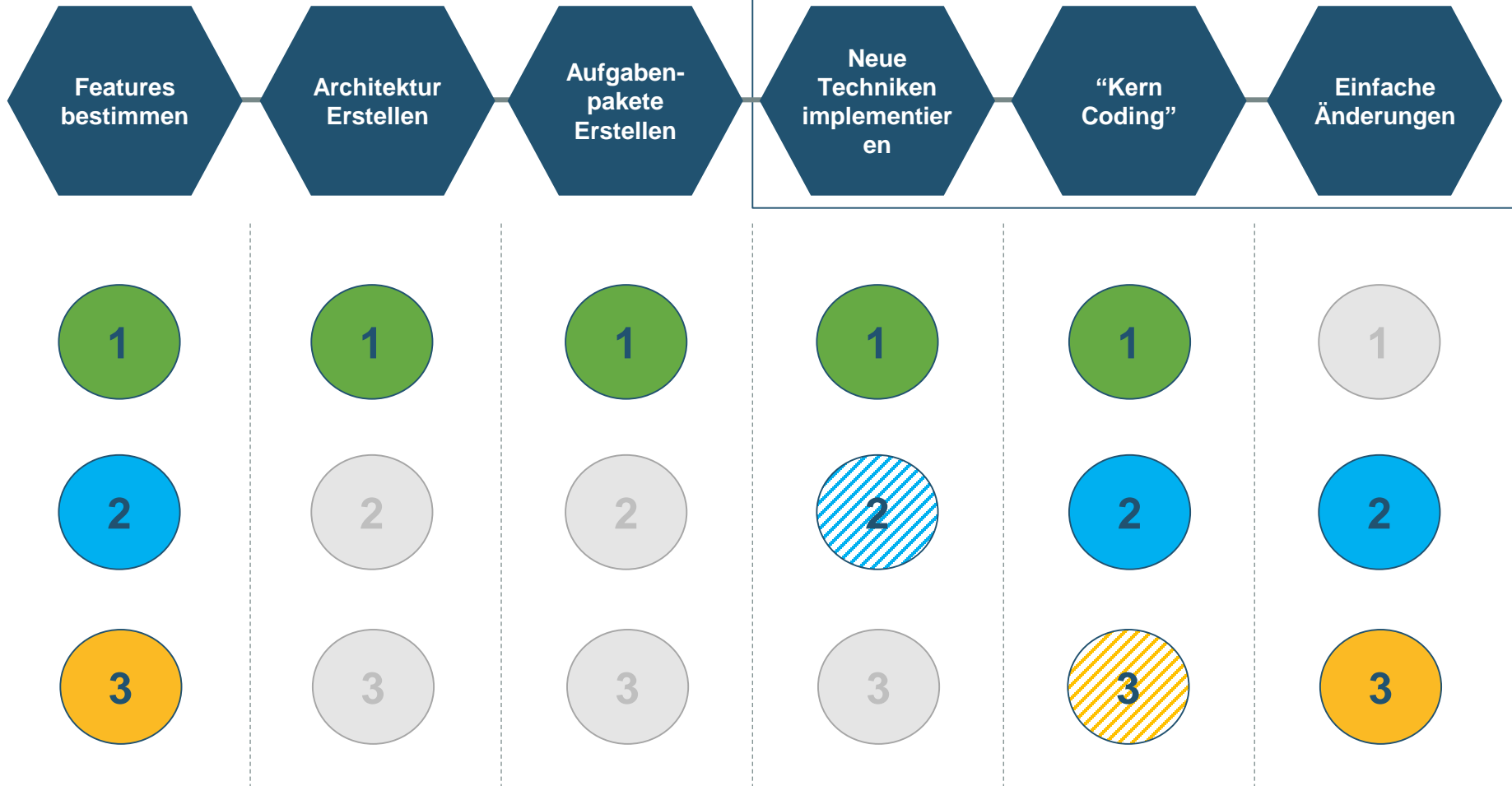
Das Team

Erfahrungsungleichheit



Das Team

Aufgabenverteilung



- 1 Erfahrene Programmierer
- 2 Programmieraffine Member
- 3 Wenig affine und erfahrene Member

Frühzeitige Nutzung von Visual Studio 1/2

Mockup als Storyboard (Powerpoint)



Geco – Parkticket kaufen

http://www.gecolive.azurewebsites.net

Home Registrieren Einloggen

Karte

Mein Ticket

Parkzeit bis

15:54 Uhr

KFZ Kennzeichen

2,5c/Minute

4h Maximale Parkdauer

5,00€ Ticketpreis

Jetzt bezahlen Abbrechen

Frühzeitige Nutzung von Visual Studio 2/2

Empfehlung: Ab Mockup-Erstellung Webprojekt in VS anlegen



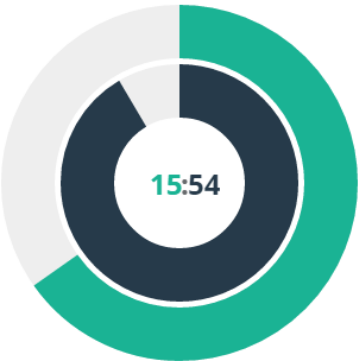
Geco – Parkticket kaufen

http://www.gecolive.azurewebsites.net

Registrieren Einloggen

Karte
Mein Ticket

Parkticket kaufen



Altstadt-Süd - Nord-Süd-Fahrt

Kennzeichen	VIE-PD-292
Minimales Ticketintervall	20
Preis pro Minute	2.5 Cent
Maximale Parkdauer	04:00:00
Rechnungsbetrag	05,00€

Jetzt Bezahlen Abbrechen

Inhaltsübersicht

- Rückblick
- **Technische Einführung**
 - **Datenbank Design**
 - MVC Modell
 - Angewandte Webtechnologien

