

Partitio

Florian Buchholz
Jörg J. Buchholz

27. Oktober 2024

Teil I

Bedienungsanleitung

Kapitel 1

Einführung

Wörtlich aus dem Lateinischen übersetzt heißt Partitio: „Teilung“, „Aufteilung“ oder „Verteilung“; und genau darum geht es: Partitio ist ein Programm, das Menschen beim Aufteilen gemeinsamer Objekte helfen kann; beispielsweise bei einer Scheidung oder beim Erben.

1.1 Warum sollten wir Partitio benutzen?

Partitio beruht auf dem Prinzip des optimalen Kompromisses, das heißt, dass die Teilnehmerinnen¹ so lange miteinander verhandeln, bis jede Einzelne mit dem Ergebnis zufrieden ist. Da die Verhandlungen aber nicht direkt, sondern über ein unparteiisches Programm geführt werden, bestimmen nicht persönliche Gefühle, sondern die wirklichen Wünsche und Ziele der einzelnen Teilnehmerinnen den ausgehandelten Kompromiss.

1.1.1 Scheidung, Trennung

„In dem Sessel hast du doch nie gesessen!“

Wenn sich zwei Partnerinnen trennen, stehen zwischen beiden häufig so viele negative Gefühle, dass es ihnen nicht möglich erscheint, im direkten Gespräch miteinander den gemeinsamen Besitz gerecht aufzuteilen. Jede Partnerin hat Angst, von der anderen „über den Tisch gezogen“ zu werden und gerät dabei selbst leicht in die Versuchung, die andere „auszutricksen“.

1.1.2 Erben

„Eigentlich brauche ich ja gar nichts aus dem Erbe meiner Mutter, aber wenn ich jetzt sehe, wie sich meine Schwestern um jeden Eierbecher streiten, . . .“

Geschwister haben in ihrer Kindheit um alles konkurriert, was ihre Eltern ihnen an materiellen und ideellen Zuwendungen geben konnten. Häufig führt diese verinnerlichte Konkurrenzsituation dann auch noch im Erbfall dazu, dass die Einzelne das Gefühl hat, ihr „Anrecht“ gegen ihre Geschwister durchsetzen zu müssen, obwohl ihr die meisten zu erbenden Objekte eigentlich relativ egal sind.

¹Dieses Dokument verwendet das generische Femininum. Männer sind automatisch mitgemeint.

1.2 Wie löst Partitio das Problem?

Das Ziel ist klar:

**Jede Teilnehmerin bekommt am Ende genau das,
was ihr am wichtigsten sind.**

Dazu erhält jede Teilnehmerin insgesamt 1000 Taler (Punkte, virtuelles Spielgeld, ...) und setzt davon – wie bei einer Auktion – beliebige Summen auf die Objekte, die sie gerne bekommen möchte. Im theoretischen Idealfall gibt es am Ende keine Überschneidungen; jede Teilnehmerin bekommt das, was sie haben möchte und alle sind zufrieden. In der Praxis funktioniert das natürlich nicht; schließlich möchte eigentlich jeder gerne die Briefmarkensammlung von Opa haben, niemand interessiert sich für den alten Fernseher und „bevor mein Bruder den ganzen Schmuck bekommt, ...“.

Dann wird es spannend. Angenommen, Kainina hat 600 Taler für die Briefmarkensammlung geboten; dann hat sie natürlich nur noch 400 Taler für die restlichen Objekte zur Verfügung. Abella hat jetzt zwei Alternativen: Entweder sie überbietet Kainina (z.B. mit 700 Talern); dann hat Kainina wieder 1000 und Abella nur noch 300 Taler für die übrigen Objekte. Oder aber Abella sagt sich, dass ihr die Briefmarkensammlung nicht mehr als 600 Taler wert ist, überlässt Kainina die Briefmarken und kann nun ihrerseits 1000 Taler für andere Objekte setzen.

Interessant ist nun, dass es – anders als bei einer klassischen Auktion – bei Partitio auch jederzeit möglich ist, Gebote wieder zu erniedrigen, um auf diese Weise wieder mehr Taler auf vielleicht doch ganz interessante andere Objekte setzen zu können. Natürlich geht frau durch eine Erniedrigung immer das Risiko ein, dass eine andere Teilnehmerin das dann günstiger gewordene Angebot übernimmt; aber schließlich kann frau sich ja jederzeit entscheiden, doch wieder mehr zu setzen.

Durch die Möglichkeit, Gebote beliebig oft zu erhöhen oder zu erniedrigen, wechseln einzelne Objekte vielleicht mehrere dutzend mal die Besitzerin und ihr „Preis“ steigt und sinkt; am Ende des Projekts bekommt sie aber doch die Teilnehmerin, der sie insgesamt am meisten wert sind.

Das Projekt ist daher genau dann im Gleichgewichtszustand und damit zu Ende, wenn alle Teilnehmerinnen zufrieden sind. Oder anders gesehen: Solange noch eine Teilnehmerin unzufrieden mit dem Ergebnis ist, wird sie es einfach ihren Vorstellungen gemäß anpassen. Darauf reagiert dann möglicherweise eine andere Teilnehmerin wieder, so dass frau selber auch wieder ein paar Taler verschieben möchte, bis ... ja ... bis eben alle zufrieden sind.

Klingt kompliziert? Probier's einfach mal aus: partitio.de

Kapitel 2

Geht los ...!

2.1 Anmelden

Durch den Aufruf von `partitio.de` startet Erika automatisch ein neues Projekt und landet auf der Anmeldeseite (Abbildung 2.1)

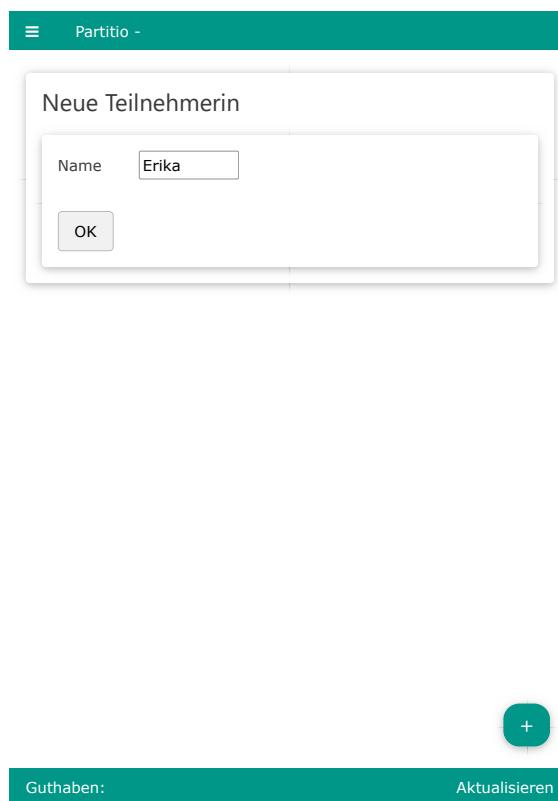


Abbildung 2.1: Anmeldeseite

auf der sie ihren Namen eintragen kann. Um den Platzbedarf in der Objektliste (Abschnitt 2.4) überschaubar zu halten, wird der Name auf 10 Zeichen begrenzt.

2.2 Lesezeichen erstellen

Nachdem Erika ihren Namen eingetragen hat, gelangt sie auf eine Informationsseite (Abbildung 2.2)

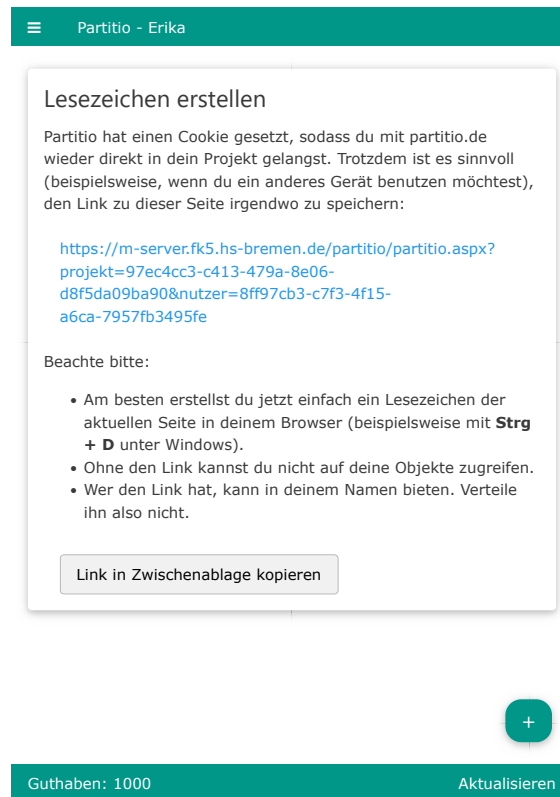


Abbildung 2.2: Lesezeichen erstellen

auf der ihr Name in der Titelzeile („Partitio - Erika“) eingetragen ist und auf der sie darüber informiert wird, dass Partitio einen Cookie im aktuellen Browser gesetzt hat, sodass sie beim nächsten Aufruf von `partitio.de` automatisch in ihr bestehendes Projekt gelangt. Um sicher zu gehen, kann Erika zusätzlich (beispielsweise unter Windows mit `Strg + D`) ein Lesezeichen in ihrem Browser setzen.

Wenn sie ihr Projekt auch in einem anderen Browser oder auf einem anderen Gerät öffnen möchte, kann sie den dazu notwendigen Link durch Anklicken des gleichnamigen Buttons in die Zwischenablage kopieren und irgendwie (E-Mail, Whatsapp, ...) auf das andere Gerät übertragen.

Der Link, der in diesem Fall

```
https://m-server.fk5.hs-bremen.de/partitio/partitio.aspx?
projekt=97ec4cc3-c413-479a-8e06-d8f5da09ba90&
nutzer=8ff97cb3-c7f3-4f15-a6ca-7957fb3495fe
```

lautet, beinhaltet sowohl die Adresse des Programmes

```
https://m-server.fk5.hs-bremen.de/partitio/partitio.aspx
```

als auch eine eindeutige aber nicht erratbare Kennung des aktuellen Projekts

```
projekt=97ec4cc3-c413-479a-8e06-d8f5da09ba90
```

und die ebenfalls nicht erratbare¹ Kennung der aktuellen Teilnehmerin:

`nutzer=8ff97cb3-c7f3-4f15-a6ca-7957fb3495fe`

2.3 Objekte fotografieren

Durch Anklicken des Plus-Buttons (rechts unten in Abbildung 2.2) gelangt Erika zu einer Abfrage, in der sie den Zugriff auf die Kamera abnickt².

Als dann zeigt die Seite das Bild der rückwärtigen Kamera des Smartphones oder der Webcam³ in einer Auflösung von 300×300 Pixeln (Abbildung 2.3).

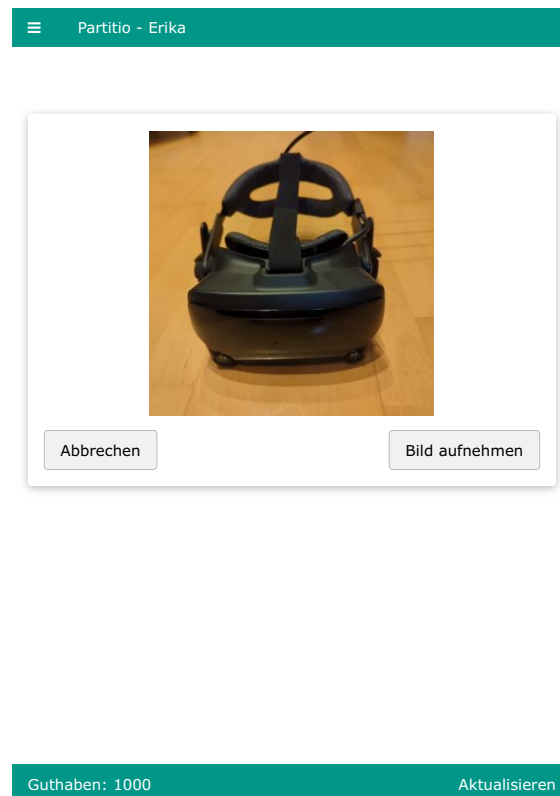


Abbildung 2.3: Bild eines neuen Objekts aufnehmen

Durch Anklicken des Bild aufnehmen-Buttons wird der aktuelle Kameraausschnitt auf den Server hochgeladen und sofort in der Objektliste angezeigt (Abbildung 2.4).

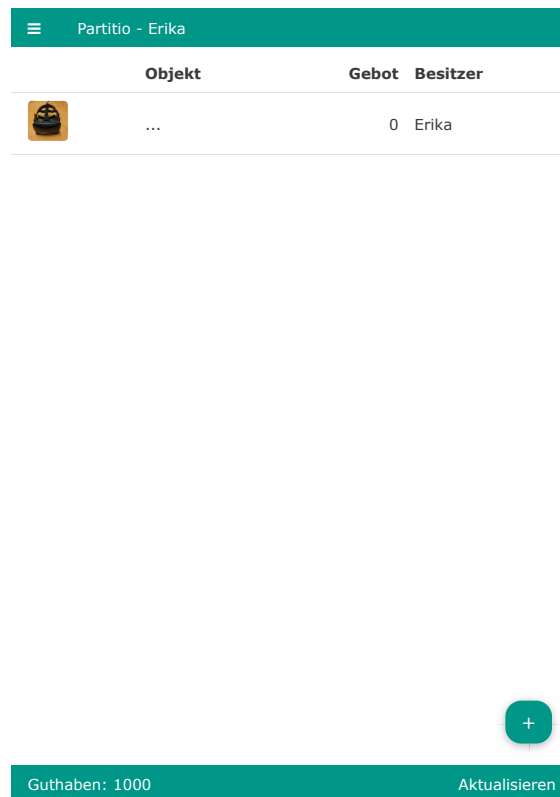
¹Die Kennungen sind sogenannte Globally Unique Identifier (GUID). Sie bestehen aus 32 zufälligen Zahlen und Buchstaben (a bis f) und lassen sich daher praktisch nicht erraten. Aus dem gleichen Grund geht auch die Wahrscheinlichkeit, dass nochmals die gleiche GUID erzeugt wird, gegen null.


