



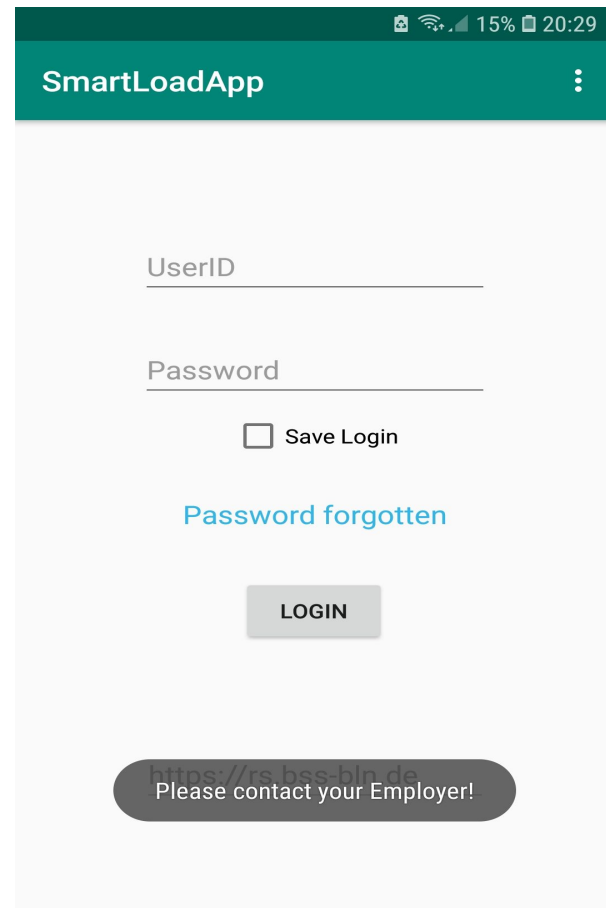
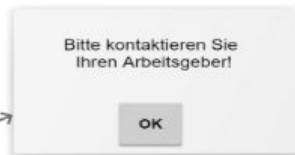
SmartLoadApp

