Audit 4 zu JetSet Food

Entwicklungsprojekt WS2020/21

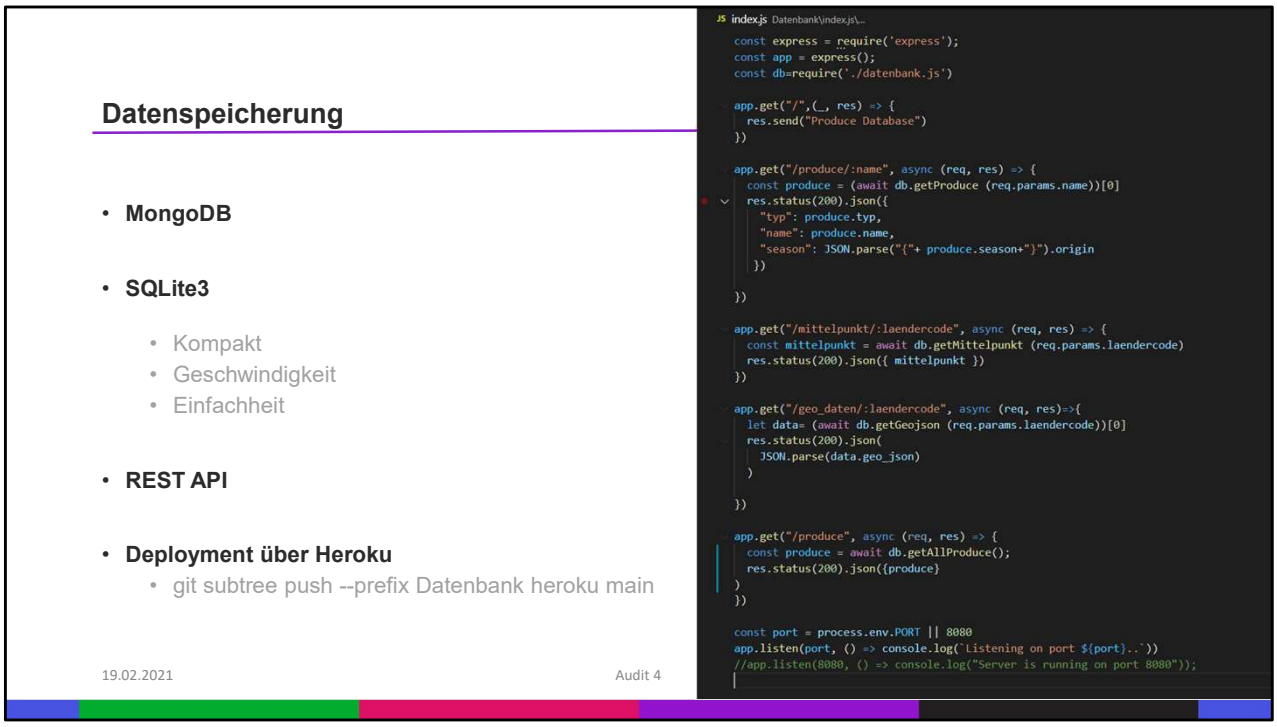
Sophia Johannsen
Markus Mohr-Dama

19.02.2021

Inhalt

Datenbank

Audit 4



MongoDB:
Die ursprüngliche Idee einer Synchronen Datenspeicherung mit MongoDB wurde verworfen, da die umfangreichen Funktionen für den momentanen Stand des Projektes nicht erforderlich sind

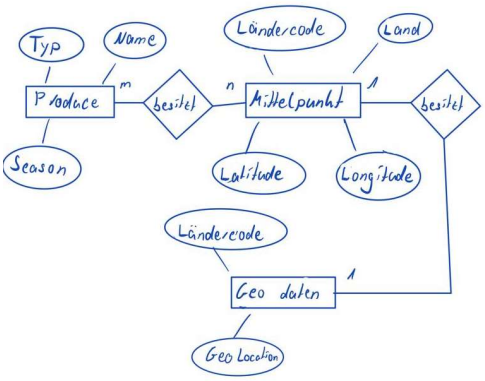
SQLite3:
Open Source, Relationales Datenbankenmanagement System
Eine Datei Pro erstellte Datenbank, Bibliothek ist nur wenige hunderte Kilobyte groß
Keine Separate Serververbindung notwendig, Dateien werden Lokal abgegriffen

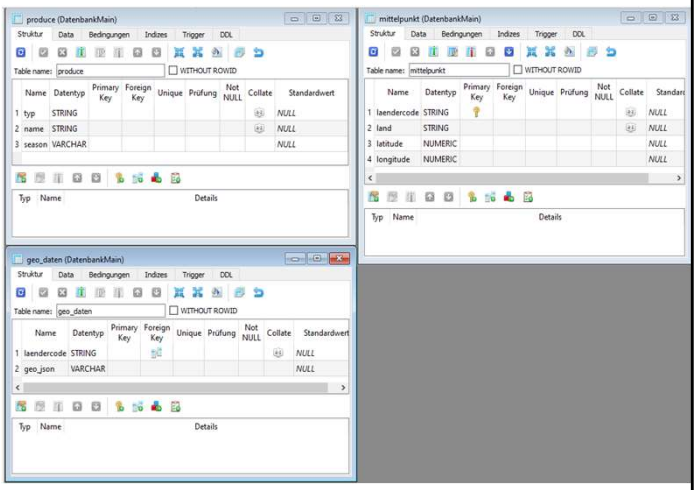
REST API
Representational State Transfer, ist ein Paradigma für die Softwarearchitektur von verteilten Systemen, speziell für Webservices
API Schnittstelle von Maschine zu Maschine.
Durch REST HTTP anfragen möglich.