²Wenn Erika den Zugriff auf ihre Kamera verbietet, kann sie keine eigenen Objekte fotografieren und hochladen. Sie kann aber trotzdem auf Objekte bieten, die andere Teilnehmerinnen hochgeladen haben.

³Üblicherweise läuft Partitio auf einem Smartphone (oder Tablet), da es damit sehr einfach ist, die zu teilenden Objekte direkt vor Ort zu fotografieren. Bei Bedarf kann Erika Objekte aber auch beispielsweise am PC oder Notebook mit einer Webcam fotografieren. Das Hochladen von bereits gespeicherten Bildern ist aus Sicherheitsgründen nicht möglich.

2.4 Objektliste

Die Objektliste (Abbildung 2.4) zeigt alle Objekte des Projekts, egal, von wem sie fotografiert wurden.



Objekt	Gebot	Besitzer
 ...	0	Erika

Guthaben: 1000 Aktualisieren

Abbildung 2.4: Objektliste

Jede Zeile der Liste beschreibt ein Objekt und besteht aus folgenden Spalten:

Bild In der Liste wird das Bild – das mit 300×300 Pixeln aufgenommen wurde – aus Gründen der Übersichtlichkeit mit 40×40 Pixeln dargestellt. Durch Zoomen oder Anklicken des Bildes (Abschnitt 2.5) wird aber die volle Auflösung sichtbar.

Objekt In Abbildung 2.4 werden drei Punkte als Objektname angezeigt, damit das neue, „unbekannte“ Objekt ganz oben in der nach dem Objektnamen sortierten Liste erscheint. Nach dem Eintragen eines Namens im Detailfenster (Abschnitt 2.5 und Abschnitt 2.10) wird dieser in der Liste angezeigt (Abbildung 2.5).

Gebot Für das neue Objekt wurde noch kein Gebot abgegeben. In der gleichnamigen Spalte wird daher in Abbildung 2.4 eine Null angezeigt. Erst wenn Erika im Detailfenster (Abschnitt 2.5) ihr Gebot für das Objekt eingetragen hat, wird auch dieses in der Liste angezeigt (Abbildung 2.5).

Besitzer Da Erika das Objekt aufgenommen hat und noch niemand ein Gebot abgegeben hat, wird sie in der entsprechenden Spalte als Besitzerin aufgelistet.

2.5 Detailfenster

Erika kann jetzt in der Objektliste (Abbildung 2.4) das Bild oder „...“ oder „0“ oder „Erika“ anklicken, um das Detailfenster über der Objektliste öffnen (Abbildung 2.5).

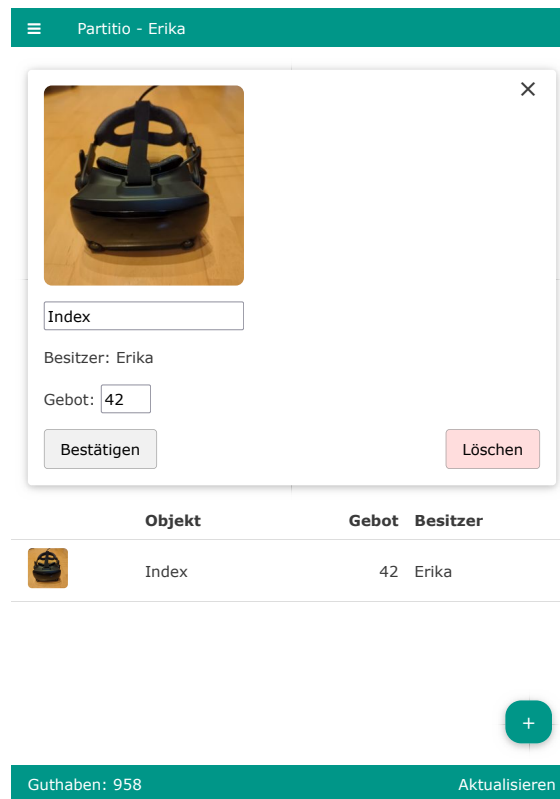


Abbildung 2.5: Detailfenster

Nachdem sie das Objekt mit einem Namen versehen („Index“), ihr Gebot („42“) für das Objekt abgegeben und den Bestätigen-Button angeklickt hat, erscheinen diese Daten in der Objektliste und das Detailfenster wird geschlossen⁴. Alternativ kann Erika das × rechts oben im Detailfenster anklicken, um es ohne Änderungen wieder zu schließen.

Erikas Guthaben (links unten in Abbildung 2.5) reduziert sich durch ihr Gebot von 42 jetzt natürlich auf $1000 - 42 = 958$.

2.6 Löschen

Jede Teilnehmerin kann Objekte – auch solche, die sie selbst nicht aufgenommen hat – löschen, indem sie im Detailfenster auf den Löschen-Button klickt. Es erscheint ein Löschenbestätigungsfenster (Abbildung 2.6) und durch Anklicken des dortigen Löschen-Button wird das Objekt aus der Liste gelöscht⁵.

⁴In Abbildung 2.5 haben wir das Detailfenster nochmals geöffnet, um die Informationen im Detailfenster und der Objektliste gemeinsam darstellen zu können.

⁵Wir klicken in Abbildung 2.6 auf den Abbrechen-Button, sodass das Objekt nicht gelöscht wird.

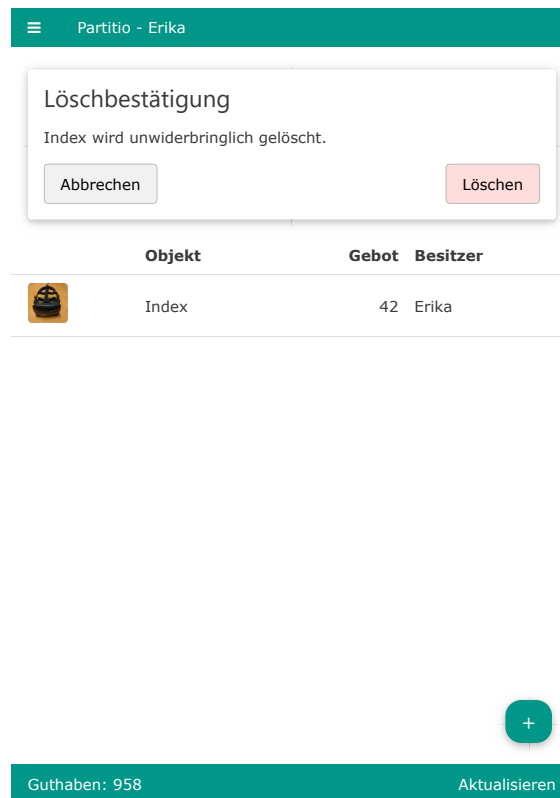


Abbildung 2.6: Löschbestätigung

2.7 Hamburger-Menü

Nach Anklicken des Hamburger-Buttons (\equiv , ganz links oben) öffnet sich das Hamburger-Menü (Abbildung 2.7).

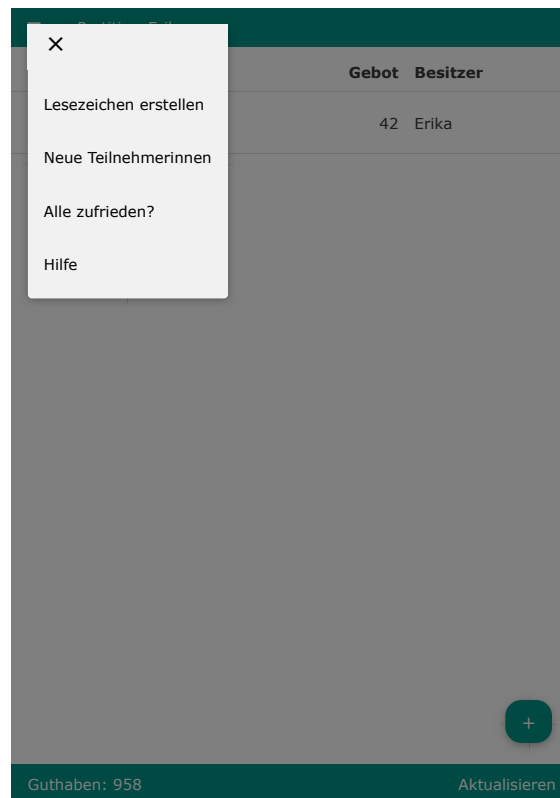


Abbildung 2.7: Hamburger Menü

Erika kann das Hamburger-Menü wieder schließen, indem sie auf das × im ersten Eintrag oder irgendwo außerhalb des Menüs klickt.

Ein Klick auf den zweiten Eintrag öffnet das schon in Abschnitt 2.2 beschriebene Lesezeichen erstellen-Fenster.

2.7.1 Neue Teilnehmerinnen

Ein Klick auf den dritten Eintrag des Hamburger-Menüs öffnet ein Fenster, in dem Erika weitere Teilnehmerinnen zu ihrem Projekt einladen kann (Abbildung 2.8).

Partitio - Erika

Neue Teilnehmerinnen


Um neue Teilnehmerinnen zu deinem Partitio-Projekt einzuladen, sendest du ihnen einfach den folgenden Link per Whatsapp, Signal, Telegram, SMS, ... oder E-Mail:

<https://m-server.fk5.hs-bremen.de/partitio/partitio.aspx?projekt=97ec4cc3-c413-479a-8e06-d8f5da09ba90&nutzer=42>

Beachte bitte:

- Wer den Link hat, kann an dem Projekt teilnehmen.
- Viele Menschen scheuen sich sinnvollerweise, auf Links in Nachrichten zu klicken. Eine E-Mail könnte im Spam-Ordner landen. Vielleicht möchtest du zusätzlich anrufen?

Link in Zwischenablage kopieren

Objekt	Gebot	Besitzer
 Index	42	Erika

Guthaben: 958 Aktualisieren

Abbildung 2.8: Neue Teilnehmerinnen

Um beispielsweise ihren Bruder Heinz als weiteren Teilnehmer einzuladen, schickt Erika ihm den im Fenster angegebenen Link. Sie kann dazu im Fenster auf „E-Mail“ klicken, woraufhin sich ihr E-Mail-Programm mit einer vorbereiteten E-Mail öffnet. Sie muss nur noch Heinz' E-Mail-Adresse eintragen und die E-Mail abschicken. Alternativ kann sie natürlich auch jeden anderen Kommunikationsweg (Whatsapp, SMS, Telefon, ...) nutzen, um Heinz den Link zukommen zu lassen.

Der Link unterscheidet sich von dem in Abschnitt 2.2 beschriebenen nur durch die angegebene Nutzer-Kennung:

Lesezeichen erstellen Mit diesem Link kann sich Erika wieder als Erika in ihrem Projekt anmelden. Die Nutzer-Kennung ist ihre eigene Kennung:

`nutzer=8ff97cb3-c7f3-4f15-a6ca-7957fb3495fe`

Neue Teilnehmerinnen Mit diesem Link lädt Erika neue Teilnehmer zu ihrem Projekt ein. Die Nutzer-Kennung ist eine Kennung, die im Projekt ganz sicher nicht vorkommt:

`nutzer=42`

Auf diese Weise wird Heinz, sobald er den Link aufruft, automatisch zur Anmelde-seite umgeleitet, wo er sich anmeldet und eine eigene Nutzer-Kennung erhält.

2.7.2 Anonymität

Das Hantieren mit Projekt- und Nutzerkennungen mag auf den ersten Blick umständlich erscheinen, garantiert aber jeder Teilnehmerin absolute Anonymität. Auf diese Weise wird auf dem Partitio-Server keine einzige E-Mail-Adresse oder eine sonstige Information gespeichert, mit der die Objekte eines Projekts einer real existierenden Teilnehmerin

zugeordnet werden könnten. Wer beispielsweise Omas Goldbarren fotografiert und das Foto auf einen Server hochlädt, möchte vermutlich nicht wirklich, dass die Goldbarren mit seiner „Adresse“ verknüpft werden, oder?

2.7.3 Alle zufrieden?

Ein Klick auf den vierten Eintrag des Hamburger-Menüs öffnet eine Seite (Abbildung 2.9)

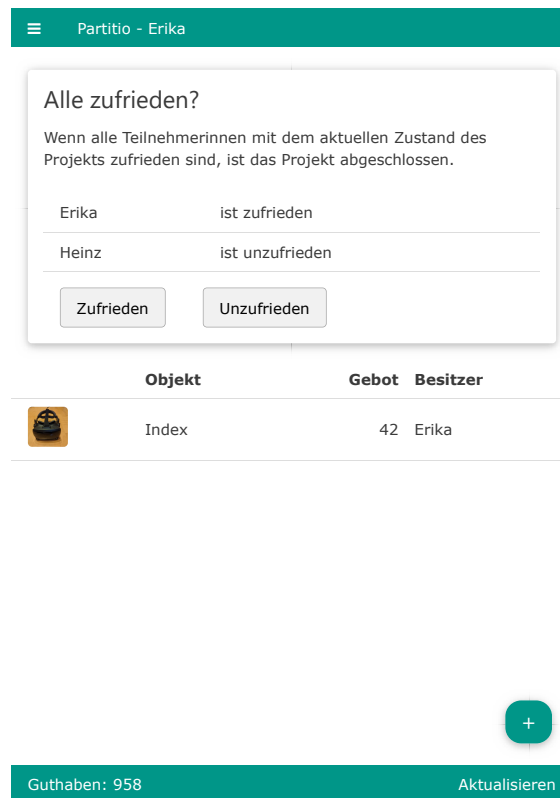


Abbildung 2.9: Alle zufrieden?

auf der eine Teilnehmerin durch Anklicken des Zufrieden-Buttons eintragen kann, dass sie mit dem aktuellen Zustand des Projekts zufrieden ist und dass ihrer Meinung nach das Projekt jetzt beendet werden könnte⁶.