Softwareentwicklungsprojekt

Geron Beli, Brice Nsikam Onla, Philipp Würfel, Alexander Schulz



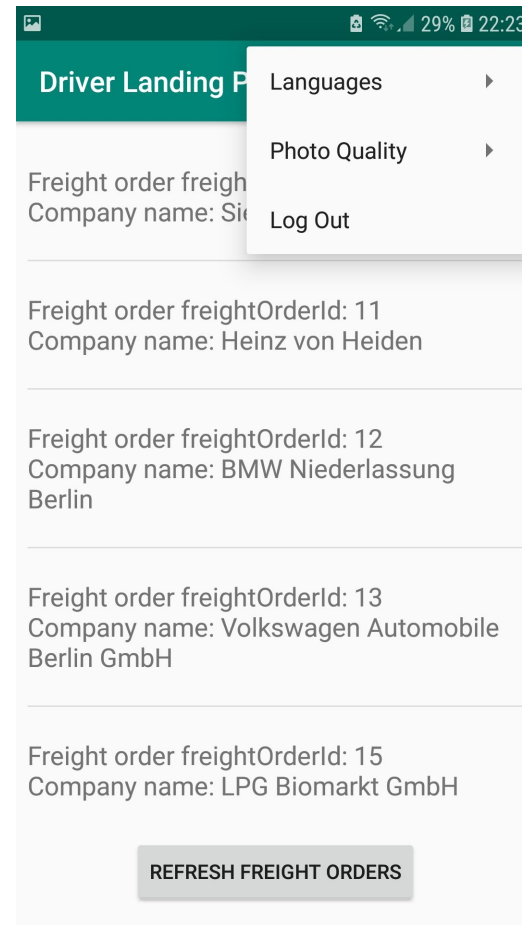
Mockup Anmeldung



App Anmeldung



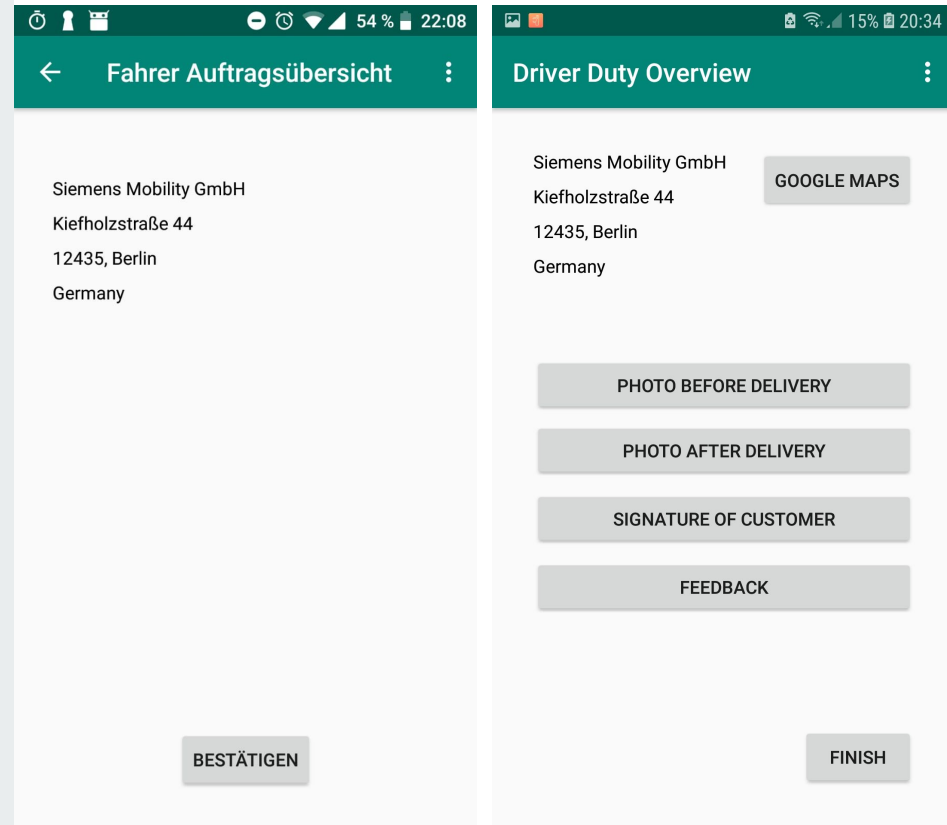
Mockup Auftragsübersicht



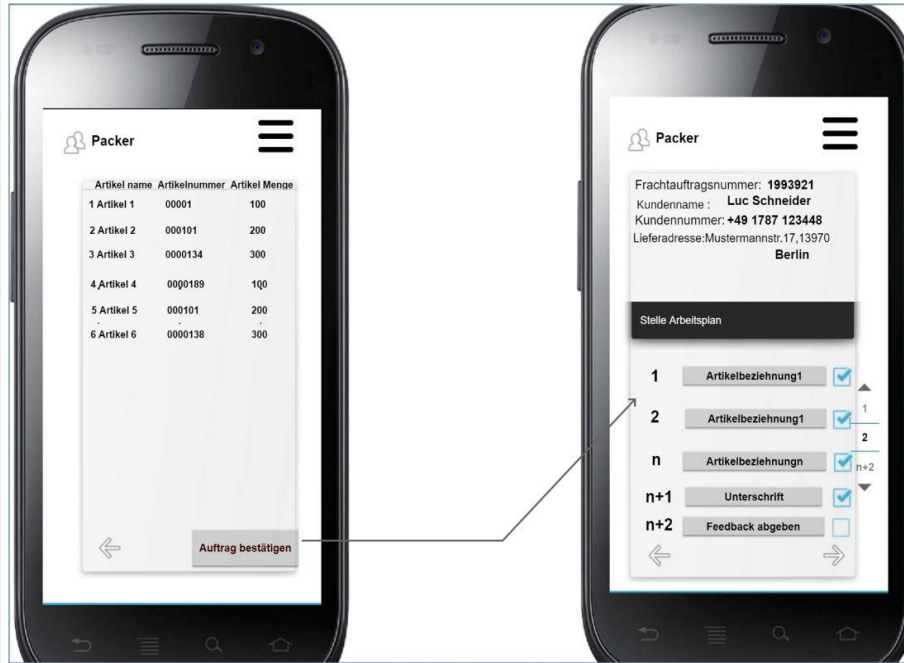
App Auftragsübersicht



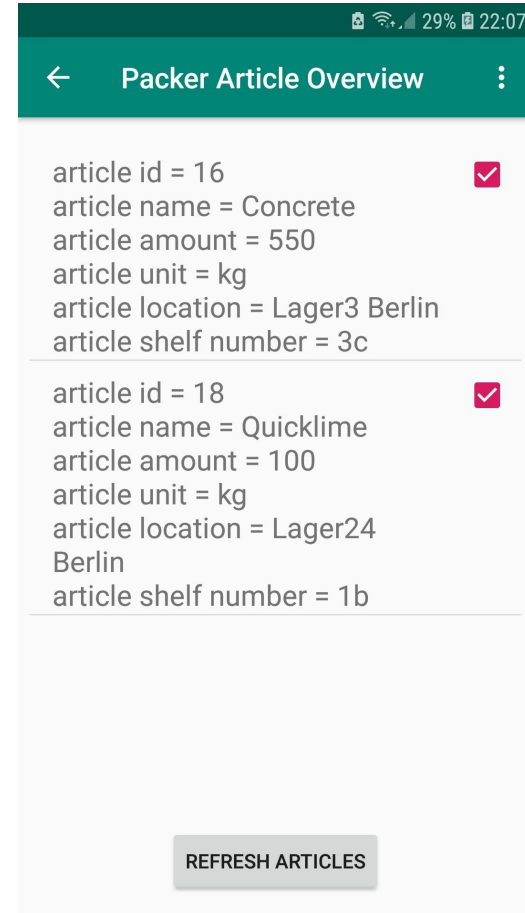
Mockup Auftragsbestätigung und Arbeitspaketübersicht