Datenbank Design

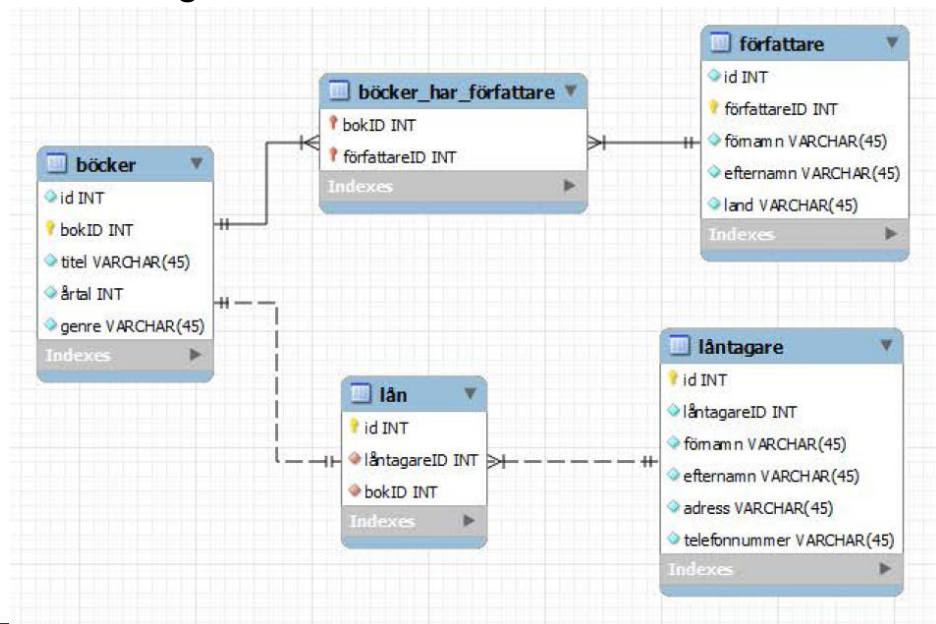
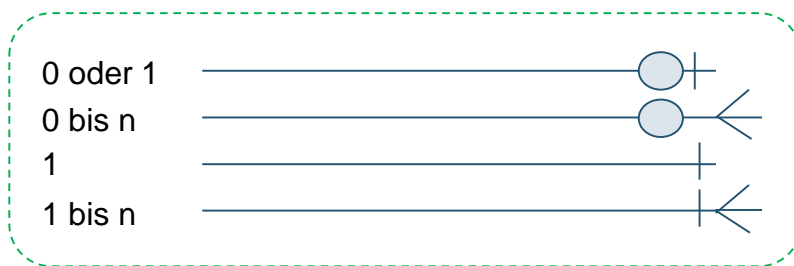
Summary

- Datenbank Design (MSSQL vs. MySQL)
- Database First Ansatz
- Krähenfußnotation

Datenbank Design



- Database First Ansatz sinnvoll und für Requirements Engineering eigentlich unabdingbar
- MySQL und MSSQL sind unterschiedlich
- Eine Konvertierung ist nicht in jedem Fall ohne weiteres möglich
- Tools wie MySQL Workbench erstellen „schöne“ Diagramme
- Diagramme können nur in MySQL Code exportiert werden
- Daher -> Nutzung von Microsoft SQL Server Manager zur Erstellung der Diagramme
- ER-Diagramm mit Krähenfußnotation ist zuerst ungewöhnlich aber deutlich übersichtlicher als Chen