Deployment über Heroku
Heroku – 2010 aufgekauft von Salesforce, Plattform-as-a-service bietet verschiedene Möglichkeiten der Bereitstellung für Apps und Web Dienste. Kostenlose Version möglich – bei Wachstum der App ist das hinzubuchen von Ressourcen kostenpflichtig möglich.
Funktioniert mit der GIT Versionskontrolle. In Heroku wird eine App erzeugt die als Container für die Bereitzustellende App fungiert. Schließend ließt Heroku die Package.json Datei aus, die von Node.js erstellt wird und alle Module Informationen enthält. Außerdem benötigt Heroku noch eine Procfile in der hinterlegt ist
mit welchem Befehl die App startet. Über die GIT Konsole kann sich mit dem Benutzerdaten von Heroku eingeloggt werden und mit dem Befehl „git subtree push --prefix Datenbank heroku main“ wird der Unterordner „Datenbank“ in den App Container von Heroku gepusht.

Datenbankmodelle

ERM zur Jet Set-Food Datenbank





19.02.2021 Audit 4 5

Auf der linken Seite ist das Entity-Relationship-Modell zu sehen, in dem werden die Beziehungen der einzelnen Tabellen zueinander dargestellt und welche Entitäten zugeordnet sind.

Auf der rechten Seite ist ein Screenshot aus dem Programm SQLiete Studio zu sehen, in dem das Relationenmodell abgebildet ist. Das Relationenschema bestimmt den Datentyp und den Namen der Spalten

REST Methodentabelle

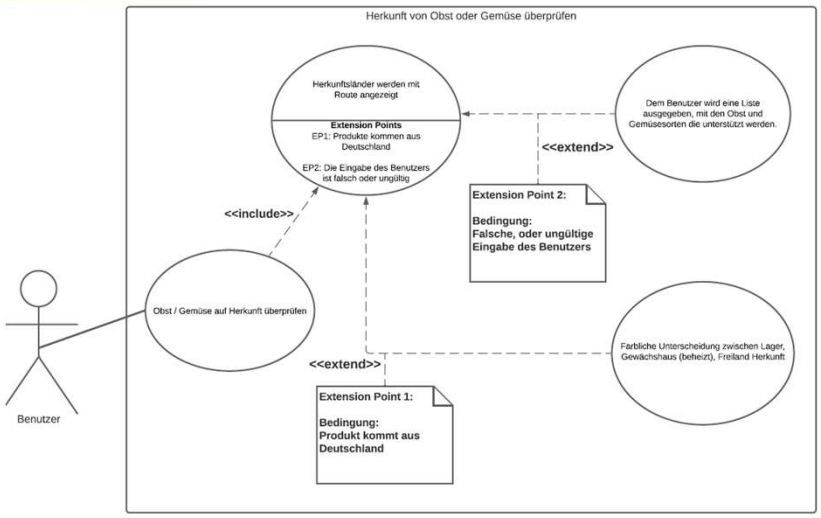
Ressource	Methode	Semantik	Content (Type Res.)	Statuscode (Success)	URI
Produce	GET	Greift auf die Repräsentation der Listenressource Produce zu.	JSON	200 OK	/GET /produce
Produce Name	GET	Greift auf die Repräsentation der Einzelressource Produce eines individuellen Namens zu.	JSON	200 OK	/GET /produce/(name)
Mittelpunkt Ländercode	GET	Greift auf die Repräsentation der Einzelressource Mittelpunkt eines individuellen Ländercode zu.	JSON	200 OK	/GET/mittelpunkt/(laendercode)
Geo Daten Ländercode	GET	Greift auf die Repräsentation der Einzelressource Geo-Daten eines individuellen Ländercodes zu.	JSON	200 OK	/GET/geo_daten/(laendercode)

In der REST Methodentabelle sind alle Methoden definiert, mit denen auf die Datensätze der Datenbank zugegriffen werden kann.
Die App benötigt ausschließlich GET Methoden.

Der Content Typ beschreibt das Ausgabeformat. Bei unsere App ist das JSON.

Mit der URI /GET/produce wird eine Liste aller Einträge der Tabelle „Produce“ ausgegeben. Mit der Zugabe des Parameter (name) kann gezielt nach einer Einzelressource Name gesucht werden,
Die URI /GET/mittelpunkt/(laendercode) erzeugt eine Ausgabe des Mittelpunkts gefiltert nach dem eingegeben Ländercode.
Die URI /GET/geo_daten/(laendercode) erzeugt eine Ausgabe der Geo Daten gefiltert nach dem eingegeben Ländercode.