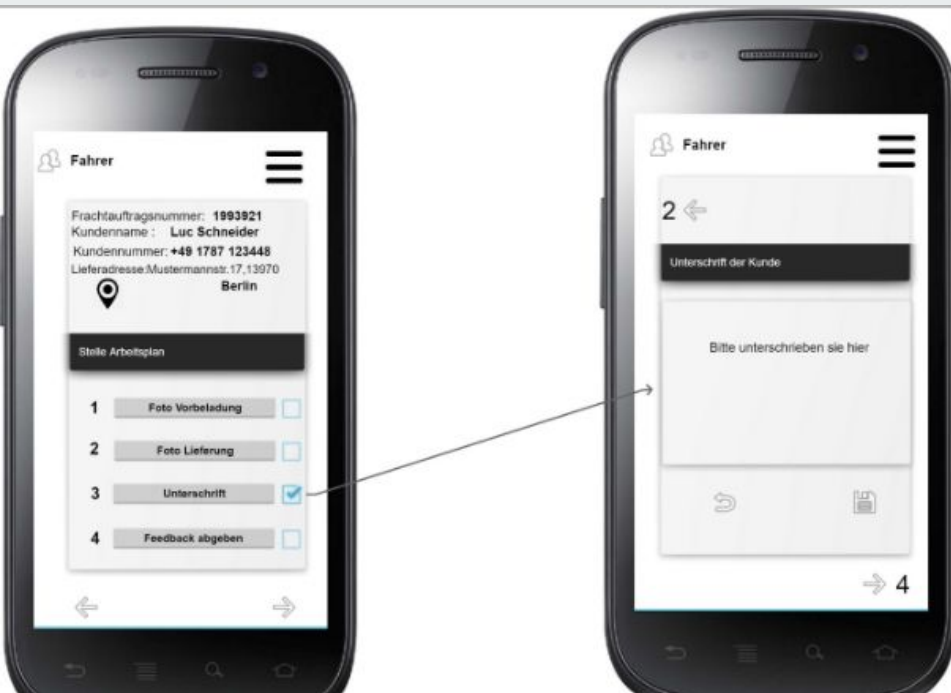
App Auftragsübersicht (Bestätigung) und Arbeitspaketübersicht



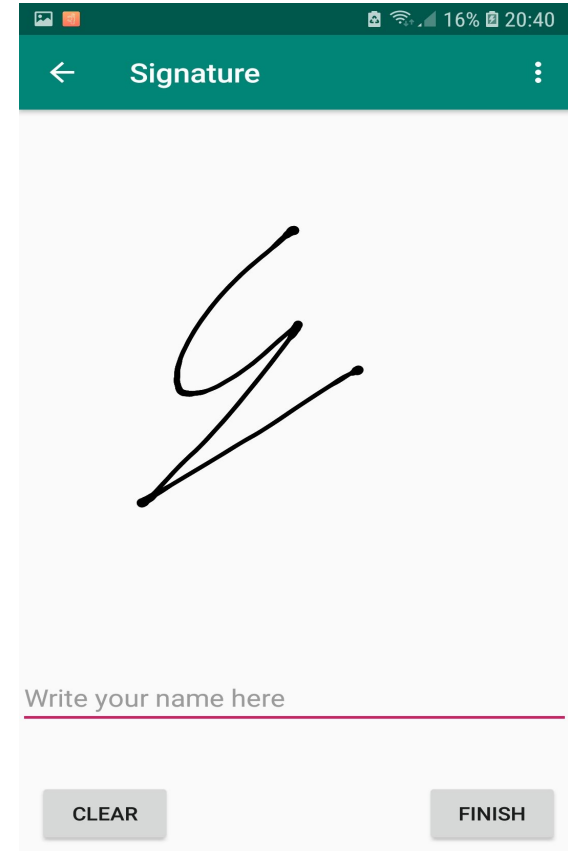
Mockup Artikelübersicht für Packer



App Artikelübersicht für Packer



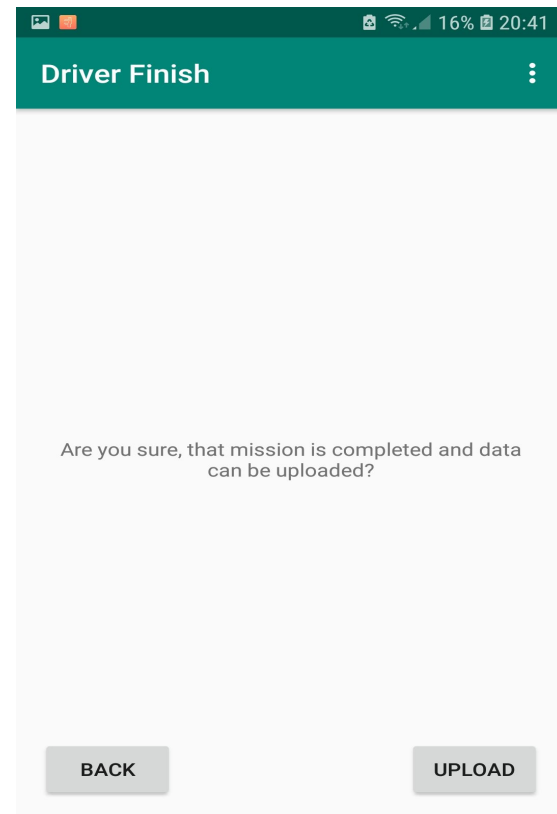
Mockup Bearbeitung Arbeitspaket "Unterschrift"



App Bearbeitung Arbeitspaket "Unterschrift"