Quelle: <http://i.stack.imgur.com/VzgTz.png>

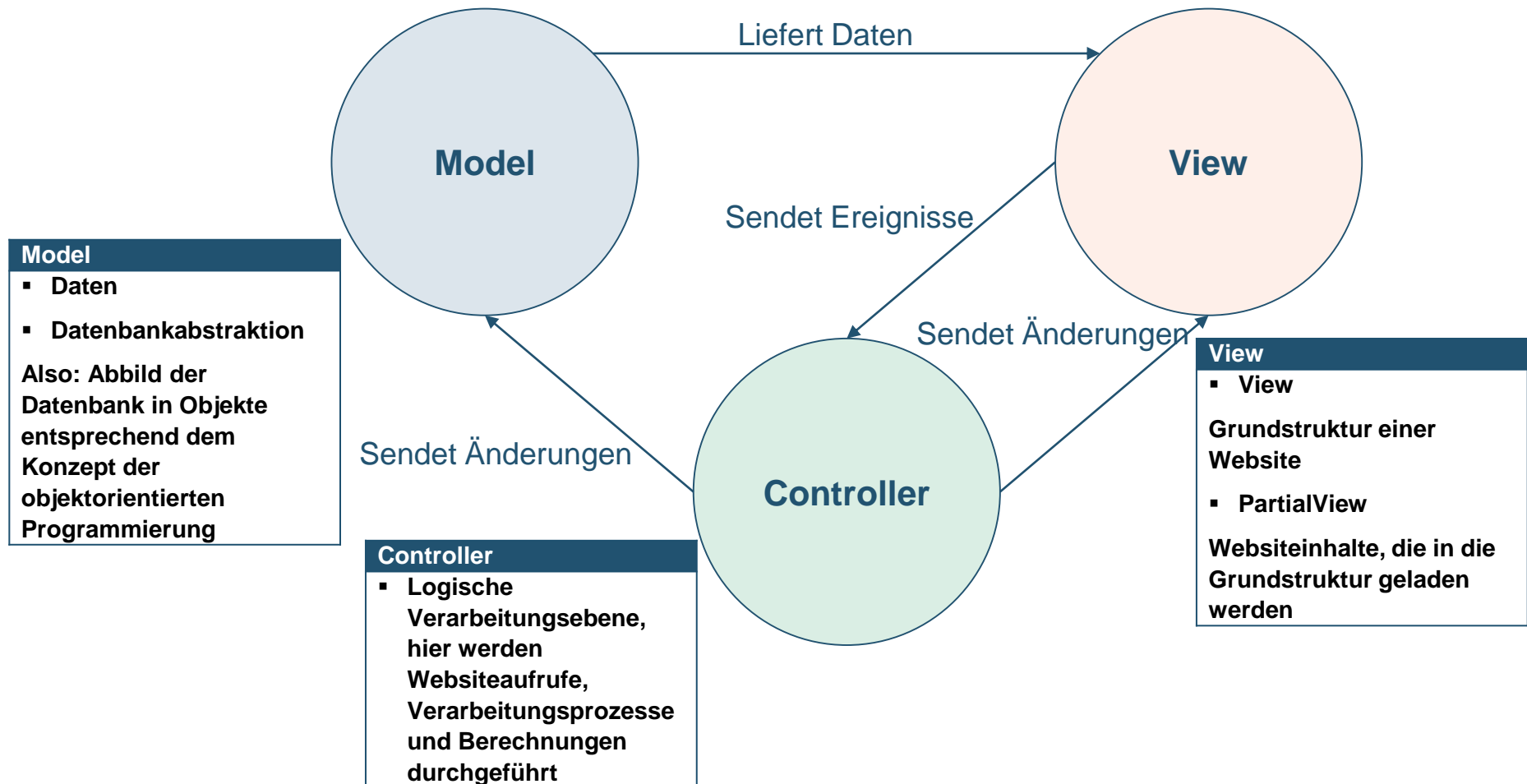
MVC Modell

Summary

- **MVC Theorie**
- **MVC Aufbau und Funktion in ASP.NET Projektmappen**
- **Angewandtes MVC**
 - **Controller Actions aufrufen**
 - **Zugriff auf Übergabeparameter**
- **ASP.NET Praxis-Grundlagen**
 - **Aufbau ViewBag**
 - **Steuerelemente (TextBox, Button, DropDownfield)**

MVC Modell

Theorie



MVC Modell

Theorie



3 Verweise

```
public class LoginViewModel
```

```
{
```

```
    [Required]
```

```
    [Display(Name = "E-Mail")]
```

```
    [EmailAddress]
```

4 Verweise

```
    public string Email { get; set; }
```

```
    [Required]
```

```
    [DataType(DataType.Password)]
```

```
    [Display(Name = "Kennwort")]
```

4 Verweise

```
    public string Password { get; set; }
```

```
    [Display(Name = "Speichern?")]
```

4 Verweise

```
    public bool RememberMe { get; set; }
```

```
}
```

Anmelden.

Lokales Konto für die Anmeldung verwenden.

E-Mail

Kennwort

☐ Speichern?

Anmelden

[Als neuer Benutzer registrieren](#)

© 2015 - Meine ASP.NET-Anwendung

```
// POST: /Account/Login
```

```
[HttpPost]
```

```
[AllowAnonymous]
```

```
[ValidateAntiForgeryToken]
```

0 Verweise

```
public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
```

```
{
```

```
    if (!ModelState.IsValid)
```

```
    {
```

```
        return View(model);
```

```
    }
```

```
    // Anmeldefehler werden bezüglich einer Kontosperrung nicht gezählt.
```

```
    // Wenn Sie aktivieren möchten, dass Kennwortfehler eine Sperre auslösen, ändern Sie in "shouldLockout: true".
```

```
    var result = await SignInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe, shouldLockout: false);
```

```
    switch (result)
```

```
    {
```

```
        case SignInStatus.Success:
```

```
            return RedirectToLocal(returnUrl);
```

```
        case SignInStatus.LockedOut:
```

```
            return View("Lockout");
```

```
        case SignInStatus.RequiresVerification:
```

```
            return RedirectToAction("SendCode", new { ReturnUrl = returnUrl, RememberMe = model.RememberMe });
```

```
        case SignInStatus.Failure:
```

```
            default:
```

```
                ModelState.AddModelError("", "Ungültiger Anmeldeversuch.");
```

```
                return View(model);
```

```
    }
```

Model

View

Controller

MVC Modell

Theorie



3 Verweise

```
public class LoginViewModel
{
    [Required]
    [Display(Name = "E-Mail")]
    [EmailAddress]
    4 Verweise
    public string Email { get; set; }

    [Required]
    [DataType(DataType.Password)]
    [Display(Name = "Kennwort")]
    4 Verweise
    public string Password { get; set; }

    [Display(Name = "Speichern?")]
    4 Verweise
    public bool RememberMe { get; set; }
}
```

Anmelden.

Lokales Konto für die Anmeldung verwenden.

• Ungültiger Anmeldeversuch.

E-Mail

test@test.de

Kennwort

☐ Speichern?

Anmelden

[Als neuer Benutzer registrieren](#)