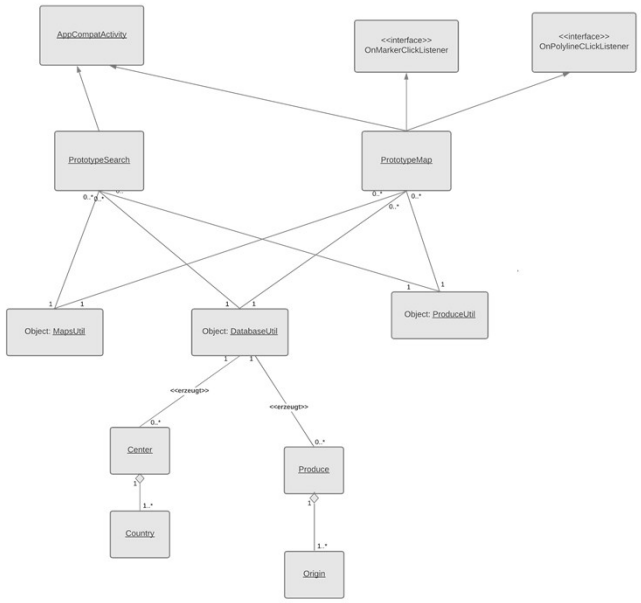
Anwendungsfalldiagram



In diesem Anwendungsfalldiagramm wir der Systemkontext „Herkunft von Obst und Gemüse überprüfen visualisiert und Programmunabhängig dargestellt.
Der erste Anwendungsfall Obst und Gemüse auf Herkunft überprüfen inkludiert den Anwendungsfall „Herkunftsländer werden mit Route angezeigt“ mit den beiden Extension Points EP1: Produkte kommen aus Deutschland Und EP2: „Die Eingabe des Benutzers ist falsch oder ungültig“
EP1 Tritt nur ein, wenn die Bedingung: „Produkt kommt aus Deutschland“ erfüllt ist
EP2 Tritt nur ein, wenn die Bedingung: „Falsche oder ungültige Eingabe des Benutzers“ erfüllt ist.


Prototyp
Audit 4

Klassendiagramm



Das Klassendiagramm zu dem Prototypen. Eine vollständige Version mit Klassenvariablen und Funktion befindet sich im Repository.
Der Prototyp wurde durch die Verwendung zweier Activitys, also Klassen mit denen der Nutzer über eine Graphical User Interface kommuniziert, realisiert. Beide Activities erben von der Klasse AppCompatActivity, einer Bibliotheksklasse in der die Funktionen zur Interaktion mit dem Nutzer bereitgestellt werden. Die Kasse PrototypeMap implementiert zudem noch zwei Interfaces. Diese ermöglichen es OnClickListener, also Funktionen die bei Klick-Ereignissen auf Polylines und Marker aufgerufen werden zu implementieren. Die Anfragen und Interaktionen werden mithilfe dreier Objekte realisiert. Diese Objekte sind Instanzlos und implementieren so das Singleton Entwurfsmuster.

Die Daten aus der Datenbank werden in Datenklassen gespeichert, welche von dem DatenbankUtil Objekt erzeugt werden. Es wurde sich für Datenklassenobjekte und ein externes Util Objekt zur Interaktion mit diese entschieden, da die Antworten der Datenbank null enthalten können und so die NullChecks und deren Behandlung in den Util-Funktionen vorgenommen werden können. Somit liegen diese nicht in der Verantwortung der Funktionsnutzer

Codebeispiele – Interaktion mit der Datenbank

```
suspend fun getOrigin(origins: List<String>): Pair<List<Country>, List<String>>? =
    try {
        var regional = listOf<String>()

        //Überprüft, ob Anbauarten in der Origins Liste vorhanden sind
        if (origins.contains("DEU")) {
            regional =
                origins.fold(listOf()) { acc, cur -> if
(ProduceUtil.farmingMethod.contains(cur.toLowerCase())) acc + cur.toLowerCase() else acc }
        }
        val countryStrings = makeQuery(origins, "mittelpunkt").map {
            GlobalScope.async { callDatabase(it) }.await()
        }
        val curCountries = countryStrings.map {
            (processResponse(it, ProduceUtil.countryType) as Center).mittelpunkt[0]
        }
        Pair(curCountries, regional)
    } catch (e: Exception) {
        Log.e("API", "Country Query failed", e)
        null
    }
```

Funktion die eine Liste von Länderkürzeln und eventuell Anbaumethoden entgegenimmt und diese aufteilt in eine Liste mit Länderobjekten und eine mit Anbaumethoden.
Zunächst werden die Anbaumethoden herausgefiltert. Dann wird die verbleibenden einzeln durchgegangen und für jeden Ländercode eine URI für GET-Anfrage erzeugt und diese dann an die Datenbank gesendet. Daraus resultiert nun eine Liste voller String Objekte welche nun in eine Liste voller Mittelpunktsobjekte konvertiert werden. Schließlich wird ein Paar aus der Mittelpunksliste und der Anbaumethoden zurückgegeben.
Falls ein Fehler geschieht, wird null zurückgegeben und dieser im Log gespeichert.

Erzeugung neuer Markericons

```
private fun vectorToBitmap(
    @DrawableRes id: Int,
    context: Context,
    scaling: Int = 11
): BitmapDescriptor {
    val vectorDrawable: Drawable? = ResourcesCompat.getDrawable(context.resources, id, null)
    if (vectorDrawable == null) {
        Log.e("Markericon", "Resource not found")
        return BitmapDescriptorFactory.defaultMarker()
    }
    val bitmap = Bitmap.createBitmap(
        vectorDrawable.intrinsicWidth / scaling,
        vectorDrawable.intrinsicHeight / scaling, Bitmap.Config.ARGB_8888
    )
    val canvas = Canvas(bitmap)
    vectorDrawable.setBounds(0, 0, canvas.width, canvas.height)
    vectorDrawable.draw(canvas)
    return BitmapDescriptorFactory.fromBitmap(bitmap)
}
```

Eine Funktion welches aus einer Scalable Vector Graphic eine Bitmap erzeugt. Die SVG muss sich drawable Ressourceordner befinden, und ihre ID an die Funktion übergeben werden. Die Bitmap wird mit einer Farbtiefe von 8Bit erzeugt und erhält die skalierte Höhe und Breite der SVG. Diese wird nun mit einem Canvasobjekt verknüpft, welches die SVG in die Bitmap schreibt. Aus dieser Bitmap wird schließlich ein BitmapDescriptor Objekt erschaffen, welches zurückgegeben wird.