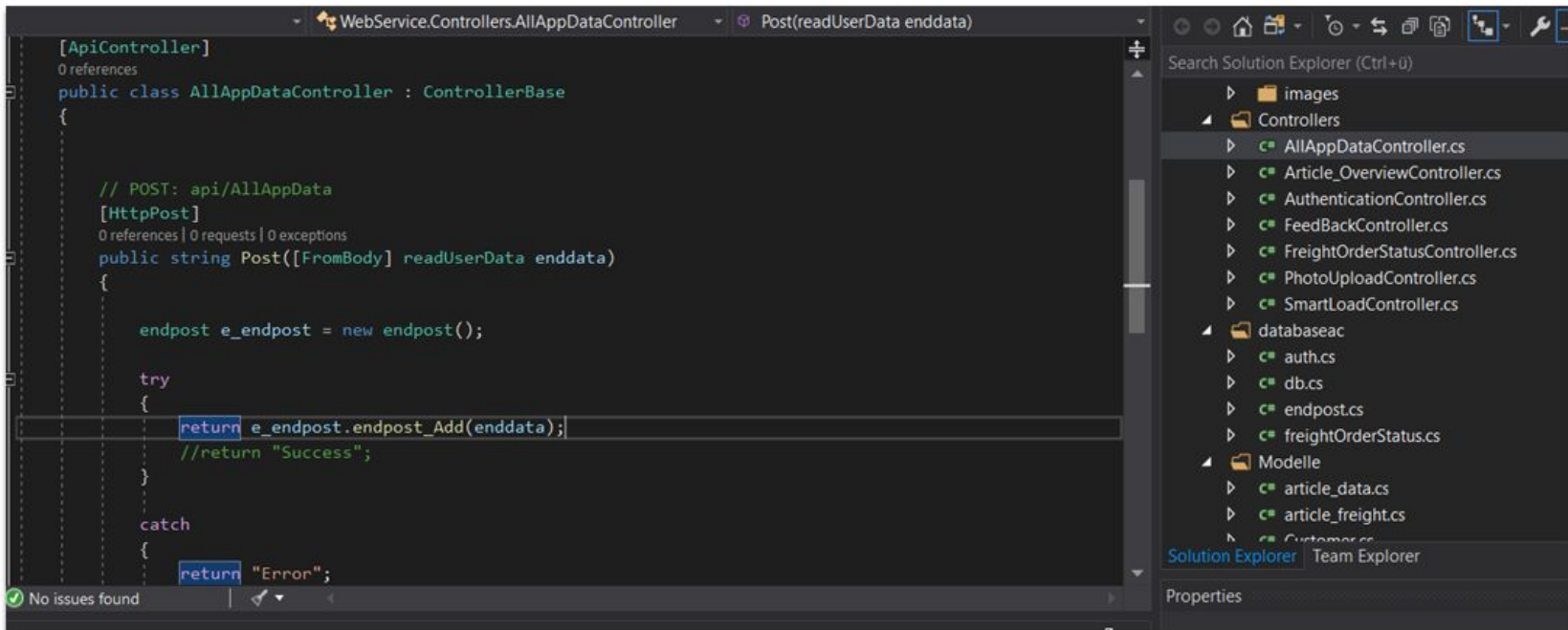


Mockup Abschluss des Frachtauftrages



App Abschluss des Frachtauftrags

Controller



The screenshot displays the Visual Studio IDE with a C# controller class and its project structure.

Controller Code:

```
[ApiController]
0 references
public class AllAppDataController : ControllerBase
{

    // POST: api/AllAppData
    [HttpPost]
    0 references | 0 requests | 0 exceptions
    public string Post([FromBody] readUserData enddata)
    {

        endpoint e_endpoint = new endpoint();

        try
        {
            return e_endpoint.endpoint_Add(enddata);
            //return "Success";
        }

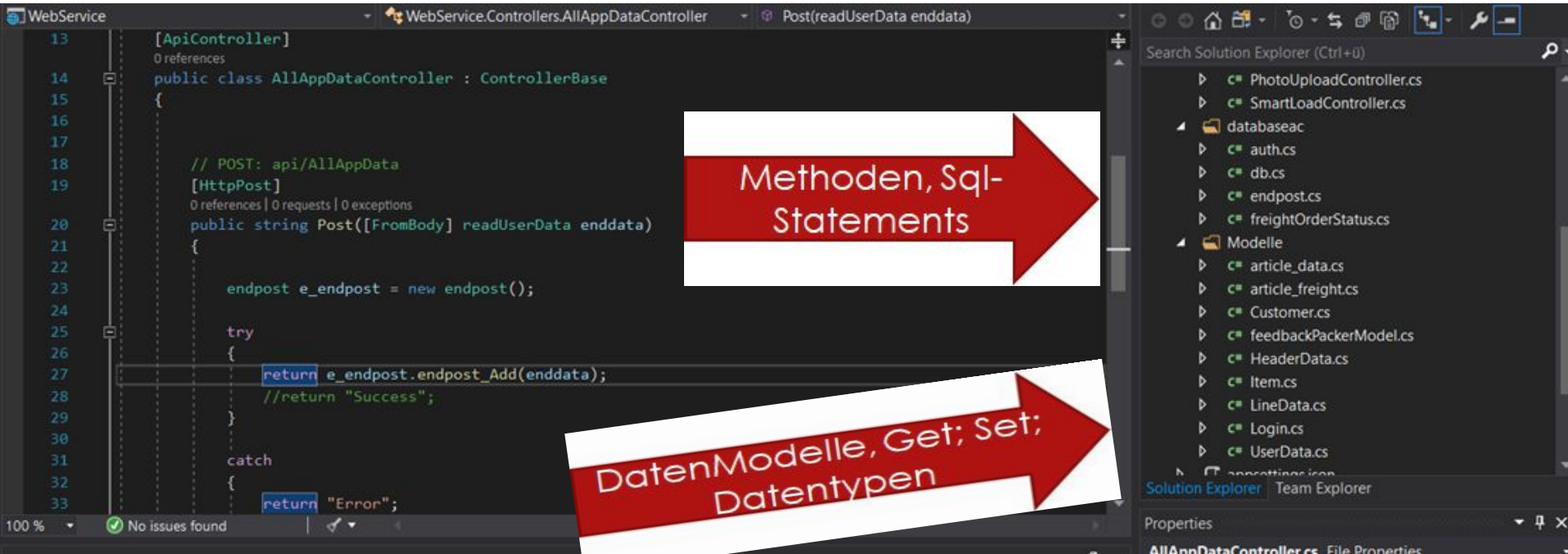
        catch
        {
            return "Error";
        }
    }
}
```