© 2015 - Meine ASP.NET-Anwendung

```
// POST: /Account/Login
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
0 Verweise
public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
{
    if (!ModelState.IsValid)
    {
        return View(model);
    }

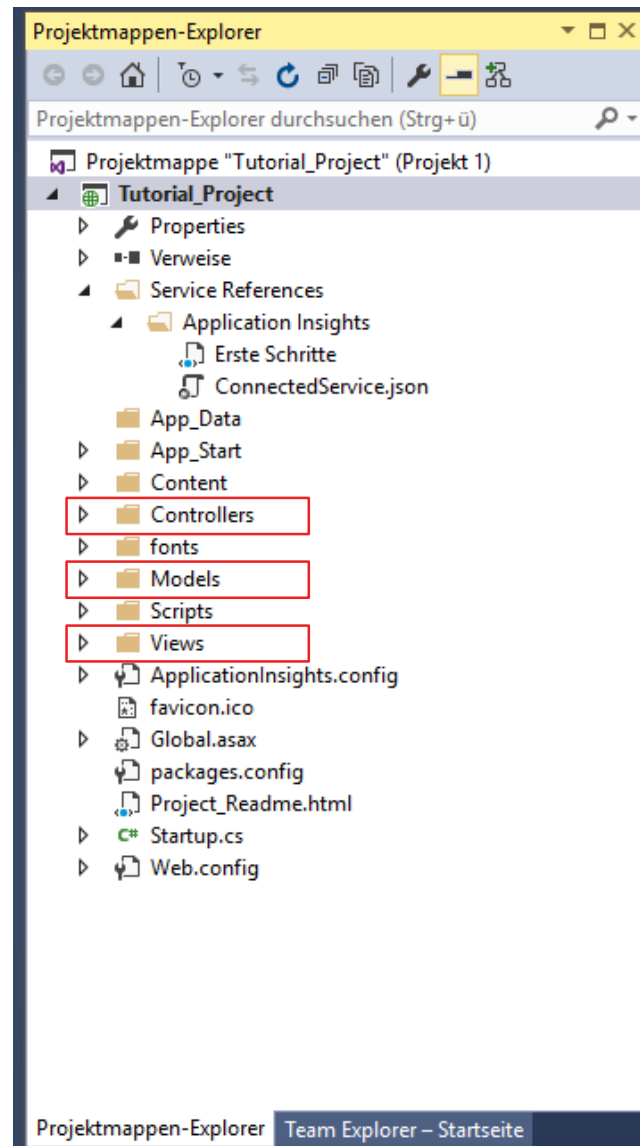
    // Anmeldefehler werden bezüglich einer Kontosperrung nicht gezählt.
    // Wenn Sie aktivieren möchten, dass Kennwortfehler eine Sperre auslösen, ändern Sie in "shouldLockout: true".
    var result = await SignInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe, shouldLockout: false);
    switch (result)
    {
        case SignInStatus.Success:
            return RedirectToLocal(returnUrl);
        case SignInStatus.LockedOut:
            return View("Lockout");
        case SignInStatus.RequiresVerification:
            return RedirectToAction("SendCode", new { ReturnUrl = returnUrl, RememberMe = model.RememberMe });
        case SignInStatus.Failure:
        default:
            ModelState.AddModelError("", "Ungültiger Anmeldeversuch.");
            return View(model);
    }
}
```

ModelState.AddModelError(...)

Model
View
Controller

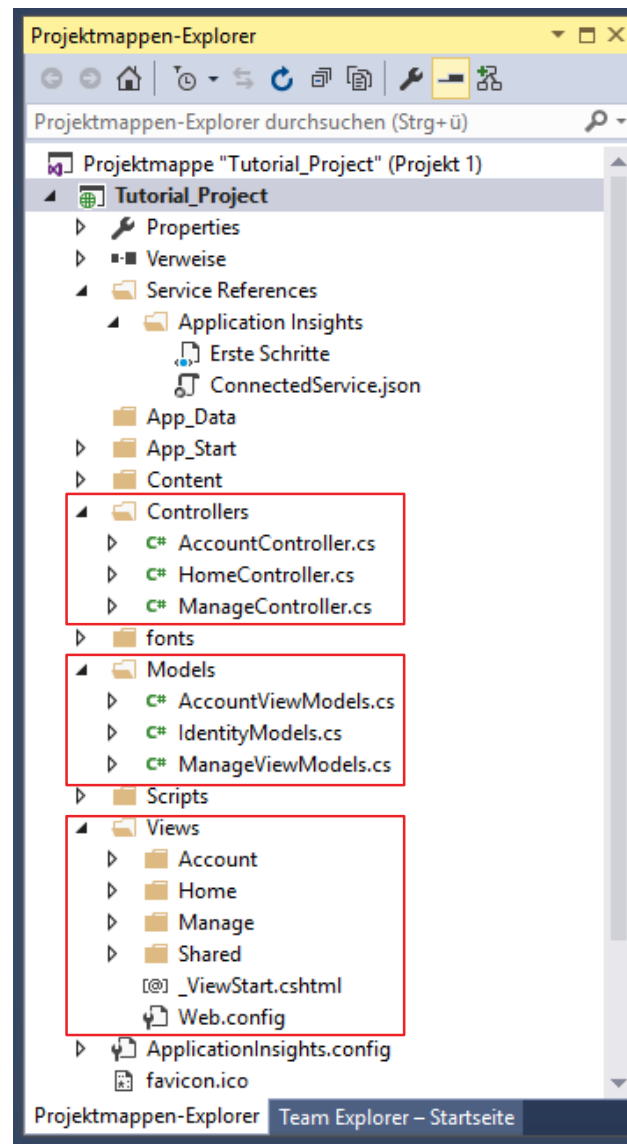
MVC Modell

Komplettansicht eines neuen Projekts im Explorer



MVC Modell

Komplettansicht eines neuen Projekts im Explorer



Beispiel einer View



```
@model Tutorial_Project.Models.ArticleDetailModel  
  
{  
    ViewBag.Title = "Details";  
}  
  
<h2>Details</h2>  
  
<div>  
    <h4>ArticleDetailModel</h4>  
    <hr />  
    <dl class="dl-horizontal">  
        <dt>  
            @Html.DisplayNameFor(model => model.name)  
        </dt>  
        <dd>  
            @Html.DisplayFor(model => model.name)  
        </dd>  
  
        <dt>  
            @Html.DisplayNameFor(model => model.price)  
        </dt>  
        <dd>  
            @Html.DisplayFor(model => model.price)  
        </dd>  
    </dl>  
</div>  
<p>  
    @Html.ActionLink("Edit", "Edit", new { id = Model.id }) |  
    @Html.ActionLink("Back to List", "Index")  
</p>
```

- Zugehöriges Model
- Zugriff auf C# Code
- Eine Zeile
- Mehrere Zeilen
- Normaler HTML-Code
- Vorgefertigte ASP.NET Website-Elemente (Werden in HTML-Code umgewandelt)

Steuerelemente

Das DropDown Feld



View

```
<div class="form-group">
    @Html.LabelFor(model => model.StudentId, "StudentId", htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.DropDownList("StudentId", String.Empty)
        @Html.ValidationMessageFor(model => model.StudentId)
    </div>
</div>
```

Controller

```
// GET: Addresses/Create
public ActionResult Create()
{
    ViewBag.CityId = new SelectList(db.Cities, "Id", "Name");
    ViewBag.StudentId = new SelectList(db.Students, "Id", "Prenome");
    return View();
}
```