Interaktion mit der Karte

```
fun addOriginClustered(
map: GoogleMap,name: String="Deutschland",
position: LatLng = germany, flaeche: Int = germanyArea,
context: Context,
markerManager: MarkerManager, polylineManager: PolylineManager) {
    map.addMarker(
        MarkerOptions()
            .position(position)
            .title(name)
            .icon(vectorToBitmap(R.drawable.ic_pinhomedark, context))
    )
    val layerGermany = GeoJsonLayer(
        map, flaeche, context, markerManager, PolygonManager(map),
        polylineManager, GroundOverlayManager(map)
    )
    layerGermany.defaultPolygonStyle.strokeWidth = 0.0f
    layerGermany.defaultPolygonStyle.fillColor = Color.LTGRAY
    layerGermany.addLayerToMap()
}
```

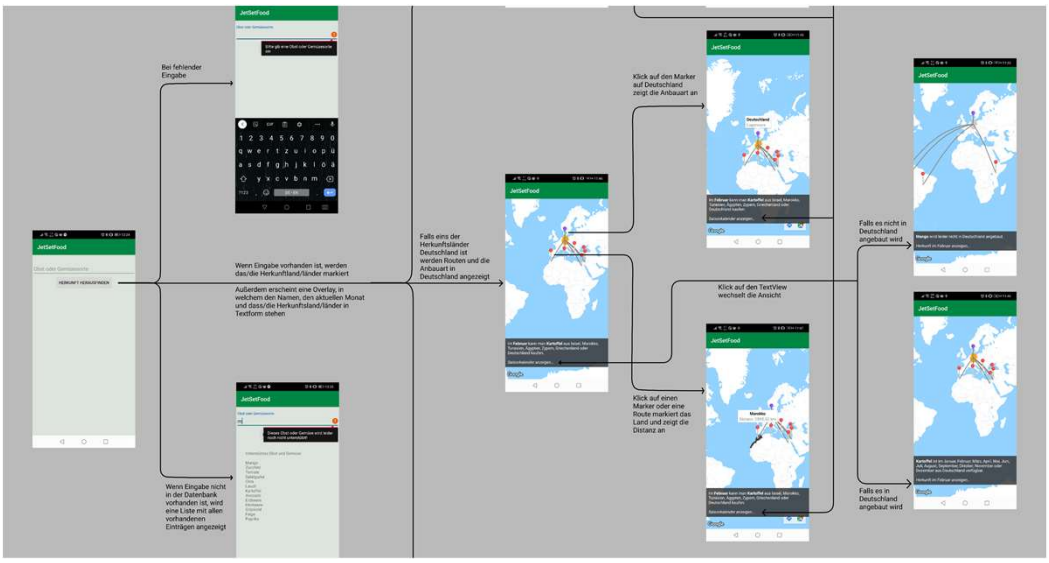
Funktion welche das Heimat/Aufenthaltsland markiert. Da momentan von der Applikation nur Deutschland als dieses unterstützt wird, sind diese Werte hier als default Werte festgelegt. Der Mittelpunkt des Landes wird mit einem Marker der den Namen des Landes enthält markiert. Dieser Marker erhält ein Icon, welches von der vorherigen Funktion in einen BitmapDescriptor umgewandelt wird.
Als nächstes wird die Fläche von Deutschland markiert. Dies geschieht als GeoJSON layer, welche einen Marker- und einen PolylineManager erhält. Diese Manager enthalten alle Polyline und Markerobjekte auf der Karte welche ein besonderes Klickereignisverhalten besitzen sollen. So kann gewährleistet werden das dieses Verhalten auch über der GeoJSON-Layer ausgeführt werden kann.

Interaktion mit dem Nutzer

```
button.setOnClickListener{
    if(produceInput.text.toString().isEmpty())
        produceInput.error=("Bitte gib eine Obst oder Gemüsesorte ein")

    else if(!produceList.map { it.toLowerCase()
}.contains(produceInput.text.toString().toLowerCase())) {


        produceInput.error ="Dieses Obst oder Gemüse wird leider noch nicht unterstützt!"
        textView.text= getString(
        R.string.produceListe, ProduceUtil.makeString(produceList,"\n", null)
        )
        textView.visibility= View.VISIBLE

    } else {

        startActivity(Intent(this, PrototypeMap::class.java)
        .putExtra("input", produceInput.text.toString()))
```

Hier ist der OnClickListener des Knopfes auf der ersten Activity gezeigt. Ein Klickereignis löst eine Überprüfung des eingegebenen Textes aus. Falls der Text nicht in der Liste aller Produkte vorhanden ist, werden dem Nutzenden alle unterstützten Produkte angezeigt. Hierfür wird eine Stringressource verwendet. Falls die Eingabe vorhanden ist, wird die zweite Activity aufgerufen und die Eingabe übergeben.