Solution Explorer:

- images
- Controllers
 - AllAppDataController.cs (selected)
 - Article_OverviewController.cs
 - AuthenticationController.cs
 - FeedBackController.cs
 - FreightOrderStatusController.cs
 - PhotoUploadController.cs
 - SmartLoadController.cs
- databaseac
 - auth.cs
 - db.cs
 - endpoint.cs
 - freightOrderStatus.cs
- Modelle
 - article_data.cs
 - article_freight.cs
 - Customer.cs

Properties: Properties

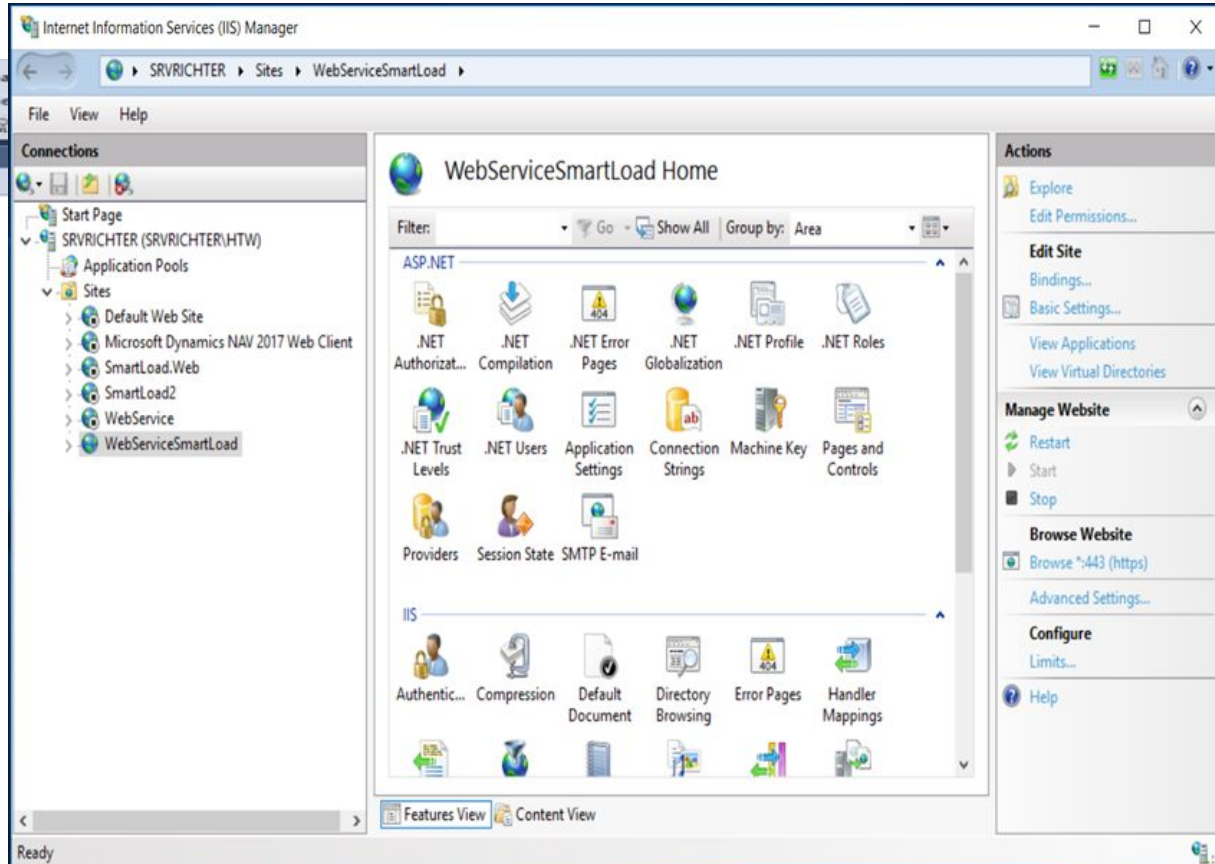
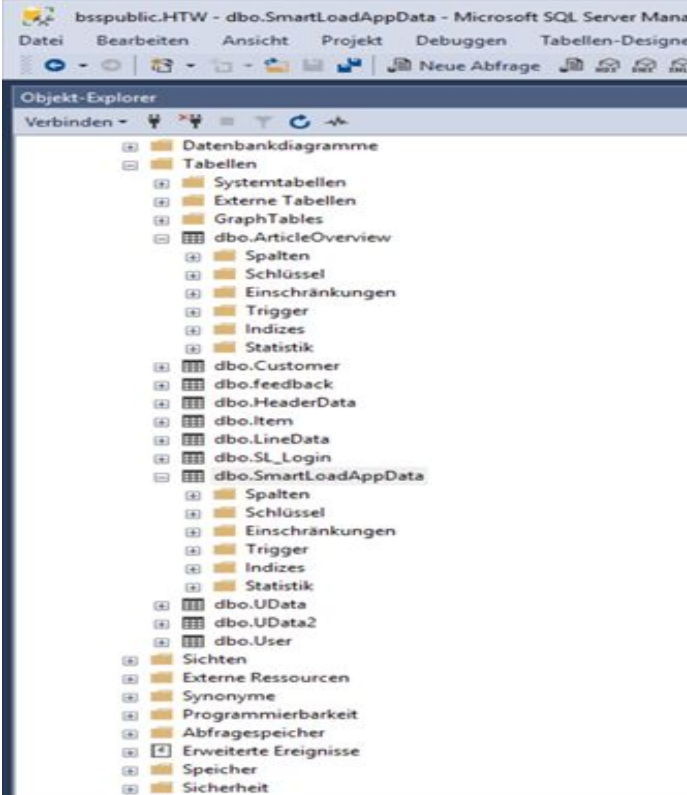
Weitere Klassen