[Home](#) / Adresse

Address

Street	<input type="text"/>
Postcode	<input type="text"/>
StreetNo	<input type="text"/>
Appendix	<input type="text"/>
StudentId	<div><div>Nadimo</div><div>Patrick</div></div>
CityId	<input type="text"/>

[Back to List](#)

- Eine Liste mit den Attributen ID und Name aus der Datenbanktabelle Students wird erstellt und der View zur Verfügung gestellt



TextBox (automatisch durch EditorFor)

```
<div class="form-group">
    @Html.LabelFor(model => model.Street, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.Street)
        @Html.ValidationMessageFor(model => model.Street)
    </div>
</div>
```

- Wählt man als Typ für sein Eingabefeld „EditorFor“, muss lediglich die zu beschreibende Variable im Model ausgewählt werden. ASP.Net erstellt dann im Fall eines Strings automatisch ein Textfeld
- Für Integer-Werte wird ein Textfeld mit Up/Down-Funktion erstellt und für Boolean-Werte Checkboxen
- @HTML.ValidationMessage Überprüft, ob die Eingabe valide war (Form)
- Alternativ erstellt @HTML.TextBoxFor explizit ein Textfeld

[Home](#) / Adresse

Address

Street	<input type="text" value="Hallo123"/>
Postcode	<input type="text"/>
StreetNo	<input type="text"/>
Appendix	<input type="text"/>
StudentId	<input type="text" value="v"/>
CityId	<input type="text" value="v"/>
	<input type="button" value="Create"/>

[Back to List](#)

ActionResult Funktion

Unterschied Post und Get



Get

- Sichtbarkeit in Adresszeile gegeben, da Parameter hier übergeben werden und nur elementare Parameter genutzt werden (Integer, String)
- Meist genutzt bei Delete, Index oder Detail Operationen

[HttpGet]

```
public ActionResult Index()
{
    var addresses = db.Addresses.Include(a => a.City)
        .Include(a => a.Student);
    return View(addresses.ToList());
}
```

Post

- Nicht sichtbar, da Datenaustausch Serverseitig
- Übergabe von Objekten möglich
- Meist genutzt bei Save oder Edit Operationen

[HttpPost]

```
public ActionResult SaveData(Address address)
{
    if (ModelState.IsValid)
    {
        if (address.Id > 0)
        {
            db.Entry(address).State = EntityState.Modified;
        }
        else
        {
            db.Addresses.Add(address);
        }
        db.SaveChanges();
    }
    return null;
}
```

Website-Design

Bootstrap Templates



Variante 1: Standard-Bootstrap-Template

Variante 2: Kostenlose Anbieter

<http://startbootstrap.com/>

- Bootstrap Templates: zur Transformation dieser Templates in MVC ASP.NET Projekte kann sich an dieser Anleitung / diesem Beispiel orientiert werden:

<https://hgminerva.wordpress.com/2015/02/08/how-to-create-an-asp-net-mvc-5-site-using-bootstrap-free-templates/>

- Alternativ ein Tool zur automatischen Template Generierung für VS:

<http://bootstrapbundle.com>

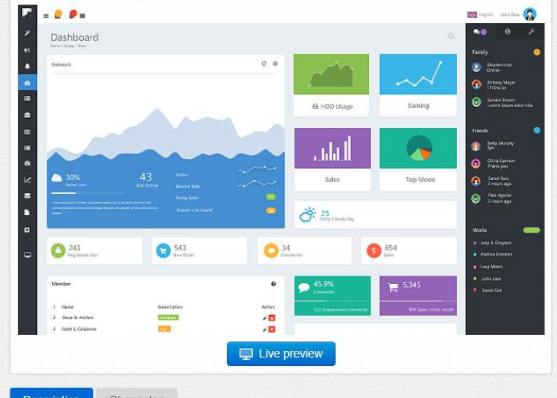
<http://bootswatch.com/>

- CSS Anpassungen und Designs für das „Standard“-Bootstrap-Template

Variante 3: Kostenpflichtige Anbieter

<https://wrapbootstrap.com/>

- Preis ca. 20\$-30\$



\$20 Buy now using PayPal >>

That's about 8 cups of coffee ☕

1305 Purchases

License: Single application

Licenses: Details >

License	Price
Single application	\$20
Multiple applications	\$80
Extended	\$1000

Item attributes

Version:	2.2.1
Type:	HTML Template
Bootstrap:	Compatible with 3.3.x
Layouts:	Responsive

ACHTUNG:

Nur Templates die explizit für **ASP.NET MVC 5** vorgesehen sind, können leicht implementiert werden!

Webtechnologien

Summary

- **AJAX in Praxis**



Partial Views

Unterschied Asp.Net Partial View und AJAX Partial View



ASP.NET MVC5 PartialView

Lädt einen bestimmten Inhalt durch das Neuladen einer gesamten Website nach, heißt:

- Teilinhalte einer Website werden in einem Block in einer HTML-Seite geladen.
- gesamter Ladeprozess einer Website
- Kein Austausch einzelner Teilinhalte oder mehrere Teilinhalte auf einer HTML-Seite

AJAX PartialView

Lädt bestimmte Inhalte durch das Neuladen eines bestimmten Teilinhalts nach, heißt:

- Teilinhalte einer Website werden in Blöcken in einer HTML-Seite geladen.
- partieller Ladeprozess von Websiteinhalten
- Austausch oder Nachladen mehrerer Teilinhalte auf einer HTML-Seite