Userjourney

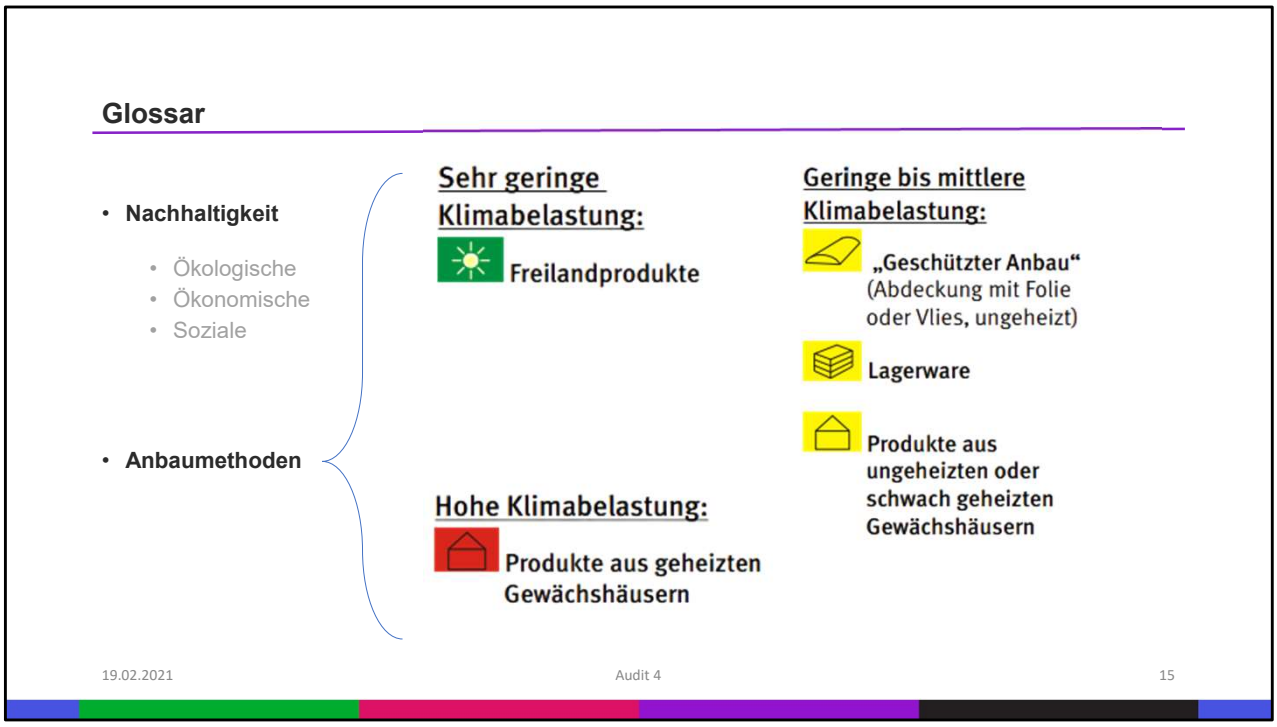


Ein Ausschnitt der Userjourney: Der Nutzende gibt eine Obst/Gemüse Sorte und erhält eine Karte, auf der Detuschland farblich markiert ist und die Routen und Mittelpunkte der anderen Herkunftsländer eingezeichnet sind. Außerdem steht im unteren Bildschirmbereich eine Aufzählung aller Herkunftsländer.

Bei Klick auf Deutschland erscheint ein Infofenster, in welchem die Anbaumethode in Deutschland steht. Dies ist nur in dem Fall, dass das Produkt in Deutschland angebaut wird vorhanden.

Bei Klick auf eine andere Markierung oder eine Route markiert dieses Land und zeigt den Namen und die Distanz zu Deutschland an. Bei Klick auf eine andere Markierung verschwindet dies wieder.

Bei Klick auf den Text im unterem Bildschirmbereich erscheint, falls vorhanden, eine Aufzählung der Monate, in denen das Produkt in Deutschland angebaut wird. Falls dies nicht vorhanden ist erscheint ein Hinweis dass dieses Produkt nicht in Deutschland angebaut wird. Bei einem erneutem Klick erscheint wieder die Liste der Herkunftsländer.



•Ökologisch: Ressourcen sollen nur in dem Ausmaß konsumiert werden, wie die Regeneration der Natur es erlaubt. Dadurch soll Artenvielfalt mit natürlichen Lebensgrundlagen für Folgegenerationen gewahrt werden.
Beispiel: Schutz von Insekten.
•Ökonomische: Leistungsfähige Wirtschaft welche Folgegenerationen keine Probleme hinterlässt soll gewährleistet werden.
Beispiel: Hilfszahlungen für Unternehmen, welche durch Pandemieeinschränkende Maßnahmen schließen mussten
•Soziale: Chancengleichheit soll gewährleistet werden. Dies bezieht sich auf Zugang zu Bildung, Wohlstand und Kultur.
Beispiel: Eliminierung moderner Sklaverei, Gewährleistung des Rechts auf körperliche Unversehrtheit
Da Ökonomische und Soziale Nachhaltigkeit stark von Ökologischer Nachhaltigkeit beeinflusst werden, steht diese im Vordergrund. Beispielsweise hat die globale Klimaveränderung durch, unter Anderem, Missernten oder der Anstieg des Meeresspiegel direkte soziale und ökonomische Folgen.

Freilandprodukte

Im Freilandanbau werden mengenmäßig bedeutende Gemüsearten wie Zwiebeln, Möhren, Kohl und Salate angebaut. Im Winterhalbjahr fällt der Selbstversorgungsgrad bei diesen Produkten gegenüber den Sommermonaten daher deutlich ab.
Die Klimabelastung bei Freilandprodukten ist um ein vielfaches geringer. Der Freilandanbau ist besonders für widerstandsfähiges Gemüse geeignet.

geschützter Anbau

Beim geschützten Anbau wird das Anbau Gut im Freiland mit einer Folie oder einem Vlies abgedeckt. Die jungen Pflanzen werden so vor Witterung oder Schädlingen geschützt. Dadurch ist eine Reduzierung des chemischen Pflanzenschutzes möglich. Dennoch besteht eine höhere Klimabelastung aufgrund der Verwendeten Abdeckungen. Nicht immer kann diese Wiederverwendet werden, sodass neue Folie oder Vlies produziert werden muss.