The screenshot displays the Visual Studio IDE with the following components:

- Code Editor:** Shows the `AllAppDataController` class in `WebService.Controllers.AllAppDataController`. The code includes a `Post` method that interacts with an `endpost` object. Line 27 contains `return e_endpost.endpost_Add(enddata);` and line 33 contains `return "Error";`.
- Annotations:**
 - A red arrow pointing right with the text "Methoden, Sql-Statements" is overlaid on the code.
 - A red arrow pointing up and to the left with the text "DatenModelle, Get; Set; Datentypen" is overlaid on the code.
- Solution Explorer:** Located on the right, it shows a project structure with folders like `databaseac` and `Modelle`, and various files including `PhotoUploadController.cs`, `SmartLoadController.cs`, `auth.cs`, `db.cs`, `endpost.cs`, `freightOrderStatus.cs`, `article_data.cs`, `article_freight.cs`, `Customer.cs`, `feedbackPackerModel.cs`, `HeaderData.cs`, `Item.cs`, `LineData.cs`, `Login.cs`, and `UserData.cs`.

Datenbank, Server



Test: HttpGet

The screenshot shows the Postman application interface. The top bar includes the Postman logo, a menu (File, Edit, View, Help), and buttons for 'New', 'Import', 'Runner', and 'My Workspace'. The right side of the top bar has an 'Invite' button and a 'Sign In' button. Below the top bar, the left sidebar contains a search bar, a 'History' tab, and a list of recent requests. The main area displays a GET request to `http://localhost:16005/api/FreightOrders`. The 'Query Params' section is empty. The 'Body' tab is selected, showing a JSON response in 'Pretty' format. The response is a list of two freight orders. The status bar at the bottom indicates 'Status: 200 OK', 'Time: 662 ms', and 'Size: 10.83 KB'.

GET `http://localhost:16005/api/FreightOrders` Send Save

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (6) Test Results Status: 200 OK Time: 662 ms Size: 10.83 KB Download

Pretty Raw Preview JSON ...

```
1 [
2   {
3     "freightOrderId": "1",
4     "companyName": "Kaufland",
5     "description": "sometext",
6     "freightOrderStatus": "ON_PACKING",
7     "timestamp": "1562509625583",
8     "postalCode": "10319",
9     "country": "Germany",
10    "city": "Berlin",
11    "streetName": "Sewanstraße",
12    "streetNumber": "15",
13    "signatureTagPacker": "1__1562505082962.jpg",
14    "feedbackPacker": "Weiterer Test2222222222222222",
15    "photoTagsPacker": "1_Packer_1562505078226.jpg",
16    "signatureTagDriver": "1_CheckCheck_1562505161393.jpg",
17    "feedbackDriver": "comment",
18    "photoTagsDriver": "1_Driver_1562505145900.jpg",
19    "userId": "Packer"
20  },
21  {
22    "freightOrderId": "10",
23    "companyName": "Siemens Mobility GmbH",
24    "description": "sometext",
```

Test: HttpPost

The screenshot displays the Postman application interface. The top bar includes the Postman logo, menu options (File, Edit, View, Help), and workspace management tools (New, Import, Runner, My Workspace, Invite, Sign In). The left sidebar shows a history of requests, including a collection of API endpoints. The main workspace is configured for a POST request to `http://localhost:16005/api/Photoupload/upload`. The request body is set to `form-data` and contains a single key-value pair: `file` with the value `file (2).jpg`. The response is displayed in the bottom pane, showing a `200 OK` status, a response time of `25 ms`, and a size of `363 B`. The response body is a JSON object: `{"length": 125990, "name": "file (2).jpg"}`.