Wenn alle Teilnehmerinnen zufrieden sind, kann das Projekt beendet werden. In der aktuellen Version wird das Projekt nicht automatisch beendet, wenn alle Teilnehmerinnen zufrieden sind. Es ist also sinnvoll, ab und zu mal auf Zufriedenheitsseite vorbeizuschauen, um sich über den aktuellen Zufriedenheitsstand des Projekts zu informieren. Natürlich kann die Übereinkunft zum Abschluss des Projekts und die dann folgende reale Aufteilung der Objekte auch über einen beliebigen anderen Kommunikationsweg erfolgen.

Wenn Heinz ein Objekt von Erika übernimmt, wird Erikas Zustand automatisch auf „unzufrieden“ gesetzt und das Projekt geht weiter.

⁶Wir haben zu Demonstrationszwecken für Abbildung 2.9 vorher schon bei Erika auf den Zufrieden-Button geklickt.

Drucken

Am Ende des Projekts, wenn alle zufrieden sind, kann jede Teilnehmerin aus dem Browser heraus die Webseite mit der Objektliste ausdrucken, um ihre „ersteigerten“ Objekte zu dokumentieren; auf Papier oder in ein PDF-Dokument.

In Firefox gibt es nach einem Rechtsklick außerdem den Eintrag „Bildschirmfoto aufnehmen“, um die gesamte Seite als Screenshot abzuspeichern.

2.7.4 Hilfe

Ein Klick auf den fünften Eintrag des Hamburger-Menüs öffnet dieses PDF-Dokument.

2.8 Sortieren

In Abbildung 2.10 haben Erika und Heinz schon fleißig Objekte aufgenommen und Gebote abgegeben.

Objekt	Gebot	Besitzer
Flipflops	1	Heinz
Helm	333	Heinz
Index	44	Heinz
Kopfhörer	74	Heinz
Ladestation	22	Heinz
Numpad	113	Heinz
Amperemeter	100	Erika
Digitalmultimeter	111	Erika
Gitarre	500	Erika
Iphone	97	Erika
Maus	66	Erika

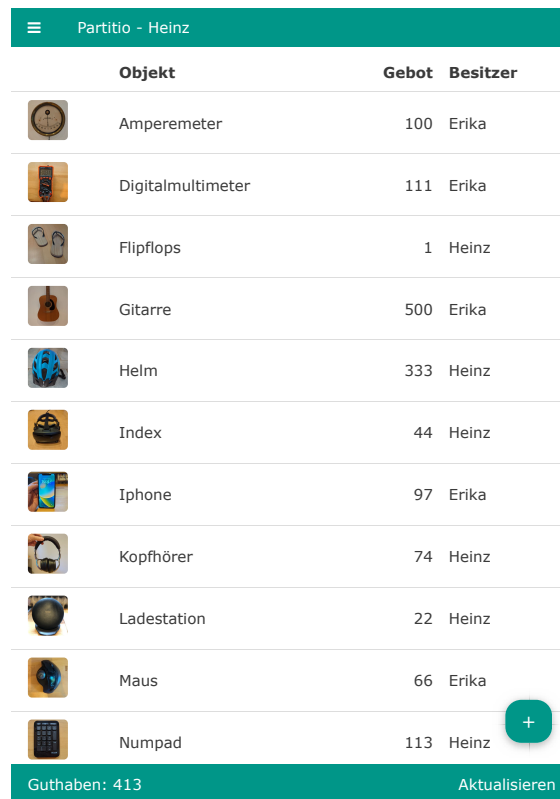
Guthaben: 413 Aktualisieren












Abbildung 2.10: Sortierreihenfolge: Besitzer, Objektname

Dabei ist anfänglich das erste Sortierkriterium in der Objektliste der Besitzer, wobei die Objekte des aktuellen Besitzers immer ganz oben angezeigt werden. Aus diesem Grund erscheinen seine Objekte in der Liste vor allen anderen, wenn Heinz das Projekt unter seinem Namen aufruft (Abbildung 2.10).

Das zweite Sortierkriterium innerhalb eines Besitzers ist der Objektname, sodass bei Heinz „Flipflops“ vor „Helm“ und bei Erika „Amperemeter“ vor „Digitalmultimeter“ gelistet sind.

In langen Listen kann es sinnvoll sein, alle Objekte – egal, wem sie gehören – alphabetisch zu sortieren. Dies geschieht durch Anklicken⁷ der Spaltenüberschrift „Objekt“ (Abbildung 2.11).



	Objekt	Gebot	Besitzer
	Amperemeter	100	Erika
	Digitalmultimeter	111	Erika
	Flipflops	1	Heinz
	Gitarre	500	Erika
	Helm	333	Heinz
	Index	44	Heinz
	Iphone	97	Erika
	Kopfhörer	74	Heinz
	Ladestation	22	Heinz
	Maus	66	Erika
	Numpad	113	Heinz

Guthaben: 413 Aktualisieren

Abbildung 2.11: Sortiert nach Objektname

Nach Anklicken der Spaltenüberschrift „Gebot“ wird die Objektliste aufsteigend nach der Höhe des Gebots sortiert (Abbildung 2.12).

⁷Ein erneutes Anklicken der Spaltenüberschrift „Objekt“ sortiert die Objekte in umgekehrter alphabetischer Reihenfolge. Das „Numpad“ würde dann als erstes Objekt gelistet.

Objekt	Gebot	Besitzer
Flipflops	1	Heinz
Ladestation	22	Heinz
Index	44	Heinz
Maus	66	Erika
Kopfhörer	74	Heinz
Iphone	97	Erika
Amperemeter	100	Erika
Digitalmultimeter	111	Erika
Numpad	113	Heinz
Helm	333	Heinz
Gitarre	500	Erika

Guthaben: 413 Aktualisieren

Abbildung 2.12: Sortiert nach Gebot

Auf diese Weise lassen sich in längeren Listen vielleicht günstige „Schnäppchen“ finden, die von anderen Teilnehmerinnen als nicht so „wertvoll“ angesehen werden.

Des Weiteren sortiert ein Klick auf die Spaltenüberschrift „Besitzer“ natürlich nach dem Namen des Besitzers⁸.

Gibt es irgendein sinnvolles Kriterium, um nach den Bildern in der ersten Spalte zu sortieren? Farbmittelwert? ...?

2.9 Über- und Unterbieten

In einer üblichen Auktion gibt es für Gebote nur eine Richtung: nach oben. In Partitio kann der Preis eines Objektes hingegen sowohl steigen als auch sinken.

Konkret kann Heinz beispielsweise das Iphone, das sich gemäß Abbildung 2.12 momentan mit 97 in Erikas Besitz befindet, von Erika übernehmen, indem er in der Detailansicht als Gebot 98 einträgt und den Bestätigen-Button anklickt oder einfach die Enter-Taste drückt.

Wenn er versucht, das Iphone mit 96 zu übernehmen, bekommt er die Information (98 ... 413 in Abbildung 2.13), dass er mindestens 98 bieten muss und maximal 413 als Guthaben zur Verfügung hat. Er kann dann entweder ein Gebot von mindestens 98 abgeben, das Detailfenster durch Anklicken von \times in der rechten oberen Ecke schließen oder einfach ein anderes Objekt auswählen.

⁸Um die anfängliche Sortierreihenfolge „Meine zuerst“ wieder herzustellen, ist in der aktuellen Partitio-Version ein erneutes Aufrufen des Projekts (partitio.de) notwendig.

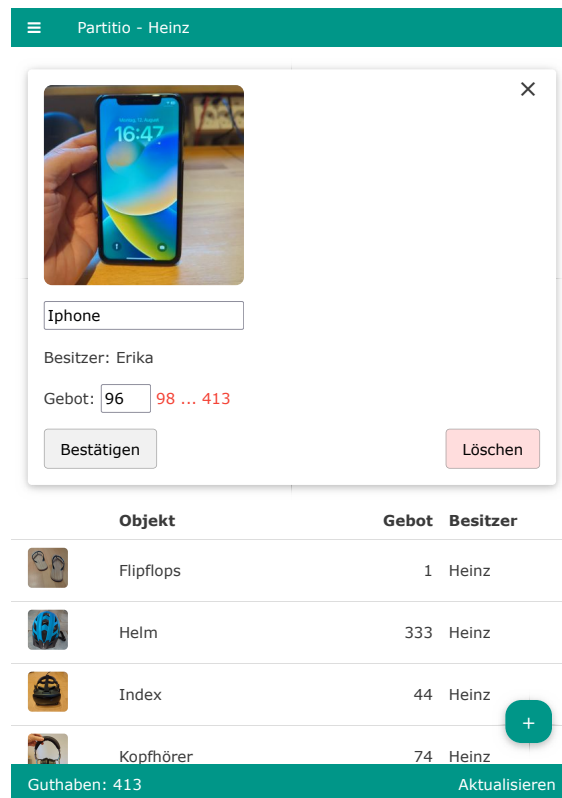


Abbildung 2.13: Übernehmen durch Unterbieten klappt nicht.

Wenn Heinz versucht, die Gitarre von Erika zu übernehmen, für die Erika ja momentan 500 geboten hat (Abbildung 2.12), wird er daran erinnert, dass sein aktuelles Guthaben (413) dazu nicht ausreicht (Abbildung 2.14).

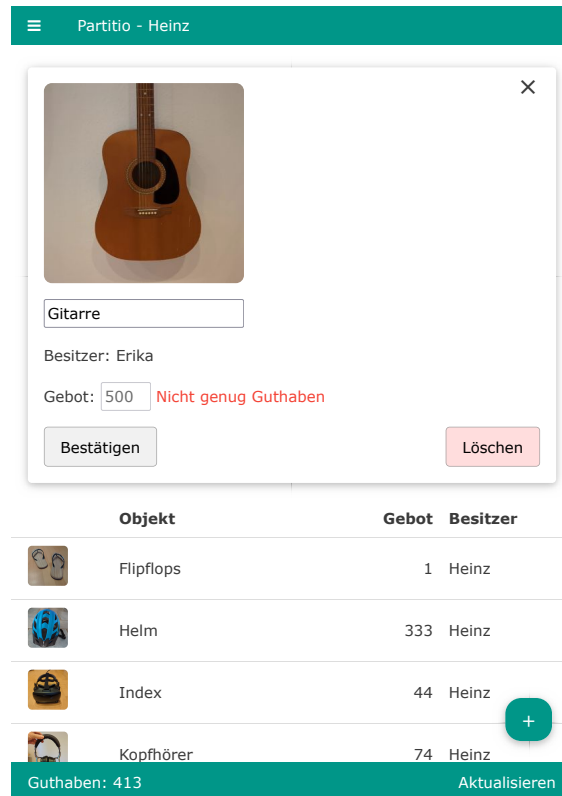


Abbildung 2.14: Nicht genug Guthaben

Wenn Heinz jetzt aber doch sehr an der Gitarre interessiert ist, kann er beispielsweise sein aktuelles Gebot für den Helm von 333 auf 200 **reduzieren**, um damit wieder 546 zur Verfügung zu haben und damit Erika die Gitarre für 501 abzunehmen (Abbildung 2.15).

Objekt	Gebot	Besitzer
Flipflops	1	Heinz
Gitarre	501	Heinz
Helm	200	Heinz
Index	44	Heinz
Kopfhörer	74	Heinz
Ladestation	22	Heinz
Numpad	113	Heinz
Amperemeter	100	Erika
Digitalmultimeter	111	Erika
Iphone	97	Erika
Maus	66	Erika

Guthaben: 45 Aktualisieren

Abbildung 2.15: Gitarre übernommen

Durch die Reduktion seines Gebotes auf den Helm von 333 auf 200 geht Heinz natürlich das Risiko ein, dass Erika im Gegenzug den Helm beispielsweise für 201 übernimmt; aber schließlich ist Heinz die Gitarre ja offensichtlich wichtiger als der Helm und er kann andererseits ja jederzeit sein Gebot für die Gitarre wieder erniedrigen und das für den Helm erhöhen.

Lange Rede, kurzer Sinn:

**Übernehmen geht nur durch Überbieten.
Gebote können aber jederzeit wieder erniedrigt werden.**

Ja, dadurch ist es möglich, ein Objekt durch Überbieten zu übernehmen und das Gebot gleich darauf wieder auf das alte Gebot (oder sogar noch weiter) zu erniedrigen und ja, das können natürlich auch andere Teilnehmerinnen machen. Auf diese Weise ist es möglich, dass ein Objekt pingpong-artig mehrfach den Besitzer wechselt, ohne dass sein Gebot faktisch steigt.

In der Spieltheorie gibt es für solche Situationen Regeln: So darf ein $K\bar{o}$ ⁹ beispielsweise erst beim **übernächsten** Zug wieder genommen werden. Wir haben in dieser Partitio-Version bewusst darauf verzichtet, komplizierte $K\bar{o}$ -Regeln einzuführen und vertrauen darauf, dass Teilnehmerinnen automatisch andere Objekte ins Visier nehmen, wenn sie merken, dass es bei einzelnen Objekten zu sinnlosen Pingpong-Kämpfen kommt.