Lagerware

Das Einlagern von Obst und Gemüse kommt nur in Frage, wenn das Produkt nach dem Ernten nur langsam nachreift, oder die Reifung durch Lagetemperatur sowie relative Luftfeuchtigkeit kontrollierbar ist. Außerdem ist der Erntezeitpunkt entscheidend. Die Klimabelastung entsteht durch die Aufrechterhaltung bestimmter Umweltbedingungen in den Lagerräumen, sowie durch den vermehrten Transport der Güter.

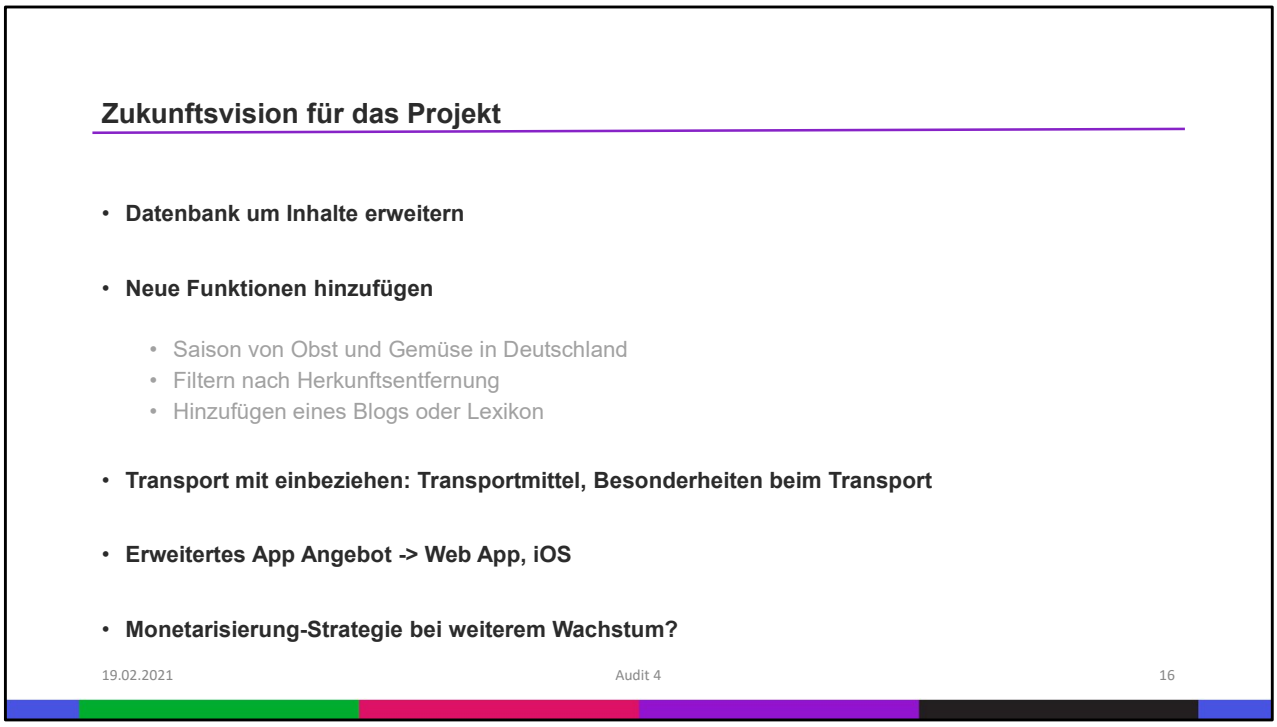
ungeheizte oder schwach geheizte Gewächshäuser

Das un- oder schwachgeheizte Gewächshaus ist ein erweiterte Form des geschützten Anbaus. Jedoch muss bei einem Gewächshaus nicht jährlich die Abdeckung erneuert werden. Außerdem besteht durch Heizen die Möglichkeit einer Klima Kontrolle innerhalb des Gewächshauses, sodass beispielsweise Frost gezielt vermieden werden kann.

geheizte Gewächshäuser

Bei frischem Gemüse oder frischem Obst außerhalb der Saison ist der Energiebedarf zur Energieerzeugung mit Hilfe von beheizten Gewächshäusern enorm.
Mehr als 90% des Energieverbrauchs für beheizte Gewächshäuser in Deutschland wird nach wie vor durch Öl- und Erdgasheizung gedeckt.

Der Energieverbrauch im Vergleich zum Freiland ist 10 bis 50 mal höher und die Klimaschädlichen Emissionen um das 5- bis 30-Fache.



Datenbank weitestgehend mit allen Obst- und Gemüsesorten zu komplettieren.

Filtern nach Herkunftsentfernung, das durch einen farblich markierten Kreis auf der Karte visuell dargestellt wird.

Obst und Gemüse anzeigen zu lassen, dass zum gegenwärtigen Zeitpunkt des Nutzers Saison in Deutschland hat, um den Benutzer noch besser bei seiner Kaufentscheidung zu unterstützen.