Postman

File Edit View Help

+ New Import Runner

My Workspace Invite

No Environment

Filter

History Collections APIs BETA

Save Responses Clear all

POST http://localhost:16005/api/Photoupload/upload

Send Save

Params Authorization Headers (11) Body Pre-request Script Tests

none form-data x-www-form-urlencoded raw binary GraphQL BETA

KEY	VALUE	DESCRIPTION
file	file (2).jpg	
	Select Files	
Key	Value	Description

Body Cookies Headers (6) Test Results

Status: 200 OK Time: 25 ms Size: 363 B Download

Pretty Raw Preview JSON

```
1 {
2   "length": 125990,
3   "name": "file (2).jpg"
4 }
```

Internet Explorer

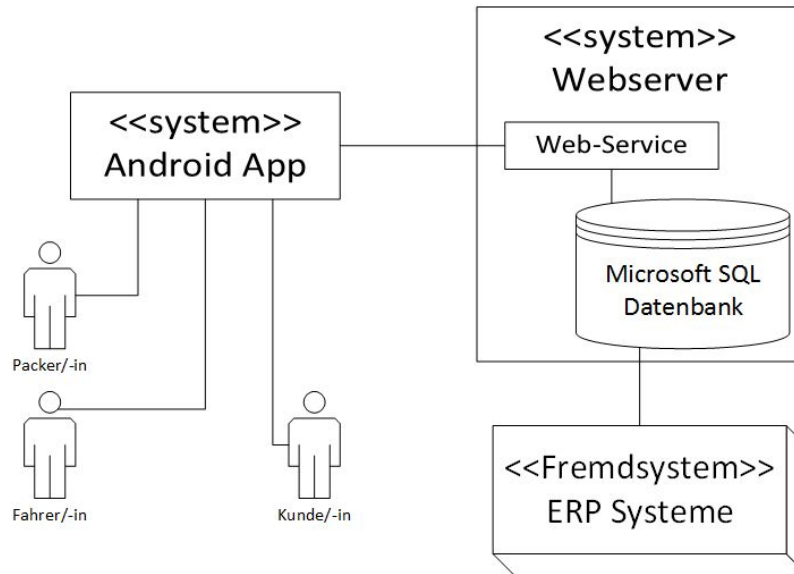
Bootcamp



Qualitätsanforderungen

Anforderung	Beschreibung
Benutzerfreundlichkeit	Die Bedienung soll möglichst intuitiv und die Software ohne Hilfe oder Tutorials einfach nutzbar sein.
Stabilität	Die App soll zu jeder Zeit flüssig und stabil laufen. Um das zu garantieren, werden die Funktionalitäten innerhalb der App möglichst minimal und zweckorientiert gehalten.
Zeitverhalten	Die Kommentar- und Fotografier Funktionen, sowie das Hochladen- und Herunterladen der Lieferscheindaten soll direkt und schnell erfolgen.

Externe Schnittstellen



Schnittstellen

- Input/Output zwischen App und Webserver
- SQL-Datenbank auf Server
- Keine Schnittstelle zu ERP-Systemen

Daten und Formate

- .jpg Für Kamerabilder
- .json Nur für Input/Output zwischen App und Webserver
- .blop Bilder auf Server als File abgelegt, Feedback in Json als String

Soll-Ist Vergleich



Anwendungsfälle (Soll)

1. Authentifizierung
2. Frachtauftrag wahrnehmen
3. Frachtauftrag abschließen
4. Folgeauftrag zum Lieferschein generieren
5. Fracht vorbereiten
6. Fracht abfotografieren
7. Fracht kommentieren
8. Unterschreiben
9. Authentifizierung aus App validieren
10. Lieferscheindaten an App senden
11. Lieferscheindaten von App empfangen
12. Navigation öffnen
13. Einstellungen konfigurieren
14. Abmeldung
15. Sprache wechseln
16. Bildgröße der Kamera wählen
17. Link zu Webserver setzen

Umsetzung (Ist)

1. Vollständig und planmäßig umgesetzt
2. Vollständig und planmäßig umgesetzt
3. Vollständig und planmäßig umgesetzt
4. Nicht umgesetzt, weil nachträglich für unwichtig erklärt
5. Vollständig und planmäßig umgesetzt
6. Umgesetzt (für n-viele Fotos)
7. Vollständig und planmäßig umgesetzt
8. Vollständig umgesetzt (der Name wird zusätzlich eingetragen)
9. Vollständig und planmäßig umgesetzt
10. Vollständig und planmäßig umgesetzt
11. Vollständig und planmäßig umgesetzt
12. Vollständig und planmäßig umgesetzt
13. Vollständig umgesetzt (Webserverlink wird an der Startseite eingestellt)
14. Vollständig und planmäßig umgesetzt
15. Vollständig und planmäßig umgesetzt
16. Vollständig und planmäßig umgesetzt
17. Webserverlink wird an der Startseite eingestellt