⁹ $K\bar{o}$ (dt. etwa: „Ewigkeit“) bezeichnet eine Stellung beim Brettspiel Go, in der beide Spieler wechselseitig einzelne Steine des Gegners schlagen könnten, wodurch die jeweils vorherige Situation wiederhergestellt würde. [1]

2.10 KI

In einer der nächsten Partitio-Versionen ist geplant, den Objektnamen von einer lokalen KI direkt aus dem Bild vorschlagen zu lassen. Erste Versuche, die marktführende ChatGPT 4o [2] mit dem Prompt

Ich lade dir ein paar Bilder hoch. Beschreibe das Objekt auf dem jeweiligen Bild mit einem Wort.

auf die Bilder loszulassen, waren schon mal ziemlich erfolgversprechend:



Amperemeter



Headset



Gitarre



Nummernblock



Ladegerät



Flip-Flops



Kopfhörer



Multimeter



Helm



Smartphone



Trackball

Natürlich kann Chatty auf Nachfrage

Um welches Smartphone handelt es sich?

weitere Vermutungen anstellen

Das hochgeladene Smartphone scheint ein Apple iPhone zu sein, basierend auf dem abgebildeten Design. Aufgrund der Displayanzeige und des Kameraausschnitts könnte es sich um ein iPhone X¹⁰ oder ein neueres Modell handeln. Genauer könnte es ein iPhone 12 oder iPhone 13 sein, da diese Modelle das

¹⁰Es ist tatsächlich ein iPhone X.

gezeigte Design teilen. Um das Modell exakt zu bestimmen, wären genauere Informationen wie die Rückseite oder technische Details erforderlich.

aber wir hatten ja ganz ausdrücklich nur nach **einem** Wort als Beschreibung gefragt.

„Trackball“ und „Multimeter“ charakterisieren die Objekte vielleicht sogar besser als „Maus“ und „Digitalmultimeter“¹¹.

¹¹Ja, es gibt auch heute noch analoge Multimeter ...

Teil II

Unter der Haube

In früheren Projekten haben wir jede einzelne Zeile selbst programmiert und mit unseren eigenen erklärenden Kommentaren versehen. In Partitio haben wir uns erstmals massiv von künstlicher Intelligenz unterstützen lassen:

Während des Programmierens hat uns der Github Copilot [3] hilfreich über die Schulter geschaut und eigene Ideen und Verbesserungen des Codes angeboten, auf die wir häufig eingegangen sind. Manche Konzepte – wie das wechselweise auf- und absteigende Sortieren (`GridView_objekte_Sorting` und `GetSortDirection`) – haben wir komplett vom Copilot programmieren lassen. Dies ging signifikant schneller und fehlerfreier als wenn wir die üblichen Verdächtigen (Stack Overflow [4], ...) nach Lösungen durchsucht oder gar selbst Fragen¹² gestellt hätten. Auch beispielsweise das Aufnehmen und Hochladen des Bildes mit Hilfe von JavaScript (`bild_aufnehmen`) stammt praktisch komplett aus der Copilot-Feder. Hier waren allerdings einige emotionale Iterationen nötig, bevor das Bild wirklich von allen getesteten Geräten (Windows, Android, iOS) verzerrungsfrei mit 300×300 Pixeln zentriert dargestellt und aufgenommen wurde. Ebenso ist die KI nahezu vollständig für den in Kapitel 6 beschriebenen HTTP-Handler zum Entgegennehmen des Bildes auf Serverseite verantwortlich.

Nach Abschluss der Programmierung haben wir ChatGPT 4o [2] genutzt, um alle Methoden und jede einzelne Codezeile kommentieren zu lassen und ein paar einleitende Worte für jedes Kapitel zu schreiben. Das klappt erstaunlich gut. Dabei müssen wir – nicht ganz ohne Neid – eingestehen, dass die Kommentare manchmal treffsicherer und prägnanter formuliert sind als wir das ohne langes Nachdenken hinbekommen hätten. Wir sparen uns daher in den folgenden Kapiteln (Kapitel 3 bis Kapitel 6) jegliche eigene Erklärungen und zeigen ausschließlich die von der KI erzeugten Kommentare.

Außerdem haben wir probiert, Chatty eine grafische Darstellung (Abbildung 3.1) der Aufrufhierarchie der wichtigsten Methoden und ihre Kurzbeschreibungen¹³ (Abschnitt 3.1.1) zu entlocken. Während wir mit den Methodenbeschreibungen (Abschnitt 3.1.1) schon relativ zufrieden sind, besitzt die grafische Darstellung der Aufrufhierarchie (Abbildung 3.1) sicherlich noch recht viel Optimierungspotenzial.

¹²Die alten Hasen in Foren wie Stack Overflow neigen verständlicherweise dazu, Fragen nur dann zu beantworten, wenn diese einem gewissen Niveau entsprechen. Häufig kommen auf „dumme Fragen“ gar keine oder wenig hilfreiche Antworten der Art: „Lies mal erst mal das Handbuch!“ oder „Was hast du denn schon probiert?“. Künstliche Intelligenzen sind da wesentlich entgegenkommender und beantworten geduldig wirklich jede Frage – egal auf welchem Niveau. Und wenn von der KI gelieferter Code tatsächlich mal nicht auf Anhieb funktioniert, präsentieren wir ihr einfach die Fehlermeldung und sie startet einen weiteren Versuch. Spätestens nach drei, vier Iterationen läuft der Code dann in der Regel fehlerfrei und macht was er soll.

¹³Wir erkennen die Tatsache, dass die Methodenbeschreibungen von der KI erzeugt wurden, beispielsweise daran, dass sie maskuline Formen wie „Benutzer“ oder „Besitzer“ verwendet. Vermutlich hätten wir sie mit ein paar eindeutigen, zusätzlichen Prompts auch anweisen können, ausschließlich die feminine Form zu verwenden ...

Kapitel 3

partitio.aspx.vb

Die Datei `partitio.aspx.vb` ist der zugehörige Code-Behind für die Webseite `partitio.aspx`. Sie enthält die serverseitige Logik für die Interaktion mit dem Benutzer und den Daten der Anwendung. Hier sind die wichtigsten Funktionen:

Initialisierung und Session-Handling: Die `Page_Load`-Methode liest Cookies und Query-Strings, um `projekt_id` und `nutzer_id` zu bestimmen. Falls kein Projekt existiert, wird ein neues Projekt-XML erstellt, und der Nutzer wird auf die neue Projektseite umgeleitet.

Panel-Management: Die Methode `Alles_verstecken` kontrolliert die Sichtbarkeit aller Panels auf der Seite, um eine übersichtliche Oberfläche zu schaffen.

Datenbindung und Anzeige: `GridView_darstellen` füllt das GridView mit Daten aus der Projekt-XML und sorgt für die richtige Sortierung. Der Inhalt im GridView zeigt alle Objekte an, die zum Projekt gehören.

Event-Handling: Verschiedene `Button_Click`- und `LinkButton_Click`-Methoden behandeln Benutzeraktionen wie das Speichern von Änderungen, das Löschen eines Objekts oder die Anzeige des Nutzer-Guthabens. Bei Bedarf werden Panels oder weitere Elemente eingeblendet.

Zufriedenheitsstatus und Nutzerverwaltung: Funktionen wie `Button_zufrieden_Click` und `Button_unzufrieden_Click` ermöglichen es Nutzern, ihren Status zu ändern. Die Funktion `Nutzer_name` liest den Namen eines Nutzers basierend auf der `nutzer_id`.

Sortierung und Filterung: Die Methode `GetSortDirection` bestimmt die Sortierrichtung für das GridView, um eine benutzerfreundliche Anzeige zu ermöglichen.

Cookies und Sitzungsdaten: `Cookies_lesen` und `Cookies_schreiben` verwalten Nutzerdaten für wiederholte Besuche, sodass diese nicht jedes Mal neu eingegeben werden müssen.

Die Datei enthält die komplette serverseitige Logik für das Partitio-Projektmanagement, einschließlich Datenverwaltung, Benutzerstatus und Seitennavigation.

3.1 Aufrufhierarchie

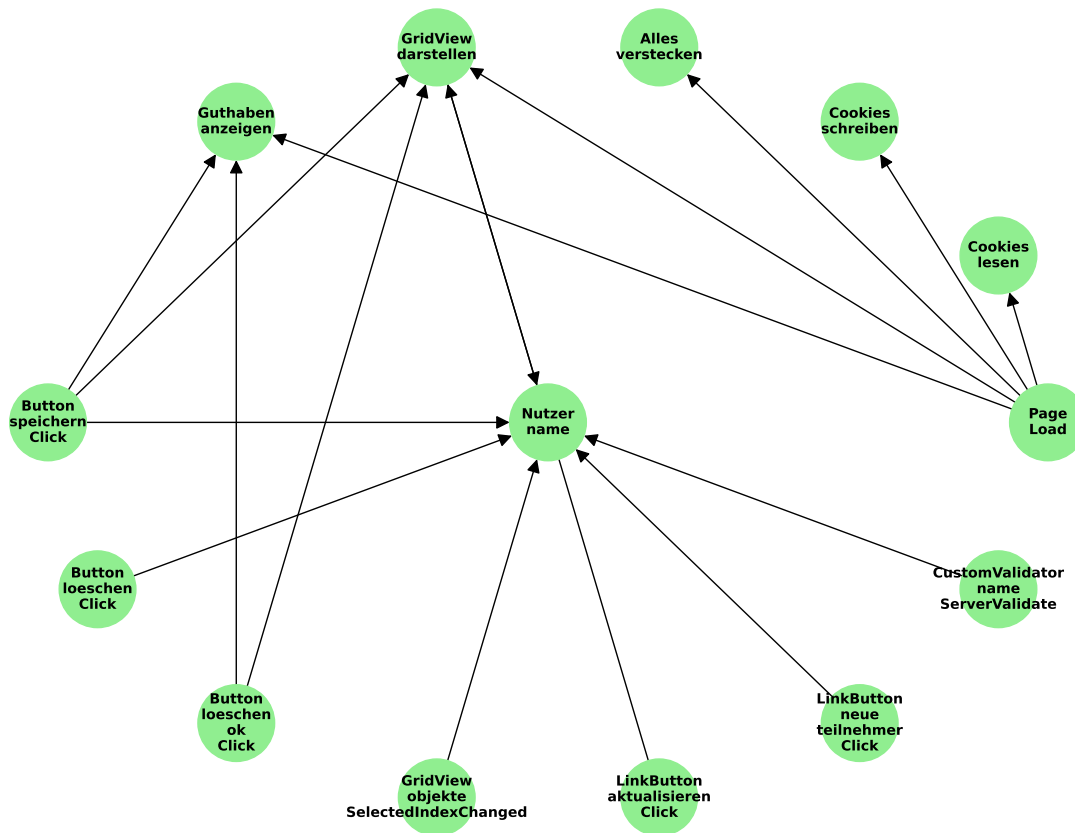


Abbildung 3.1: KI-generierte Aufrufhierarchie der wichtigsten Methoden

3.1.1 Methodenkurzbeschreibung

Hier ist eine kurze Beschreibung, wie die einzelnen Methoden in Partitio miteinander arbeiten:

Page_Load Wird beim Laden der Seite ausgeführt und initialisiert das Projekt. Sie ruft die Methoden `Cookies_lesen`, `GridView_darstellen`, `Alles_verstecken` und `Guthaben_anzeigen` auf, um Projekt- und Nutzerdaten zu laden und die Oberfläche anzupassen.

Cookies_lesen Liest die gespeicherten Projekt- und Nutzerdaten aus den Cookies und gibt sie an `Page_Load` zurück, um diese zu verwenden.

Cookies_schreiben Speichert die Projekt- und Nutzerdaten in Cookies und wird von `Page_Load` aufgerufen, um die Daten für spätere Besuche zu sichern.

Alles_verstecken Versteckt alle Panels auf der Benutzeroberfläche, um die Sichtbarkeit der Elemente basierend auf dem Benutzerstatus zu steuern. Sie wird von `Page_Load` aufgerufen, wenn die Oberfläche initialisiert wird.

GridView_darstellen Lädt die Objekte aus der XML-Datei und zeigt sie im GridView an. Sie wird von `Page_Load` und nach Änderungen an Objekten aufgerufen, um die Ansicht zu aktualisieren.

- GridView_objekte_SelectedIndexChanged** Wird aufgerufen, wenn ein Benutzer ein Objekt im GridView auswählt. Sie zeigt die Details des ausgewählten Objekts an und ruft `Nutzer_name` auf, um den Namen des Besitzers zu ermitteln.
- Button_speichern_Click** Speichert Änderungen am ausgewählten Objekt (z. B. Gebote oder Name) und aktualisiert die XML-Datei sowie das GridView durch einen erneuten Aufruf von `GridView_darstellen`.
- Guthaben_anzeigen** Berechnet und zeigt das verfügbare Guthaben des Nutzers basierend auf den Geboten. Diese Methode wird nach jeder Änderung an den Objekten (z. B. bei Speichern oder Löschen) aufgerufen.
- Button_loeschen_Click** Zeigt das Lösch-Panel für das ausgewählte Objekt an und bereitet die Löschung vor.
- Button_loeschen_ok_Click** Löscht das ausgewählte Objekt aus der XML-Datei und ruft anschließend `GridView_darstellen` und `Guthaben_anzeigen` auf, um die Änderungen anzuzeigen.
- Nutzer_name** Liefert den Namen eines Nutzers basierend auf seiner Nutzer-ID. Diese Methode wird von mehreren anderen Methoden (z. B. `GridView_darstellen`, `GridView_objekte_SelectedIndexChanged`) aufgerufen, um den Namen eines Nutzers anzuzeigen.
- Button_zufrieden_Click** **und** **Button_unzufrieden_Click** Setzen den Zufriedenheitsstatus des Nutzers und speichern diese Änderung in der XML-Datei.
- LinkButton lesezeichen_erstellen_Click** Erstellt einen Lesezeichen-Link für den aktuellen Projekt- und Nutzerzustand.
- CustomValidator_name_ServerValidate** Validiert den eingegebenen Nutzernamen und erstellt bei Bedarf einen neuen Nutzer in der XML-Datei.
- LinkButton_alle_zufrieden_Click** Diese Methode zeigt die Zufriedenheitsstatus aller Nutzer an, indem sie die Projekt-XML-Datei lädt, eine LINQ-Abfrage durchführt und die Daten in einem GridView anzeigt. Sie ruft keine weiteren Methoden direkt auf, interagiert jedoch mit der XML-Datei, ähnlich wie `GridView_darstellen`, um die Daten der Nutzer zu laden.
- LinkButton_aktualisieren_Click** Wird verwendet, um das GridView manuell zu aktualisieren, ohne die Seite neu zu laden. Sie ruft `GridView_darstellen` mit den im ViewState gespeicherten Sortierparametern (falls vorhanden) auf, um die aktuelle Sortierung beizubehalten.
- LinkButton_neue_teilnehmer_Click** Erstellt einen Einladungslink für neue Teilnehmer zum Projekt und zeigt diesen auf der Seite an. Es gibt keinen direkten Datenaustausch mit anderen Methoden, aber der generierte Link verwendet die `projekt_id`, die zuvor in `Page_Load` oder `Cookies_lesen` festgelegt wurde.
- LinkButton_hilfe_Click** Leitet den Nutzer auf ein externes Hilfedokument (PDF) weiter, das auf dem Server gespeichert ist. Diese Methode führt nur eine Umleitung aus und hat keinen Datenaustausch mit anderen Methoden.
- LinkButton_seitenleiste_oeffnen_Click** Macht das Panel für die Seitenleiste sichtbar, um zusätzliche Navigationsoptionen oder Funktionen anzuzeigen. Es gibt keine Interaktion mit anderen Methoden, die Funktion besteht nur darin, die Sichtbarkeit eines Panels zu ändern.