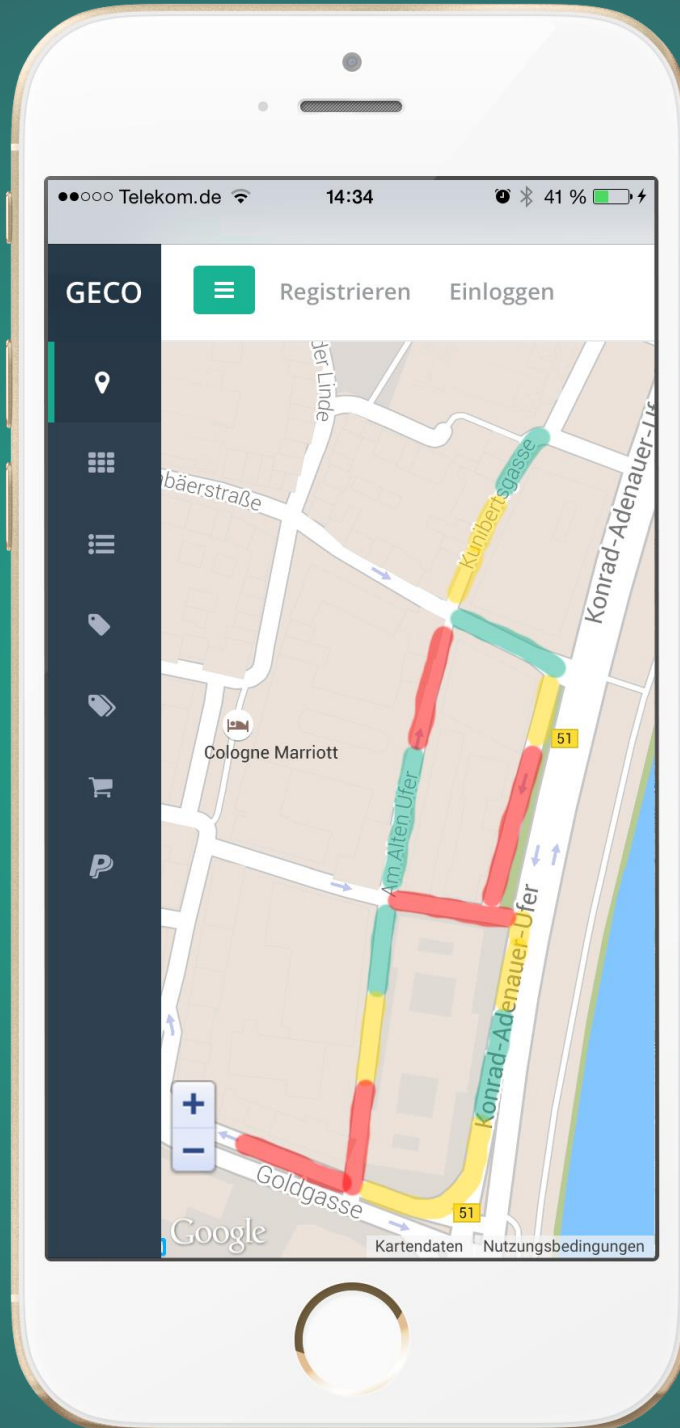
- Um Inhalte zu laden muss beispielsweise ein Button per HTML (also nicht mit ASP.NET Hilfe) implementiert werden, der eine JavaScript Funktion aufruft
- Zur Hilfe stellen wir euch unser Framework bereit, welches wir im LCD entwickelt haben

```
<input type="button" value="Create New"
      onclick="partialLoad('/Students/_Create', 'GET', '_partialStudents', 'html', '');"/> />
```

Appendix

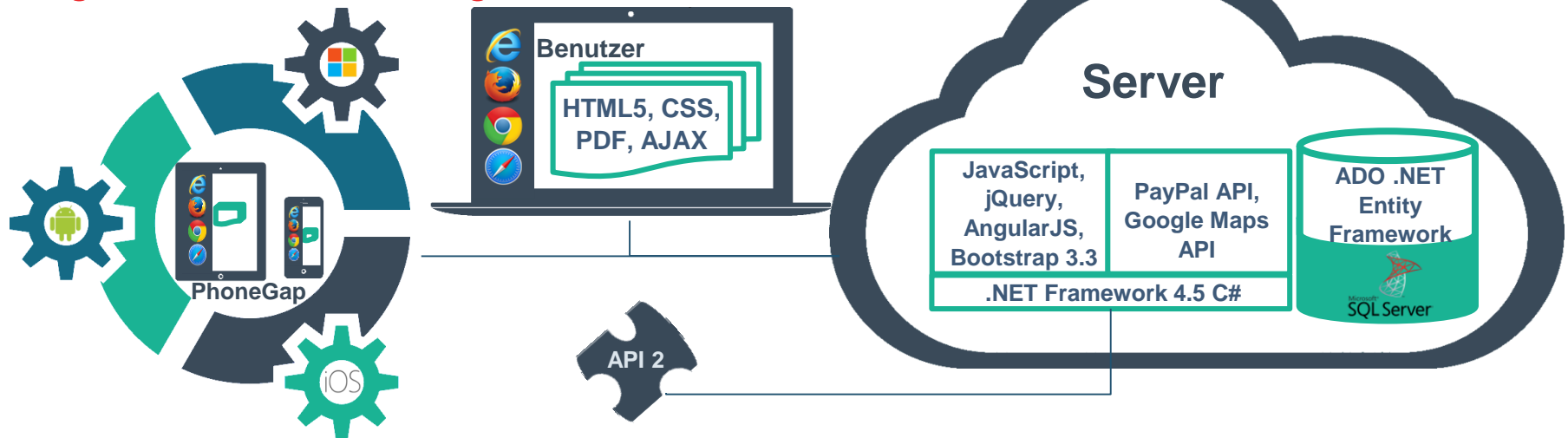


Quelle:



Webtechnologien

Angewandte Technologien



Vorteile

- Clientseitige Verwendung modernster Webstandards mit maximaler Geräteunabhängigkeit
- Schonung von Netzressourcen durch asynchrone Datenübertragung mit AJAX (Prinzip: lade nur, was nötig ist)
- Verwendung bewährter Frameworks, in aktueller Version
- Drittanbieter-Nutzung und API-Schaffung zur Erweiterung des Service-Angebots
- Zentralisierung: Singleton-Entwurfsmuster für APIs