Soll-Ist Vergleich

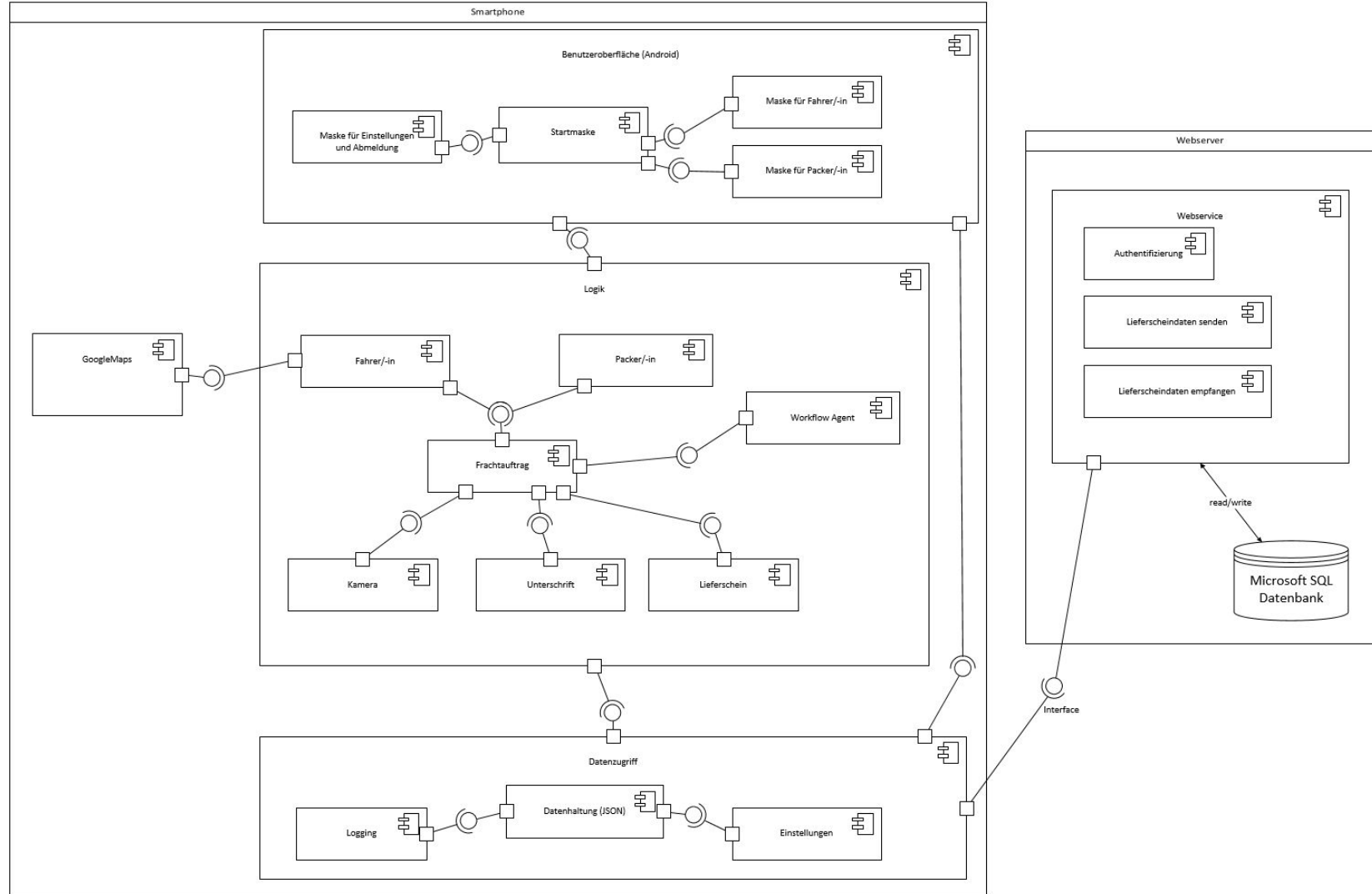


Kriterien (Soll)

1. Anmeldung (innerhalb von 5 Sekunden)
2. Abmeldung (innerhalb von 3 Sekunden)
3. Automatische Anmeldung (auch nach Neustart der App)
4. Fehlermeldung bei falschen Anmeldedaten
5. Rollenabhängige Auftragsübersicht nach Anmeldung
6. Wahrgenommene Aufträge können abgebrochen werden
7. Frachtaufträge können nicht doppelt angenommen werden
8. App besitzt zwei Masken in Abhängigkeit von Rolle
9. Fotos sind vor dem Hochladen in einer Vorschau sichtbar
10. Navigation zum Lieferort per Google Maps

Umsetzung (Ist)

1. Ja: in weniger als 5 Sekunden
2. Ja: in weniger als 3 Sekunden
3. Teils: Nur Speicherung der Login-Daten
4. Ja: MessageBox ersetzt durch Toast
5. Ja: Zusatzinformationen in weiterer Aktivität ausgelagert
6. Ja: Abbruch erfolgt ohne Warnfenster
7. Ja: Synchronisation erfolgt direkt (< 60 Sekunden)
8. Ja
9. Ja: Vorschau wurde in Fotoaktivität umgelagert
10. Ja





Derzeit offene Schwachstellen

- Bug: Artikelübersicht CheckBox ändert Status beim Scrollen und Schließen der Aktivität automatisch auf false
- Zustand der App wird nicht zwischengespeichert (Problemfall bei Absturz oder Schließen der App)
- Keine Offlinefähigkeit: Synchronisation funktioniert nur bei aktiver Internetverbindung