GridView_objekte_Sorting Wird aufgerufen, wenn der Benutzer auf eine Kopfzeile im GridView klickt, um die Objekte nach dieser Spalte zu sortieren. Sie ruft `GetSortDirection` auf, um die Sortierrichtung zu bestimmen, und dann `GridView_darstellen`, um das GridView mit den sortierten Daten neu zu füllen.

GetSortDirection Bestimmt die Sortierrichtung (aufsteigend oder absteigend) basierend auf der aktuellen Spalte, nach der sortiert wird. Sie wird von `GridView_objekte_Sorting` aufgerufen und speichert die aktuelle Sortierspalte und -richtung im ViewState, um bei einem erneuten Klick auf die gleiche Spalte die Sortierrichtung zu ändern.

Diese Methoden interagieren miteinander, indem sie Projekt- und Nutzerdaten aus den Cookies und der XML-Datei laden, diese anzeigen und bei Änderungen die Daten aktualisieren.

3.2 Page_Load

```

Protected Sub Page_Load(sender As Object, e As EventArgs) Handles
    Me.Load
    ' Diese Methode wird jedes Mal aufgerufen, wenn die Seite geladen wird,
    ' → unabhängig davon,
    ' ob es sich um einen erstmaligen Aufruf oder ein Postback (z.B. durch
    ' → einen Button-Klick) handelt.
    ' Sie initialisiert die Seite und sorgt dafür, dass die entsprechenden
    ' → Daten korrekt geladen und dargestellt werden.

    ' Cookies lesen, um zu sehen, ob projekt_id und nutzer_id in Cookies
    ' → gespeichert wurden
    ' und diese dann in die entsprechenden Variablen schreiben.
    Cookies_lesen()

    ' Überprüft, ob im Query-String eine "projekt"-ID übergeben wurde.
    ' Wenn das der Fall ist, wird die projekt_id aus dem Query-String
    ' → entnommen und der Variablen zugewiesen.
    If Request.QueryString("projekt") IsNot Nothing Then
        projekt_id = Request.QueryString("projekt")
    End If

    ' Überprüft, ob im Query-String eine "nutzer"-ID übergeben wurde.
    ' Wenn das der Fall ist, wird die nutzer_id aus dem Query-String
    ' → entnommen und der Variablen zugewiesen.
    If Request.QueryString("nutzer") IsNot Nothing Then
        nutzer_id = Request.QueryString("nutzer")
    End If

    ' Definiert ein JavaScript-Kommando, das beim Klicken auf ein
    ' → Bildauswahl-Element aufgerufen wird.
    ' Es übergibt die projekt_id und nutzer_id als Parameter an die
    ' → JavaScript-Funktion "bild_aufnehmen",

```

```

' um sie in der Funktion weiterzuverarbeiten und an
↳ "bild_empfangen.ashx" zu senden.
bild_aufnahme.OnClientClick = "bild_aufnehmen('" & projekt_id & "', '"
↳ & nutzer_id & "')"; return false;"

' Bestimmt den Dateipfad der XML-Datei, in der die Projektdaten
↳ gespeichert sind.
' Die Datei wird in einem Ordner namens "projekte" gespeichert und
↳ verwendet die projekt_id als Dateinamen.
projekt_xml_datei = MapPath("projekte/" & projekt_id & ".xml")

' Überprüft, ob die Projekt-XML-Datei existiert. Falls nicht, wird eine
↳ neue Projekt-ID generiert,
' eine neue XML-Datei erstellt und der Nutzer auf eine URL mit der
↳ neuen projekt_id umgeleitet.
If Not IO.File.Exists(projekt_xml_datei) Then
    projekt_id = System.Guid.NewGuid.ToString()
    projekt_xml_datei = MapPath("projekte/" & projekt_id & ".xml")

    ' Erstellt eine neue XML-Struktur für das Projekt, die leere
    ↳ Platzhalter für Nutzer und Objekte enthält.
    Dim neue_xml As XElement =
        <alles>
            <nutzers></nutzers>
            <objekte></objekte>
        </alles>

    ' Speichert die neue XML-Datei im entsprechenden Verzeichnis.
    neue_xml.Save(projekt_xml_datei)

    ' Leitet den Benutzer auf die gleiche Seite mit der neuen
    ↳ projekt_id weiter.
    Dim link = "partitio.aspx?projekt=" & projekt_id
    Response.Redirect(link)
End If

' Lädt die Projekt-XML-Datei, um die darin gespeicherten Daten
↳ weiterzuverarbeiten.
Dim projekt_xml = XElement.Load(projekt_xml_datei)

' Führt eine LINQ-Abfrage aus, um den aktuellen Nutzer anhand der
↳ nutzer_id in der XML-Datei zu suchen.
Dim aktueller_nutzer = From nutzer In projekt_xml.<nutzers>.<nutzer>
↳ Where nutzer.<nutzer_id>.Value = nutzer_id

' Überprüft, ob der Nutzer in der XML-Datei existiert. Wenn nicht, wird
↳ ein Panel eingeblendet,

```

```
' das dem Nutzer erlaubt, sich als neuer Teilnehmer anzumelden.
If aktueller_nutzer.Count = 0 Then
    Panel_neue_teilnehmerin.Visible = True
Else
    ' Wenn der Nutzer existiert, wird sein Name in einem Label
    ↪ angezeigt und die Seiten-Titelzeile entsprechend angepasst.
    Label_nutzer_name.Text = Nutzer_name(nutzer_id)
    Me.Title = "Partitio - " & Label_nutzer_name.Text

    ' Versteckt alle Panels, um die Benutzeroberfläche vorzubereiten.
    Alles_verstecken()

    ' Überprüft, ob eine Sitzungseinstellung gesetzt ist, die angibt,
    ↪ dass ein neues Lesezeichen erstellt werden soll.
    ' Falls ja, wird ein Klick-Event auf den Button zur
    ↪ Lesezeichen-Erstellung simuliert.
    If Session("Lesezeichen_erstellen_anzeigen") = "true" Then
        LinkButton lesezeichen_erstellen_Click(sender, e)
        Session("Lesezeichen_erstellen_anzeigen") = "false"
    End If

    ' Zeigt das aktuelle Guthaben des Nutzers auf der Seite an.
    Guthaben_anzeigen()

    ' Schreibt die aktuellen Projekt- und Nutzerdaten in Cookies, damit
    ↪ sie für spätere Besuche gespeichert bleiben.
    Cookies_schreiben()

    ' Wenn keine nutzer_id im Query-String vorhanden ist, leitet die
    ↪ Methode den Nutzer mit der aktuellen Projekt- und Nutzer-ID
    ↪ weiter,
    ' damit er die URL als Lesezeichen speichern kann.
    If Request.QueryString("nutzer") Is Nothing Then
        Response.Redirect("partitio.aspx?projekt=" & projekt_id &
            ↪ "&nutzer=" & nutzer_id)
    End If

    ' Überprüft, ob die Seite nicht durch ein Postback (z.B. durch das
    ↪ Klicken eines Buttons) geladen wurde.
    ' Wenn es kein Postback ist (also der erste Seitenaufruf), wird die
    ↪ GridView mit den Objekten gefüllt.
    If Not Page.IsPostBack Then
        GridView_darstellen(Nothing, Nothing)
    End If
End If
End Sub
```

3.3 Cookies_lesen

```
Protected Sub Cookies_lesen()  
    ' Diese Methode liest die in vorherigen Besuchen gespeicherten Cookies  
    ' → für "projekt_id"  
    ' und "nutzer_id" aus. Wenn die Cookies vorhanden sind, werden die  
    ' → Werte in die entsprechenden  
    ' Variablen "projekt_id" und "nutzer_id" geladen. Dies ermöglicht es,  
    ' → bei einem erneuten Besuch  
    ' die Daten des Nutzers und des Projekts automatisch zu laden.  
  
    ' Überprüft, ob das Cookie "partitio_projekt" existiert. Wenn ja, wird  
    ' → der gespeicherte Wert  
    ' des Cookies in die Variable "projekt_id" geladen.  
    If Request.Cookies("partitio_projekt") IsNot Nothing Then  
        projekt_id = Request.Cookies("partitio_projekt").Value  
    End If  
  
    ' Überprüft, ob das Cookie "partitio_nutzer" existiert. Wenn ja, wird  
    ' → der gespeicherte Wert  
    ' des Cookies in die Variable "nutzer_id" geladen.  
    If Request.Cookies("partitio_nutzer") IsNot Nothing Then  
        nutzer_id = Request.Cookies("partitio_nutzer").Value  
    End If  
  
    ' Wenn eines der Cookies nicht existiert, bleiben die entsprechenden  
    ' → Variablen unverändert,  
    ' und der Nutzer muss diese Informationen möglicherweise manuell  
    ' → eingeben oder sie werden  
    ' auf andere Weise abgerufen.  
End Sub
```

3.4 Cookies_schreiben

```
Protected Sub Cookies_schreiben()  
    ' Diese Methode speichert die Werte für "projekt_id" und "nutzer_id"  
    ' → als Cookies,  
    ' damit sie für zukünftige Besuche der Seite wiederverwendet werden  
    ' → können.  
    ' So kann der Nutzer ohne erneutes Eingeben seiner Daten direkt mit dem  
    ' → Projekt  
    ' weiterarbeiten.  
  
    ' Ein neues Cookie-Objekt für das Projekt erstellen. Der Cookie-Name  
    ' → ist "partitio_projekt".
```

```

' Der Wert des Cookies wird aus der Variable "projekt_id" genommen.
' Das Ablaufdatum wird auf ein Jahr ab dem aktuellen Zeitpunkt
  ↳ gesetzt.
Dim cookie_projekt As New HttpCookie("partitio_projekt", projekt_id)
  ↳ With {
      .Expires = DateTime.Now.AddYears(1)
  }

' Ein weiteres Cookie-Objekt für die Nutzer-ID erstellen. Der
  ↳ Cookie-Name ist "partitio_nutzer".
' Der Wert dieses Cookies ist die "nutzer_id", und das Cookie wird
  ↳ ebenfalls auf ein Jahr
  ↳ gültig sein.
Dim cookie_nutzer As New HttpCookie("partitio_nutzer", nutzer_id) With {
  .Expires = DateTime.Now.AddYears(1)
}

' Das Projekt-Cookie wird zur Sammlung der Cookies hinzugefügt, die mit
  ↳ der Antwort an den Client
' gesendet werden. Der Browser des Nutzers speichert dieses Cookie, das
  ↳ bei zukünftigen Anfragen
  ↳ an den Server zurückgeschickt wird.
Response.Cookies.Add(cookie_projekt)

' Dasselbe geschieht mit dem Nutzer-Cookie. Es wird dem Response-Objekt
  ↳ hinzugefügt,
' sodass der Browser des Benutzers dieses Cookie ebenfalls speichert
  ↳ und bei zukünftigen
  ↳ Besuchen der Seite an den Server zurücksendet.
Response.Cookies.Add(cookie_nutzer)
End Sub

```

3.5 Alles_verstecken

```

Protected Sub Alles_verstecken()
' Diese Methode versteckt alle relevanten Panels auf der Seite, um die
  ↳ Benutzeroberfläche
' für den aktuellen Nutzer anzupassen. So wird vermieden, dass unnötige
  ↳ Elemente angezeigt werden.

' Versteckt das Panel für die Seitenleiste, das möglicherweise
  ↳ Navigationselemente oder Optionen enthält.
Panel_seitenleiste.Visible = False

' Versteckt das Detail-Panel, das Informationen zu einem ausgewählten
  ↳ Objekt oder Teilnehmer anzeigt.