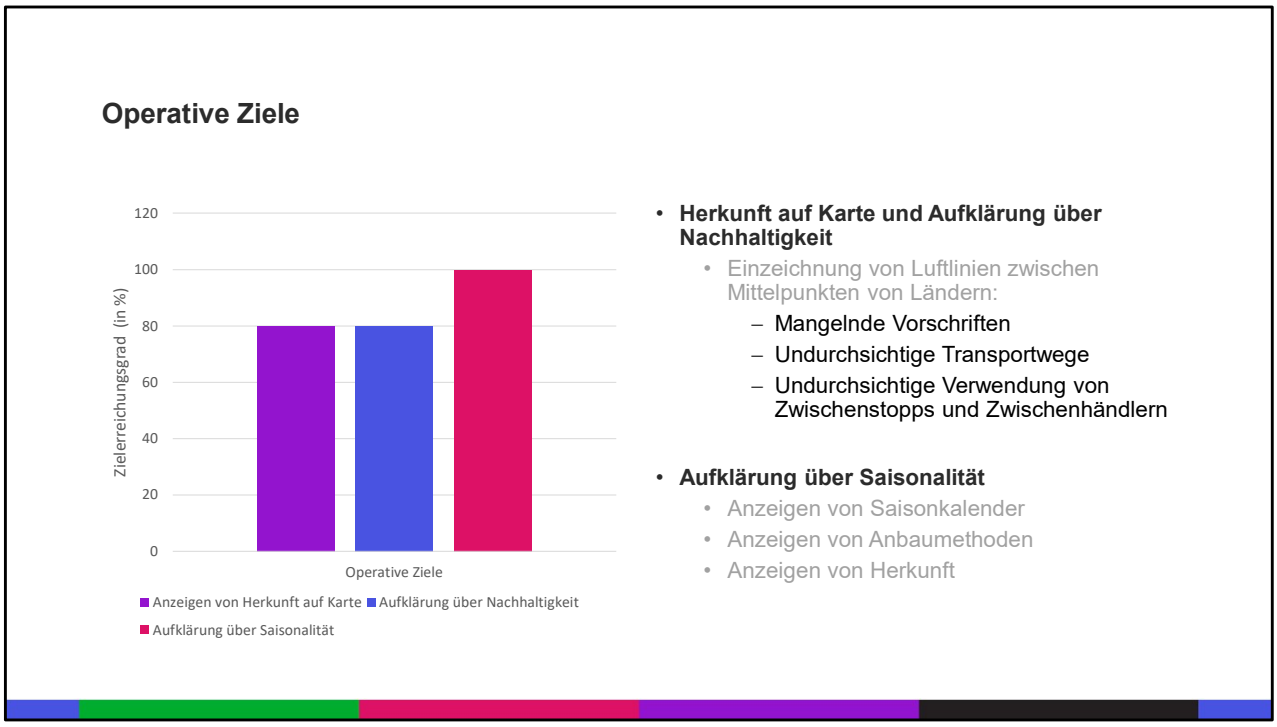
Bananen werden 3 Monate vor dem eigentlichen Verkauf geerntet. Während des zweiwöchigen Transportes im Schiff muss eine Temperatur von ca. 12°C herrschen. In Europa müssen die Bananen anschließend in einer Reifekammer zwischengelagert werden. In den Reifekammern herrscht eine kontrollierte Temperatur und das Gas Ethylen wird zur Unterstützung des Reifeprozess hinzugegeben. Anschließend erfolgt der Transport in den Einzelhandel.

Lexikon oder Blog, in dem die Wichtigsten Begriffe und Informationen im Bezug auf saisonales und regionales Einkaufen, sowie Nachhaltigkeit im Allgemeinen zum Nachlesen bereitgestellt wird.

JetSet Food ein Non-Profit-Projekt bleibt. Um die Kosten der App-Entwicklung und des Hostings zu decken, könnte das Modell „Gratis App mit In-App Advertisement“ genutzt werden. Zusätzlich könnte eine Werbefreie Premium-Version als In-App Kauf angeboten werden. Sollten die Kosten nicht durch Werbeeinnahmen gedeckt werden können, könnten einzelne Funktionen als kostenpflichtige Erweiterungen angeboten werden.

Kritische Bewertung der Zielerreichung

Audit 4



„Anzeigen von Herkunft auf Karte“: Durch Markierung des Landes und des Mittelpunktes erreicht.