Umsetzungsroadmap

Am Beispiel der Stadt Köln



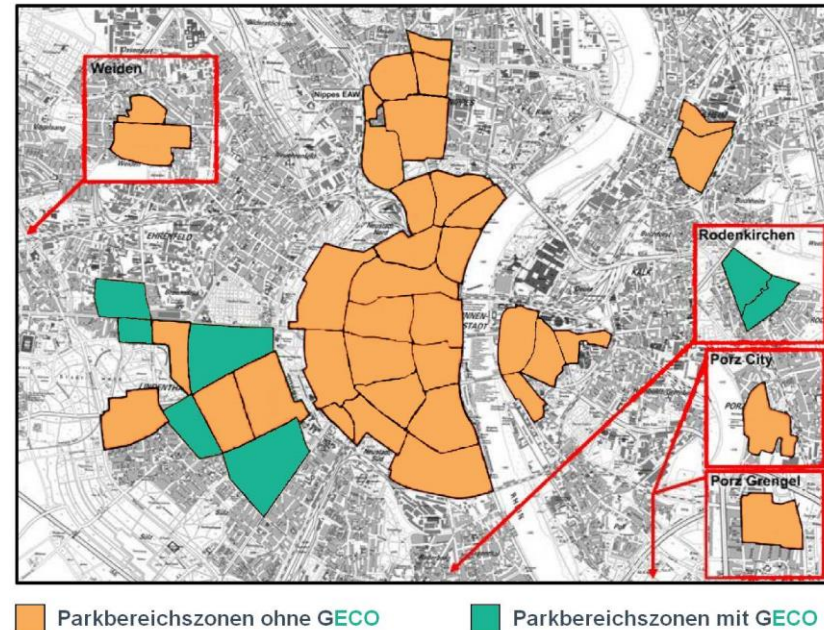
Jahr 1

Jahr 2

Jahr 3

Jahr 4

- Beginn im **nicht** erschlossenen Parkraum
- Aufbau eines **Prototypen**
- Einrichten des Systems auf lauffähigen Servern
- Werbe- und Informationskampagne
 - Ausstatten der Parkscheinautomaten mit notwendigen Informationen
 - Freie Werbung in der Stadt
 - Nutzung der Medien
- **Anreizschaffung** durch Punktesystem
 - Punktevergabe für jedes gekaufte Ticket
 - Punkteinsatz zum vergünstigten Ticket Erwerb
- Umrüstung defekter Parkautomaten



Umsetzungsroadmap

Am Beispiel der Stadt Köln



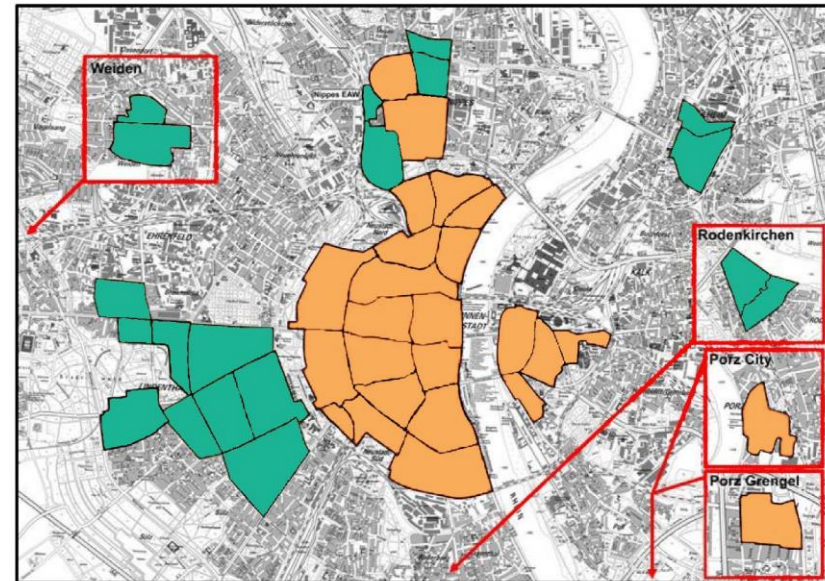
Jahr 1

Jahr 2

Jahr 3

Jahr 4

- Einführung des Systems in 20% der **erschlossenen** Parkbereichszonen
- Einpflegen der Parkbereichszonen in das System
- **Umrüstung** der Parkscheinautomaten mit neuer Technologie
- Ausweitung der Werbe- und Informationskampagne



Orange Parkbereichszonen ohne GECO

Green Parkbereichszonen mit GECO

Umsetzungsroadmap

Am Beispiel der Stadt Köln



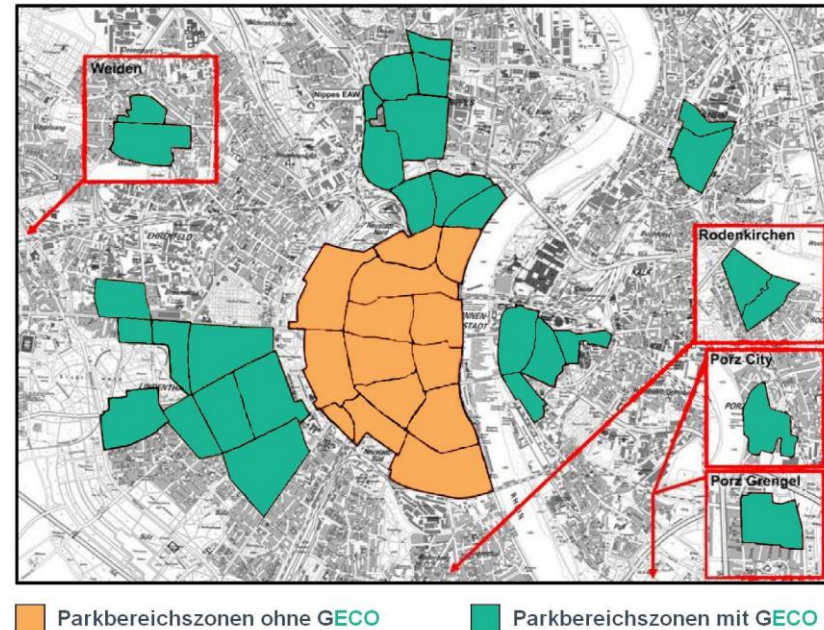
Jahr 1

Jahr 2

Jahr 3

Jahr 4

- **Erschließung weiterer 30% der Parkbereichszonen**
 - Einpflegen in das System
 - Umbau der Parkscheinautomaten
- **Fortführung der Werbe- und Informationskampagne**
- **Beginn mit Rückbau der Parkscheinautomaten**