```

```
Panel_detail.Visible = False

' Versteckt das Lösch-Panel, das normalerweise verwendet wird, um die
↳ Löschung eines Objekts zu bestätigen.
Panel_loeschen.Visible = False

' Versteckt das Panel für die Erstellung eines Lesezeichens, das einen
↳ direkten Link zum aktuellen Zustand der Seite enthält.
Panel lesezeichen_erstellen.Visible = False

' Versteckt das Panel, das zur Aufnahme neuer Teilnehmerinnen in das
↳ Projekt verwendet wird.
Panel_neue_teilnehmerinnen.Visible = False

' Versteckt das Zufriedenheits-Panel, welches den Status der
↳ Zufriedenheit der Teilnehmer darstellt.
Panel_zufriedenheit.Visible = False

' Versteckt das Panel für neue Teilnehmerinnen, welches angezeigt wird,
↳ wenn ein Nutzer noch nicht existiert.
Panel_neue_teilnehmerin.Visible = False
End Sub
```

3.6 GridView_darstellen

```
Protected Sub GridView_darstellen(sortExpression As String,
↳ sortDirection As String)
' Diese Methode zeigt alle Objekte aus der Projekt-XML-Datei im
↳ GridView an.
' Die Objekte können nach einer bestimmten Sortierreihenfolge angezeigt
↳ werden,
' die durch die Parameter "sortExpression" und "sortDirection" bestimmt
↳ wird.

' Lädt die Projekt-XML-Datei, um die darin gespeicherten Objekte zu
↳ verarbeiten.
Dim projekt_xml = XElement.Load(projekt_xml_datei)

' Führt eine LINQ-Abfrage aus, um alle Objekte aus der XML-Datei zu
↳ extrahieren.
' Sortiert die Objekte zunächst danach, ob der aktuelle Nutzer der
↳ Besitzer ist (dann zuerst),
' dann nach dem Namen des Nutzers und schließlich nach dem Namen des
↳ Objekts.
Dim alle_objekte =
```



```
From objekt In projekt_xml.<objekte>.<objekt>
Order By If(objekt.<besitzer_id>.Value = nutzer_id, 0, 1),
        Nutzer_name(objekt.<besitzer_id>.Value),
        objekt.<objekt_name>.Value

' Erstellt eine neue DataTable, um die Objektdaten im GridView
↪ anzuzeigen.
Dim DataTable_objekte As New DataTable

' Fügt der DataTable eine Spalte für die Objekt-ID hinzu.
DataTable_objekte.Columns.Add("ID")

' Fügt der DataTable eine Spalte für die Bild-URL des Objekts hinzu.
DataTable_objekte.Columns.Add("Bild")

' Fügt der DataTable eine Spalte für den Objektnamen hinzu.
DataTable_objekte.Columns.Add("Objekt")

' Fügt der DataTable eine Spalte für das Gebot hinzu. Die Daten werden
↪ als Integer gespeichert,
' damit eine ordnungsgemäße Sortierung der Gebote möglich ist.
DataTable_objekte.Columns.Add("Gebot", GetType(Integer))

' Fügt der DataTable eine Spalte für den Namen des Besitzers des
↪ Objekts hinzu.
DataTable_objekte.Columns.Add("Besitzer")

' Iteriert über alle Objekte, die durch die LINQ-Abfrage zurückgegeben
↪ wurden,
' und fügt sie der DataTable hinzu.
For Each objekt In alle_objekte

    ' Extrahiert die Objekt-ID aus der XML-Datei.
    Dim objekt_id = objekt.<objekt_id>.Value

    ' Erstellt die Bild-URL für das Objekt basierend auf der Objekt-ID.
    Dim bild_url = "bilder/" & objekt_id & ".jpg"

    ' Extrahiert den Namen des Objekts aus der XML-Datei.
    Dim objekt_name = objekt.<objekt_name>.Value

    ' Extrahiert das Gebot für das Objekt aus der XML-Datei.
    Dim objekt_gebot = objekt.<objekt_gebot>.Value

    ' Extrahiert die Besitzer-ID des Objekts und konvertiert sie in den
    ↪ Namen des Besitzers.
    Dim objekt_besitzer_id = objekt.<besitzer_id>.Value
```

```
' Fügt die extrahierten Daten (ID, Bild-URL, Objektname, Gebot,
↪ Besitzername)
' als neue Zeile in die DataTable ein.
DataTable_objekte.Rows.Add(
    objekt_id,
    bild_url,
    objekt_name,
    objekt_gebot,
    Nutzer_name(objekt_besitzer_id))

Next

' Überprüft, ob eine Sortierspalte und -richtung angegeben wurde.
' Wenn ja, wird die DataTable nach der angegebenen Sortierung
↪ geordnet.
If Not String.IsNullOrEmpty(sortExpression) Then
    DataTable_objekte.DefaultView.Sort = sortExpression & " " &
    ↪ sortDirection
End If

' Weist das GridView die sortierte DataTable als Datenquelle zu.
GridView_objekte.DataSource = DataTable_objekte

' Bindet die Daten an das GridView, sodass die Daten in der
↪ Benutzeroberfläche angezeigt werden.
GridView_objekte.DataBind()

End Sub
```

3.7 GridView_objekte_SelectedIndexChanged

```
Protected Sub GridView_objekte_SelectedIndexChanged(sender As Object, e
↪ As EventArgs) Handles GridView_objekte.SelectedIndexChanged
' Diese Methode wird aufgerufen, wenn ein Nutzer im GridView ein Objekt
↪ auswählt.
' Sie lädt die Details des ausgewählten Objekts aus der
↪ Projekt-XML-Datei und zeigt sie
' in den entsprechenden Feldern auf der Seite an.

' Lädt die Projekt-XML-Datei, um die Informationen des ausgewählten
↪ Objekts zu verarbeiten.
Dim projekt_xml = XElement.Load(projekt_xml_datei)

' Extrahiert die ID des ausgewählten Objekts aus den Daten des
↪ GridView.
```

```
' Die ID wird verwendet, um das Objekt in der XML-Datei zu
↪ identifizieren.
Dim objekt_id As String = GridView_objekte.DataKeys(GridView_objekte.Se_
↪ lectedIndex).Value.ToString()

' Setzt die URL des Bildes für das ausgewählte Objekt, das in einem
↪ Image-Element angezeigt wird.
' Die Bild-URL basiert auf der Objekt-ID.
Image_objekt.ImageUrl = "bilder/" & objekt_id & ".jpg"

' Führt eine LINQ-Abfrage aus, um das ausgewählte Objekt in der
↪ XML-Datei anhand der Objekt-ID zu finden.
Dim objekt =
    From obj In projekt_xml.<objekte>.<objekt>
    Where obj.<objekt_id>.Value = objekt_id

' Setzt den Namen des ausgewählten Objekts in das entsprechende
↪ Textfeld.
Textbox_objekt_name.Text = objekt.<objekt_name>.Value

' Setzt das aktuelle Gebot des Objekts in das Textfeld für das Gebot.
' Das Textfeld wird für den Nutzer aktiviert, sodass er das Gebot
↪ ändern kann.
Textbox_objekt_gebot.Text = objekt.<objekt_gebot>.Value
Textbox_objekt_gebot.Enabled = True

' Setzt den Namen des Besitzers des Objekts in ein Label auf der
↪ Seite.
Label_besitzer.Text = Nutzer_name(objekt.<besitzer_id>.Value)

' Überprüft, ob der aktuelle Nutzer der Besitzer des ausgewählten
↪ Objekts ist.
' Wenn ja, wird der Bereich für die Gebotsänderung entsprechend
↪ angepasst.
If objekt.<besitzer_id>.Value = nutzer_id Then

    ' Der Nutzer ist der Besitzer, daher wird der Minimalwert für das
    ↪ Gebot auf 0 gesetzt,
    ' und der Maximalwert auf die Summe aus dem aktuellen Guthaben des
    ↪ Nutzers und dem aktuellen Gebot.
    RangeValidator_gebot.MinimumValue = 0
    RangeValidator_gebot.MaximumValue = CInt(Label_guthaben.Text) +
    ↪ CInt(Textbox_objekt_gebot.Text)

    ' Setzt die Fehlermeldung für den RangeValidator auf den erlaubten
    ↪ Bereich für das Gebot.
    RangeValidator_gebot.ErrorMessage =
    ↪ RangeValidator_gebot.MinimumValue & " ... " &
    ↪ RangeValidator_gebot.MaximumValue
```

```
Else
    ' Wenn der aktuelle Nutzer nicht der Besitzer des Objekts ist, wird
    ↪ der Minimalwert für das Gebot
    ' auf das aktuelle Gebot + 1 gesetzt, da ein höheres Gebot
    ↪ notwendig ist.

    ' Überprüft, ob das Guthaben des Nutzers ausreicht, um ein Gebot
    ↪ abzugeben, das höher als das aktuelle ist.
    If CInt(Label_guthaben.Text) >= CInt(Textbox_objekt_gebot.Text) + 1
    ↪ Then
        ' Setzt den Minimalwert für das Gebot auf das aktuelle Gebot +
        ↪ 1.
        RangeValidator_gebot.MinimumValue =
        ↪ CInt(Textbox_objekt_gebot.Text) + 1

        ' Setzt den Maximalwert für das Gebot auf das aktuelle Guthaben
        ↪ des Nutzers.
        RangeValidator_gebot.MaximumValue = CInt(Label_guthaben.Text)

        ' Setzt die Fehlermeldung für den RangeValidator auf den
        ↪ erlaubten Bereich für das Gebot.
        RangeValidator_gebot.ErrorMessage =
        ↪ RangeValidator_gebot.MinimumValue & " ... " &
        ↪ RangeValidator_gebot.MaximumValue

    Else
        ' Wenn das Guthaben des Nutzers nicht ausreicht, wird der
        ↪ Minimal- und Maximalwert des Validators
        ' auf einen Wert gesetzt, der ungültig ist (1001). Dadurch kann
        ↪ kein Gebot abgegeben werden.

        RangeValidator_gebot.MinimumValue = 1001
        RangeValidator_gebot.MaximumValue = 1001

        ' Setzt die Fehlermeldung auf "Nicht genug Guthaben", um dem
        ↪ Nutzer zu signalisieren,
        ' dass er kein Gebot abgeben kann.
        RangeValidator_gebot.ErrorMessage = "Nicht genug Guthaben"

        ' Markiert das Gebot als ungültig und deaktiviert das Textfeld
        ↪ für die Gebotseingabe.
        RangeValidator_gebot.IsValid = False
        Textbox_objekt_gebot.Enabled = False
    End If
End If

End If
```

```

' Setzt die Fehlermeldung des RequiredFieldValidators auf die des
  ↳ RangeValidators,
' um sicherzustellen, dass die Validierung die gleichen Werte
  ↳ verwendet.
RequiredFieldValidator_gebot.ErrorMessage =
  ↳ RangeValidator_gebot.ErrorMessage

' Zeigt das Detail-Panel für das ausgewählte Objekt an, sodass der
  ↳ Nutzer die Informationen sehen und bearbeiten kann.
Panel_detail.Visible = True

End Sub

```

3.8 Button_speichern_Click

```

Protected Sub Button_speichern_Click(sender As Object, e As EventArgs)
  ↳ Handles Button_speichern.Click
' Diese Methode wird aufgerufen, wenn der Nutzer den "Speichern"-Button
  ↳ drückt,
' um Änderungen am ausgewählten Objekt zu speichern, z.B. das Gebot
  ↳ oder den Objektnamen.

' Lädt die Projekt-XML-Datei, um die aktuellen Objektdaten zu
  ↳ verarbeiten.
Dim projekt_xml = XElement.Load(projekt_xml_datei)

' Holt die ID des ausgewählten Objekts aus den Daten des GridView,
' um das Objekt in der XML-Datei zu finden.
Dim objekt_id As String = GridView_objekte.DataKeys(GridView_objekte.Se_
  ↳ lectedIndex).Value.ToString()

' Führt eine LINQ-Abfrage aus, um das spezifische Objekt anhand der
  ↳ Objekt-ID
' in der Projekt-XML-Datei zu finden.
Dim objekt =
  From obj In projekt_xml.<objekte>.<objekt>
  Where obj.<objekt_id>.Value = objekt_id

' Überprüft, ob der Besitzer des Objekts gewechselt wird, indem die
  ↳ aktuelle
' Besitzer-ID mit der Nutzer-ID verglichen wird.
If Not objekt.<besitzer_id>.Value = nutzer_id Then

  ' Falls der Besitzer wechselt, wird der alte Besitzer auf
  ↳ "unzufrieden" gesetzt.

```

```
' Die Nutzer-ID des alten Besitzers wird extrahiert.
Dim alter_besitzer_id = objekt.<besitzer_id>.Value

' Führt eine LINQ-Abfrage aus, um den alten Besitzer in der
↪ XML-Datei zu finden.
Dim alter_besitzer =
    From nutzer In projekt_xml.<nutzers>.<nutzer>
    Where nutzer.<nutzer_id>.Value = alter_besitzer_id

' Setzt den Zufriedenheitsstatus des alten Besitzers auf "false".
alter_besitzer.<zufrieden>.Value = "false"

End If

' Setzt die Besitzer-ID des Objekts auf die Nutzer-ID des aktuellen
↪ Nutzers.
' Damit wird der aktuelle Nutzer zum neuen Besitzer des Objekts.
objekt.<besitzer_id>.Value = nutzer_id

' Setzt das neue Gebot für das Objekt auf den Wert, der im Textfeld
↪ "Textbox_objekt_gebot" eingegeben wurde.
' Der CInt-Befehl stellt sicher, dass der Wert als Integer gespeichert
↪ wird, damit z.B. "010" als 10 interpretiert wird.
objekt.<objekt_gebot>.Value = CInt(Textbox_objekt_gebot.Text)

' Speichert den neuen oder geänderten Objektnamen, indem der Inhalt des
↪ Textfelds "Textbox_objekt_name"
' verwendet wird. Der Trim-Befehl entfernt eventuell überflüssige
↪ Leerzeichen am Anfang und Ende.
objekt.<objekt_name>.Value = Textbox_objekt_name.Text.Trim()

' Speichert die aktualisierten Daten in der Projekt-XML-Datei.
projekt_xml.Save(projekt_xml_datei)

' Aktualisiert das GridView, um die Änderungen an den Objekten zu
↪ reflektieren,
' z.B. geänderte Namen oder Gebote.
GridView_darstellen(Nothing, Nothing)

' Aktualisiert das Guthaben des Nutzers auf der Seite, da das Gebot
↪ eines Objekts
' die Höhe des verfügbaren Guthabens beeinflusst.
Guthaben_anzeigen()

End Sub
```