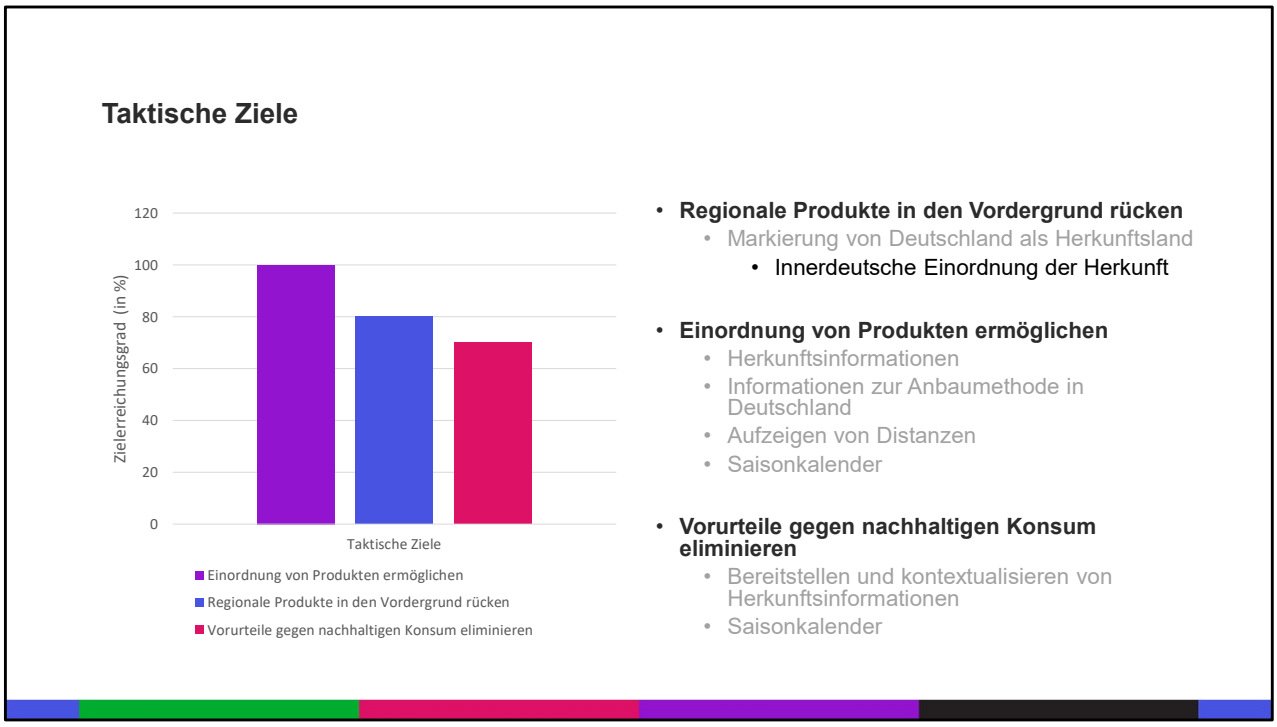
Keine genaueren Angaben, da in Deutschland, wenn überhaupt, nur die Angabe des Herkunftslandes verpflichtend ist. Händler kaufen das importierte Obst und Gemüse von Importeuren, also Zwischenhändlern, die die Früchte von verschiedenen Produzenten eines (oder mehreren) Ländern ankaufen, weshalb sich nicht gewährleisten lässt das eine bestimmte Obst oder Gemüse Sorte bei einem Händler immer vom selben Erzeuger stammt. Es existieren zwar Projekte, welche es dem Konsumenten ermöglichen bei bestimmten Händlern den genauen Anbauort von bestimmten Sorten herauszufinden, allerdings bis dies die Norm oder sogar Vorschrift ist, ist es nicht möglich genauere Angaben zu tätigen und einzuzeichnen.

„Aufklärung über Nachhaltigkeit“: Durch das Errechnen und Anzeigen der kürz möglichsten Strecken, welche ein Produkt zurücklegen muss, auf einer Karte sowie für Deutschland die farbliche Bewertung der vermutlichen Anbaumethode erreicht. Die Routen werden zwischen dem Mittelpunkt des Herkunftslandes und dem

Mittelpunkt von Deutschland errechnet. Diese Routen werden nicht feingranular errechnet und angezeigt, da auch diese stark variieren. Es ist zwar ersichtlich, welche Flug- und Schifffahrtshäfen für den generellen Warenimport/Export genutzt werden, aber nicht woher genau das Gemüse kommt, welches dort verkauft wird, wo der Nutzende einkauft. Außerdem werden auf der Transportroute Zwischenstopps, beispielsweise in Reifereien in welchen unreif geernteten Früchten nachgereift werden, eingelegt. Aus diesen Gründen würde bei dem aktuellen Informationsstand eine feinere Granularität der Transportrouten Fehlinformationen verbreiten.

Die farbliche Bewertung der Anbaumethoden setzt sich aus der Klimabelastung der Anbaumethoden zusammen.

„Aufklärung über Saisonalität“ wurde voll erreicht. Dem Nutzenden wird deutsches Obst und Gemüse klar hervorgehoben gezeigt, sowie zu jedem Produkt angezeigt ob und wann es aus deutschem Anbau verfügbar ist.



„Regionale Produkte in den Vordergrund rücken“: durch die deutliche Markierung von Deutschland als Herkunftsland zum Teil erreicht. Zum vollen Erreichen dieses Ziels wäre nun noch eine innerdeutsche Einordnung der Herkunft nötig.

„Einordnung von Produkten ermöglichen“: voll erreicht. Der Nutzende bekommt Herkunftsinformation sowie Informationen und eine Bewertung bezüglich der Anbaumethode für heimische Produkte und kann so informiert eine Kaufentscheidung treffen. Durch die Darstellung auf einer Karte sowie das Anzeigen der Entfernungen bietet dem Nutzenden eine Vergleichsmöglichkeit und erinnert ihn an Distanzen. Des Weiteren wird durch das Aufzeigen eines Saisonkalenders der Nutzende darauf hingewiesen, dass dieses Produkt zu anderen Zeiten auch heimisch verfügbar ist, auch falls es nicht in Deutschland angebaut wird, wird der Nutzende darauf hingewiesen.

„Vorurteile gegen nachhaltigen Konsum eliminieren“: schwer quantifizierbar. Durch das Bereitstellen und kontextualisieren der Herkunftsinformationen durch das Anzeigen auf einer Karte sowie das Aufzeigen eines Saisonkalenders wird der Nutzenden auf nachhaltigere Entscheidungen hingewiesen.

Der Erreichungsgrad der strategischen Ziele lässt sich momentan nur schwer Bewerten. Durch die hohen Erreichungsgrade der untergeordneten Ziele sowie der konkreten Ansatzpunkte für weitere Entwicklung lässt sich diese allerdings in Aussicht stellen.