Umsetzungsroadmap

Am Beispiel der Stadt Köln



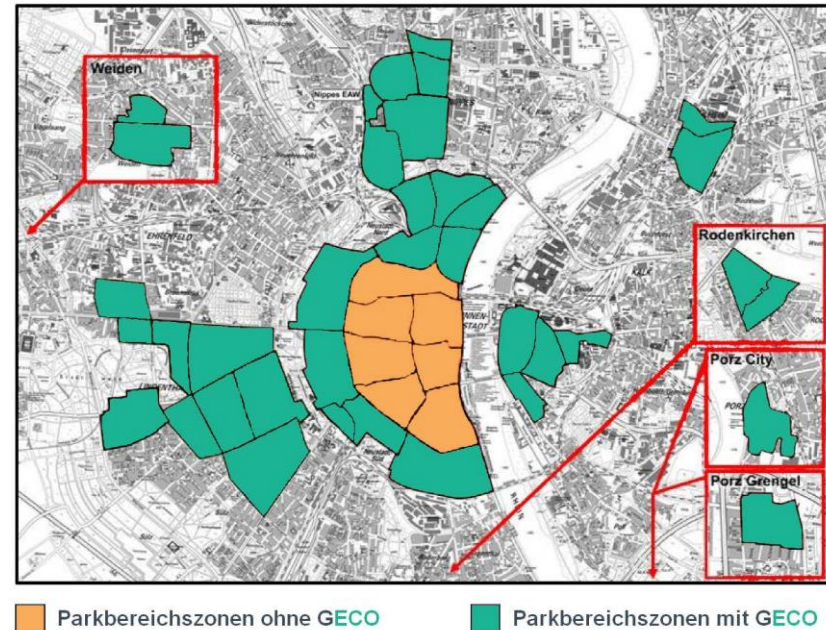
Jahr 1

Jahr 2

Jahr 3

Jahr 4

- **Erschließung weiterer 14% der Parkbereichszonen**
 - Einpflegen in das System
 - Umbau der Parkscheinautomaten
- **Weiterer Abbau der Automaten**
- **Verringerung der Werbe- und Informationskampagne**

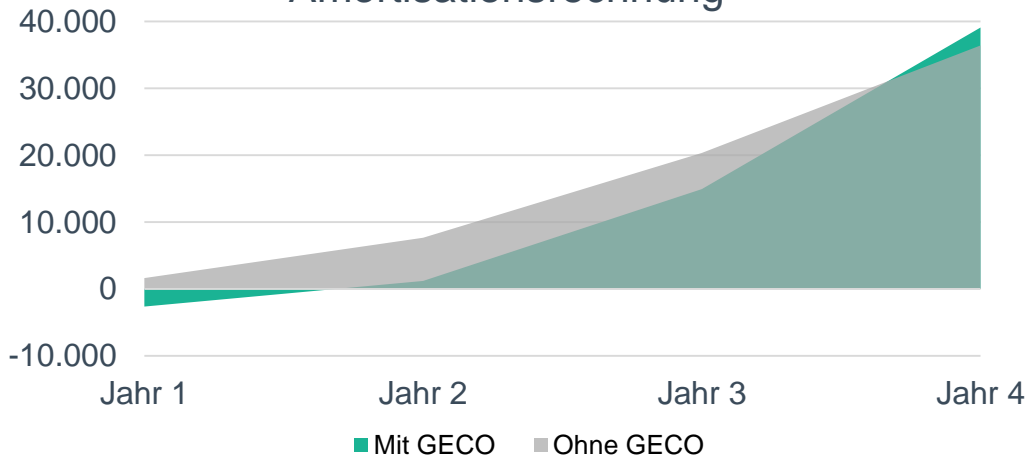


Business Case (in Tsd.€)

Am Beispiel der Stadt Köln



Amortisationsrechnung



Mit GECO

	Jahr 1	Jahr 2	Jahr 3	Jahr 4
Einnahmen	2.380,00 €	7.945,00 €	17.797,00 €	27.087,00 €
Ausgaben	5.012,45 €	4.121,79 €	4.040,50 €	2.942,28 €
Gewinn / Verlust	-2.632,55 €	3.823,21 €	13.757,20 €	24.144,85 €
Akkumuliert	-2.632,55 €	1.190,66 €	14.947,86 €	39.092,71 €

Ohne GECO

Einnahmen	2.280,00 €	6.900,00 €	14.046,00 €	17.530,00 €
Ausgaben	629,13 €	911,63 €	1.278,88 €	1.448,38 €
Gewinn / Verlust	1.650,88 €	5.988,38 €	12.767,13 €	16.081,63 €
Akkumuliert	1.650,88 €	7.639,26 €	20.315,39 €	36.397,02 €

Erklärung

- Erhöhte Investitionskosten durch **GECO** führen zu 2.632,55 Tsd. € Verlust im ersten Jahr (7 Gebiete)
- Im zweiten Jahr (17 Gebiete) werden erste Gewinne realisiert
- Mit **GECO** Gewinn im dritten Jahr (30 Gebiete) erstmals höher als ohne **GECO**
- 3,7 Jahre nach Einführung von **GECO** ist die Nutzenschwelle erreicht
 - 80 % der Parkbereichszonen erschlossen
 - **GECO** rentabel
- **2.695,69 Tsd. € Gewinn**