3.9 Guthaben_anzeigen

```
Private Sub Guthaben_anzeigen()  
    ' Diese Methode berechnet und zeigt das verfügbare Guthaben des  
    ↪ aktuellen Nutzers an.  
    ' Das Guthaben wird basierend auf den Geboten des Nutzers auf  
    ↪ verschiedene Objekte berechnet.  
  
    ' Lädt die Projekt-XML-Datei, um die darin gespeicherten Objekte und  
    ↪ deren Gebote zu verarbeiten.  
Dim projekt_xml = XElement.Load(projekt_xml_datei)  
  
    ' Führt eine LINQ-Abfrage aus, um alle Objekte zu finden, deren  
    ↪ Besitzer der aktuelle Nutzer ist.  
    ' Für jedes Objekt, das dem aktuellen Nutzer gehört, wird das Gebot  
    ↪ summiert.  
Dim summe_meiner_gebote =  
    (From objekt In projekt_xml.<objekte>.<objekt>  
     Where objekt.<besitzer_id>.Value = nutzer_id  
     Select CInt(objekt.<objekt_gebot>.Value)).Sum()  
  
    ' Das verfügbare Guthaben des Nutzers wird berechnet, indem von einem  
    ↪ Startbetrag von 1000  
    ' die Summe aller Gebote des Nutzers abgezogen wird.  
Label_guthaben.Text = 1000 - summe_meiner_gebote  
  
    ' Der berechnete Wert wird im Label "Label_guthaben" auf der Seite  
    ↪ angezeigt,  
    ' sodass der Nutzer sein aktuelles verfügbares Guthaben sehen kann.  
End Sub
```

3.10 Button_loeschen_Click

```
Private Sub Button_loeschen_Click(sender As Object, e As EventArgs)  
    ↪ Handles Button_loeschen.Click  
    ' Diese Methode wird aufgerufen, wenn der Nutzer den "Löschen"-Button  
    ↪ drückt.  
    ' Sie dient dazu, das ausgewählte Objekt aus der Projekt-XML-Datei zu  
    ↪ laden und dem Nutzer anzuzeigen,  
    ' dass das Objekt gelöscht werden soll.  
  
    ' Lädt die Projekt-XML-Datei, um die darin gespeicherten Objekte zu  
    ↪ verarbeiten.  
Dim projekt_xml = XElement.Load(projekt_xml_datei)
```

```
' Holt die ID des ausgewählten Objekts aus den Daten des GridView,  
' um das entsprechende Objekt in der XML-Datei zu finden.  
Dim objekt_id As String = GridView_objekte.DataKeys(GridView_objekte.Se_  
↪ lectedIndex).Value.ToString()  
  
' Führt eine LINQ-Abfrage aus, um das Objekt in der XML-Datei zu  
↪ finden,  
' das der ID des ausgewählten Objekts entspricht.  
Dim objekt =  
    From obj In projekt_xml.<objekte>.<objekt>  
    Where obj.<objekt_id>.Value = objekt_id  
  
' Setzt den Namen des zu löschenden Objekts in das Label  
↪ "Label_loeschen",  
' um dem Nutzer den Namen des Objekts anzuzeigen, das gelöscht wird.  
Label_loeschen.Text = objekt.<objekt_name>.Value & " wird  
↪ unwiderbringlich gelöscht."  
  
' Macht das Lösch-Panel sichtbar, sodass der Nutzer die Löschung des  
↪ Objekts bestätigen kann.  
Panel_loeschen.Visible = True  
  
End Sub
```

3.11 Button_loeschen_ok_Click

```
Protected Sub Button_loeschen_ok_Click(sender As Object, e As  
↪ EventArgs) Handles Button_loeschen_ok.Click  
' Diese Methode wird aufgerufen, wenn der Nutzer das Löschen eines  
↪ Objekts bestätigt (OK-Klick).  
' Sie entfernt das ausgewählte Objekt aus der Projekt-XML-Datei und  
↪ löscht auch die entsprechende Bilddatei.  
  
' Lädt die Projekt-XML-Datei, um die Daten der darin gespeicherten  
↪ Objekte zu bearbeiten.  
Dim projekt_xml = XElement.Load(projekt_xml_datei)  
  
' Holt die ID des ausgewählten Objekts aus den Daten des GridView,  
' um das entsprechende Objekt in der XML-Datei zu finden.  
Dim objekt_id As String = GridView_objekte.DataKeys(GridView_objekte.Se_  
↪ lectedIndex).Value.ToString()  
  
' Führt eine LINQ-Abfrage aus, um das Objekt in der XML-Datei zu  
↪ finden,  
' das der ID des ausgewählten Objekts entspricht.
```



```
Dim objekt =
    From obj In projekt_xml.<objekte>.<objekt>
    Where obj.<objekt_id>.Value = objekt_id

' Setzt den Namen des zu löschenden Objekts in ein Label,
' um den Namen des gelöschten Objekts dem Nutzer anzuzeigen.
Label1.Text = objekt.<name>.Value

' Entfernt das ausgewählte Objekt aus der XML-Struktur.
objekt.Remove()

' Speichert die Änderungen in der Projekt-XML-Datei, nachdem das Objekt
  ↳ gelöscht wurde.
projekt_xml.Save(projekt_xml_datei)

' Bestimmt den Pfad der Bilddatei des gelöschten Objekts,
' indem der physische Pfad der Bilddatei mit der Objekt-ID erstellt
  ↳ wird.
Dim bild_datei = MapPath("bilder/" & objekt_id & ".jpg")

' Überprüft, ob die Bilddatei für das Objekt existiert.
' Falls ja, wird die Datei vom Server gelöscht.
If IO.File.Exists(bild_datei) Then
    IO.File.Delete(bild_datei)
End If

' Aktualisiert das GridView, um die gelöschten Daten zu entfernen und
  ↳ die restlichen Objekte anzuzeigen.
GridView_darstellen(Nothing, Nothing)

' Aktualisiert das Guthaben des Nutzers, da das Löschen eines Objekts
  ↳ das Guthaben beeinflussen kann.
Guthaben_anzeigen()

End Sub
```

3.12 Nutzer_name

```
Private Function Nutzer_name(nutzer_id As String) As String
' Diese Funktion gibt den Namen eines Nutzers zurück, basierend auf der
  ↳ übergebenen Nutzer-ID.
' Sie durchsucht die Projekt-XML-Datei nach dem entsprechenden Nutzer
  ↳ und gibt dessen Namen zurück.

' Lädt die Projekt-XML-Datei, um die darin gespeicherten Nutzerdaten zu
  ↳ durchsuchen.
```

```
Dim projekt_xml = XElement.Load(projekt_xml_datei)

' Führt eine LINQ-Abfrage aus, um den Nutzer in der XML-Datei zu
↪ finden,
' dessen Nutzer-ID mit der übergebenen nutzer_id übereinstimmt.
Dim aktueller_nutzer =
    From nutzer In projekt_xml.<nutzers>.<nutzer>
    Where nutzer.<nutzer_id>.Value = nutzer_id

' Gibt den Namen des gefundenen Nutzers als Ergebnis der Funktion
↪ zurück.
' Dieser Wert wird aus dem XML-Element <nutzer_name> extrahiert.
Return aktueller_nutzer.<nutzer_name>.Value

End Function
```

3.13 Button_zufrieden_Click

```
Protected Sub Button_zufrieden_Click(sender As Object, e As EventArgs)
↪ Handles Button_zufrieden.Click
' Diese Methode wird aufgerufen, wenn der Nutzer den "Zufrieden"-Button
↪ klickt.
' Sie setzt den Zufriedenheitsstatus des aktuellen Nutzers auf "true"
↪ in der Projekt-XML-Datei.

' Lädt die Projekt-XML-Datei, um die Nutzerdaten zu bearbeiten.
Dim projekt_xml = XElement.Load(projekt_xml_datei)

' Führt eine LINQ-Abfrage aus, um den aktuellen Nutzer in der XML-Datei
↪ zu finden,
' dessen Nutzer-ID der aktuell angemeldeten nutzer_id entspricht.
Dim aktueller_nutzer =
    From nutzer In projekt_xml.<nutzers>.<nutzer>
    Where nutzer.<nutzer_id>.Value = nutzer_id

' Setzt den Zufriedenheitsstatus des aktuellen Nutzers auf "true".
' Das XML-Element <zufrieden> wird entsprechend geändert.
aktueller_nutzer.<zufrieden>.Value = "true"

' Speichert die Änderungen in der Projekt-XML-Datei, um
↪ sicherzustellen,
' dass der geänderte Zufriedenheitsstatus permanent gespeichert wird.
projekt_xml.Save(projekt_xml_datei)

End Sub
```

3.14 Button_unzufrieden_Click

```
Protected Sub Button_unzufrieden_Click(sender As Object, e As
    EventArgs) Handles Button_unzufrieden.Click
    ' Diese Methode wird aufgerufen, wenn der Nutzer den
    ' "Unzufrieden"-Button klickt.
    ' Sie setzt den Zufriedenheitsstatus des aktuellen Nutzers auf "false"
    ' in der Projekt-XML-Datei.

    ' Lädt die Projekt-XML-Datei, um die Nutzerdaten zu bearbeiten.
    Dim projekt_xml = XElement.Load(projekt_xml_datei)

    ' Führt eine LINQ-Abfrage aus, um den aktuellen Nutzer in der XML-Datei
    ' zu finden,
    ' dessen Nutzer-ID der aktuell angemeldeten nutzer_id entspricht.
    Dim aktueller_nutzer =
        From nutzer In projekt_xml.<nutzers>.<nutzer>
        Where nutzer.<nutzer_id>.Value = nutzer_id

    ' Setzt den Zufriedenheitsstatus des aktuellen Nutzers auf "false".
    ' Das XML-Element <zufrieden> wird entsprechend geändert.
    aktueller_nutzer.<zufrieden>.Value = "false"

    ' Speichert die Änderungen in der Projekt-XML-Datei, um
    ' sicherzustellen,
    ' dass der geänderte Zufriedenheitsstatus permanent gespeichert wird.
    projekt_xml.Save(projekt_xml_datei)

End Sub
```

3.15 LinkButton lesezeichen_erstellen_Click

```
Protected Sub LinkButton lesezeichen_erstellen_Click(sender As Object,
    e As EventArgs) Handles LinkButton lesezeichen_erstellen.Click
    ' Diese Methode wird aufgerufen, wenn der Nutzer auf den "Lesezeichen
    ' erstellen"-LinkButton klickt.
    ' Sie erstellt einen Link, den der Nutzer als Lesezeichen speichern
    ' kann, um das aktuelle Projekt und den Nutzerzustand zu speichern.

    ' Erstellt einen URL-Link basierend auf der aktuellen Projekt-ID und
    ' Nutzer-ID,
    ' den der Nutzer als Lesezeichen speichern kann. Der Link ermöglicht
    ' es, das Projekt und den Nutzer
    ' direkt wieder aufzurufen.
    Label_link lesezeichen.Text =
```

```
        "https://m-server.fk5.hs-bremen.de/partitio/partitio.aspx?projekt="
        ↪ &
        projekt_id &
        "&nutzer=" &
        nutzer_id

' Macht das Panel für die Lesezeichen-Erstellung sichtbar, sodass der
↪ Nutzer den erstellten Link sehen und speichern kann.
Panel lesezeichen_erstellen.Visible = True

End Sub
```

3.16 CustomValidator_name_ServerValidate

```
Protected Sub CustomValidator_name_ServerValidate(source As Object,
↪ args As ServerValidateEventArgs) Handles
↪ CustomValidator_name.ServerValidate
' Diese Methode wird aufgerufen, um zu überprüfen, ob der eingegebene
↪ Name bereits als Nutzernamen in der XML-Datei vorhanden ist.
' Sie wird als Server-Validierung für das Eingabefeld des Nutzernamens
↪ verwendet.

' Holt den eingegebenen Namen aus den Validierungsargumenten und
↪ entfernt überflüssige Leerzeichen.
Dim name = args.Value.Trim()

' Holt die Projekt-ID aus dem Query-String, um die richtige
↪ Projekt-XML-Datei zu laden.
Dim projekt_id = Request.QueryString("projekt")

' Bestimmt den physischen Pfad der Projekt-XML-Datei basierend auf der
↪ Projekt-ID.
Dim projekt_xml_datei = MapPath("projekte/" & projekt_id & ".xml")

' Lädt die Projekt-XML-Datei, um die darin gespeicherten Nutzerdaten zu
↪ durchsuchen.
Dim projekt_xml = XElement.Load(projekt_xml_datei)

' Führt eine LINQ-Abfrage aus, um alle Nutzernamen in der XML-Datei zu
↪ extrahieren.
' Jeder Nutzernamen wird in einer Liste (alle_nutzer_namen)
↪ gespeichert.
' Die einfache Variante, um alle Nutzernamen zu listen
' Dim alle_nutzer = projekt_xml...<nutzer_name>.Value
' funktioniert hier nicht.
```

```
' Es steht dann in jeder Zeile des Gridviews nur ein Buchstabe des
↪ ersten Namens.
' Deshalb wird die "normale" Variante verwendet:

Dim alle_nutzer_namen =
  From nutzer In projekt_xml.<nutzers>.<nutzer>
  Select nutzer.<nutzer_name>.Value

' Überprüft, ob der eingegebene Name bereits in der Liste der
↪ Nutzernamen vorhanden ist.
' Wenn der Name bereits existiert, schlägt die Validierung fehl.
For Each aktueller_nutzer_name In alle_nutzer_namen
  If aktueller_nutzer_name.ToLower = name.ToLower Then
    ' Setzt die Validierung auf "ungültig", da der Name bereits
    ↪ existiert.
    args.IsValid = False

    ' Zeigt die Liste aller Nutzernamen im GridView an, um dem
    ↪ Nutzer die bereits verwendeten Namen zu zeigen.
    GridView_nutzer.DataSource = alle_nutzer_namen
    GridView_nutzer.DataBind()

    ' Macht das Panel mit der Liste aller Nutzer sichtbar, sodass
    ↪ der Nutzer die Namen sehen kann.
    Panel_alle_nutzer.Visible = True

    ' Beendet die Validierung, da der Name bereits existiert.
    Exit Sub
  End If
Next

' Wenn der eingegebene Name nicht existiert, wird ein neuer Nutzer mit
↪ einer neuen Nutzer-ID erstellt.

' Generiert eine neue eindeutige Nutzer-ID.
Dim nutzer_id = System.Guid.NewGuid.ToString()

' Erstellt ein neues XML-Element für den Nutzer, das die Nutzer-ID, den
↪ Nutzernamen und den Zufriedenheitsstatus enthält.
Dim neuer_nutzer As XElement =
  <nutzer>
    <nutzer_id><%= nutzer_id %></nutzer_id>
    <nutzer_name><%= name %></nutzer_name>
    <zufrieden>false</zufrieden>
  </nutzer>

' Fügt den neuen Nutzer der Liste der Nutzer in der XML-Datei hinzu.
```

```
projekt_xml.<nutzers>.FirstOrDefault.Add(neuer_nutzer)

' Speichert die geänderte XML-Datei, um den neuen Nutzer zu speichern.
projekt_xml.Save(projekt_xml_datei)

' Setzt eine Session-Variable, die anzeigt, dass der Nutzer ein
↳ Lesezeichen für den neuen Teilnehmer erstellen kann.
Session("Lesezeichen_erstellen_anzeigen") = "true"

' Leitet den Nutzer auf die aktuelle Seite mit den neuen Projekt- und
↳ Nutzerparametern weiter.
Response.Redirect("partitio.aspx?projekt=" & projekt_id & "&nutzer=" &
↳ nutzer_id)

End Sub
```

3.17 LinkButton_alle_zufrieden_Click

```
Protected Sub LinkButton_alle_zufrieden_Click(sender As Object, e As
↳ EventArgs) Handles LinkButton_alle_zufrieden.Click
' Diese Methode wird aufgerufen, wenn der Nutzer auf den LinkButton
↳ "Alle zufrieden" klickt.
' Sie zeigt die Zufriedenheitsstatus aller Nutzer in einem GridView
↳ an.

' Lädt die Projekt-XML-Datei, um die darin gespeicherten Nutzerdaten zu
↳ verarbeiten.
Dim projekt_xml = XElement.Load(projekt_xml_datei)

' Führt eine LINQ-Abfrage aus, um alle Nutzer und deren
↳ Zufriedenheitsstatus aus der XML-Datei zu extrahieren.
' Für jeden Nutzer wird der Name und der entsprechende
↳ Zufriedenheitsstatus (ob "zufrieden" oder "unzufrieden")
↳ extrahiert.
Dim zufriedenheiten =
  From nutzer In projekt_xml.<nutzers>.<nutzer>
  Select
  Nutzer = nutzer.<nutzer_name>.Value,
  Zufriedenheit = If(nutzer.<zufrieden>.Value, "ist zufrieden", "ist
↳ unzufrieden")

' Weist das Ergebnis der LINQ-Abfrage (die Namen und
↳ Zufriedenheitsstatus der Nutzer) dem GridView als Datenquelle zu.
GridView_zufriedenheit.DataSource = zufriedenheiten
```

```
' Bindet die Daten an das GridView, sodass die Namen und  
↳ Zufriedenheitsstatus der Nutzer auf der Seite angezeigt werden.  
GridView_zufriedenheit.DataBind()  
  
' Macht das Panel sichtbar, das die Zufriedenheitsdaten der Nutzer  
↳ enthält,  
' sodass der Nutzer die Daten auf der Benutzeroberfläche sehen kann.  
Panel_zufriedenheit.Visible = True  
  
End Sub
```

3.18 LinkButton_aktualisieren_Click

```
Protected Sub LinkButton_aktualisieren_Click(sender As Object, e As  
↳ EventArgs) Handles LinkButton_aktualisieren.Click  
' Diese Methode wird aufgerufen, wenn der Nutzer auf den  
↳ "Aktualisieren"-LinkButton klickt.  
' Sie aktualisiert die Anzeige der Objekte im GridView, basierend auf  
↳ der aktuellen Sortierung.  
  
' Ruft die Methode "GridView_darstellen" auf, um das GridView neu zu  
↳ füllen.  
' Die aktuellen Sortierparameter ("SortExpression" und "SortDirection")  
↳ werden aus dem ViewState geladen.  
' Diese Parameter sorgen dafür, dass die Objekte in der gleichen  
↳ Reihenfolge wie vor der Aktualisierung angezeigt werden.  
GridView_darstellen(ViewState("SortExpression"),  
↳ ViewState("SortDirection"))  
  
End Sub
```

3.19 LinkButton_neue_teilnehmer_Click

```
Protected Sub LinkButton_neue_teilnehmer_Click(sender As Object, e As  
↳ EventArgs) Handles LinkButton_neue_teilnehmer.Click  
' Diese Methode wird aufgerufen, wenn der Nutzer auf den LinkButton  
↳ "Neue Teilnehmer einladen" klickt.  
' Sie erstellt einen Einladungslink für neue Teilnehmer und stellt  
↳ diesen dem aktuellen Nutzer bereit.  
  
' Erstellt einen Einladungslink basierend auf der aktuellen  
↳ Projekt-ID.
```

```
' Der neue Teilnehmer wird standardmäßig mit der Nutzer-ID "42"
↪ eingeladen.
Label_link_neue_teilnehmer.Text =
    "https://m-server.fk5.hs-bremen.de/partitio/partitio.aspx?projekt="
    ↪ &
    projekt_id &
    "&nutzer=42"

' Erstellt den Betreff für die Einladung, die per E-Mail verschickt
↪ werden soll.
Dim betreff =
    "Einladung zur Teilnahme an unserem Partitio-Projekt"

' Erzeugt den Inhalt der E-Mail, die die Einladung zur Teilnahme am
↪ Projekt enthält.
' Hierbei wird die Projekt-ID und die standardmäßige Nutzer-ID "42" als
↪ Platzhalter für den neuen Teilnehmer verwendet.
' Die URL-Zeichen wie Gleichheitszeichen und &-Zeichen werden
↪ ordnungsgemäß kodiert, um Kompatibilitätsprobleme in E-Mails zu
↪ vermeiden.
Dim inhalt =
    "Hiermit lade ich dich zu unserem gemeinsamen Partitio-Projekt ein.
    ↪ " &
    "%0D%0A%0D%0A" &
    "Bitte benutze dazu den folgenden Link:" &
    "%0D%0A%0D%0A" &
    "https://m-server.fk5.hs-bremen.de/partitio/partitio.aspx%3Fprojekt_
    ↪ %3D"
    ↪ &
    projekt_id &
    "%26nutzer%3D42" &
    "%0D%0A%0D%0A" &
    "Mit freundlichen Grüßen" &
    "%0D%0A%0D%0A" &
    Nutzer_name(nutzer_id)

' Erstellt einen Mailto-Link, der eine E-Mail-Adresse für die neuen
↪ Teilnehmer enthält,
' zusammen mit dem zuvor definierten Betreff und Inhalt.
' Durch Anklicken des Links wird im E-Mail-Programm des Nutzers eine
↪ neue E-Mail mit dem Einladungstext geöffnet.
Label_email_neue_teilnehmer.Text =
    "<a href='mailto:E-Mail-Adressen_neuer_Teilnehmerinnen@partitio.de"
    ↪ &
    "?subject=" &
    betreff &
    "&body=" &
```



```
    inhalt &
    "'>E-Mail</a>"

    ' Macht das Panel für die Einladung neuer Teilnehmerinnen sichtbar,
    ' sodass der Nutzer den Einladungslink und den vorbereiteten
    ' ↪ E-Mail-Link sehen und verwenden kann.
    Panel_neue_teilnehmerinnen.Visible = True

End Sub
```

3.20 LinkButton_hilfe_Click

```
Protected Sub LinkButton_hilfe_Click(sender As Object, e As EventArgs)
    ↪ Handles LinkButton_hilfe.Click
    ' Diese Methode wird aufgerufen, wenn der Nutzer auf den LinkButton
    ' ↪ "Hilfe" klickt.
    ' Sie leitet den Nutzer auf ein PDF-Dokument weiter, das die
    ' ↪ Hilfeanweisungen für die Anwendung enthält.

    ' Führt eine Umleitung zur PDF-Datei "partitio.pdf" aus, die auf dem
    ' ↪ Server liegt.
    ' Diese Datei enthält die Hilfedokumentation für die Anwendung,
    ' welche dem Nutzer weiterführende Anweisungen und Informationen
    ' ↪ bietet.
    Response.Redirect("partitio.pdf")

End Sub
```

3.21 LinkButton_seitenleiste_oeffnen_Click

```
Protected Sub LinkButton_seitenleiste_oeffnen_Click(sender As Object, e
    ↪ As EventArgs) Handles LinkButton_seitenleiste_oeffnen.Click
    ' Diese Methode wird aufgerufen, wenn der Nutzer auf den LinkButton
    ' ↪ "Seitenleiste öffnen" klickt.
    ' Sie macht das Panel für die Seitenleiste sichtbar, das möglicherweise
    ' ↪ zusätzliche Optionen oder Navigationselemente enthält.

    ' Macht das Panel "Panel_seitenleiste" sichtbar, damit der Nutzer auf
    ' ↪ die Inhalte der Seitenleiste zugreifen kann.
    ' Die Seitenleiste könnte Navigationslinks, Einstellungen oder weitere
    ' ↪ Funktionen enthalten.
    Panel_seitenleiste.Visible = True
```

```
End Sub
```

3.22 GridView_objekte_Sorting

```
Protected Sub GridView_objekte_Sorting(sender As Object, e As
    GridViewSortEventArgs)
    ' Diese Methode wird aufgerufen, wenn der Nutzer auf die Kopfzeile
    ' einer Spalte im GridView klickt,
    ' um die Objekte nach dieser Spalte zu sortieren.

    ' Bestimmt die Sortierrichtung basierend auf der aktuellen
    ' Sortierspalte.
    ' Die Methode "GetSortDirection" wird verwendet, um die Sortierrichtung
    ' (aufsteigend oder absteigend) zu bestimmen.
    Dim sortDirection As String = GetSortDirection(e.SortExpression)

    ' Ruft die Methode "GridView_darstellen" auf, um das GridView mit den
    ' sortierten Daten neu zu füllen.
    ' Die Spalte, nach der sortiert werden soll (e.SortExpression), und die
    ' Sortierrichtung (sortDirection) werden übergeben.
    GridView_darstellen(e.SortExpression, sortDirection)

End Sub
```

3.23 GetSortDirection

```
Private Function GetSortDirection(column As String) As String
    ' Diese Methode bestimmt die Sortierrichtung (aufsteigend oder
    ' absteigend) basierend auf der aktuellen Sortierspalte.
    ' Wenn der Nutzer erneut auf eine bereits sortierte Spalte klickt, wird
    ' die Sortierrichtung umgekehrt.

    ' Standardmäßig wird die Sortierrichtung auf "ASC" (aufsteigend)
    ' gesetzt.
    Dim sortDirection As String = "ASC"

    ' Holt den aktuellen Sortierausdruck (die Spalte, nach der zuletzt
    ' sortiert wurde) aus dem ViewState.
    ' "TryCast" versucht, den Sortierausdruck in einen String zu
    ' konvertieren.
    Dim sortExpression As String = TryCast(ViewState("SortExpression"),
    ' String)
```

```
' Überprüft, ob ein Sortierausdruck im ViewState vorhanden ist (d.h.,  
↪ ob bereits nach einer Spalte sortiert wurde).  
If sortExpression IsNot Nothing Then  
    ' Wenn die aktuelle Spalte (column) die gleiche ist wie die zuletzt  
    ↪ sortierte Spalte,  
    ' wird die Sortierrichtung umgekehrt.  
    If sortExpression = column Then  
        ' Holt die letzte Sortierrichtung (ASC oder DESC) aus dem  
        ↪ ViewState.  
        Dim lastDirection As String =  
        ↪ TryCast(ViewState("SortDirection"), String)  
  
        ' Wenn die letzte Sortierrichtung "ASC" (aufsteigend) war,  
        ↪ ändert sie sich zu "DESC" (absteigend).  
        If lastDirection IsNot Nothing AndAlso lastDirection = "ASC"  
        ↪ Then  
            sortDirection = "DESC"  
        End If  
    End If  
End If  
  
' Speichert die neue Sortierrichtung im ViewState, um sie für  
↪ zukünftige Klicks auf die gleiche Spalte zu merken.  
ViewState("SortDirection") = sortDirection  
  
' Speichert die Spalte, nach der gerade sortiert wird, im ViewState.  
ViewState("SortExpression") = column  
  
' Gibt die ermittelte Sortierrichtung (ASC oder DESC) zurück.  
Return sortDirection  
  
End Function
```

Kapitel 4

partitio.aspx

Die Datei `partitio.aspx` ist eine ASP.NET-Webseite, die als Benutzeroberfläche für ein Projektmanagementsystem dient. Sie enthält sowohl HTML- als auch serverseitige ASP.NET-Elemente, um den Nutzern verschiedene Funktionen und interaktive Bedienelemente zur Verfügung zu stellen:

Seitenleiste und Navigation: Die Datei beinhaltet eine ausblendbare Seitenleiste (Panel `Panel_seitenleiste`) mit verschiedenen Navigations-Links und Optionen für Benutzeraktionen. Über diese können Nutzer beispielsweise Lesezeichen erstellen, Teilnehmer einladen und Unterstützung anfordern.

Kamerasteuerung und Foto-Session: Es gibt Funktionen, mit denen die Nutzer die Kamera ihres Geräts aktivieren und Bilder aufnehmen können. Der Button `button_kamera_starten` öffnet eine Foto-Session und ermöglicht es, Bilder für das Projekt aufzunehmen und hochzuladen.

GridView-Komponenten für Datenanzeige: `partitio.aspx` nutzt mehrere `GridView`-Steuerelemente, um Listen und Daten übersichtlich anzuzeigen, wie etwa Teilnehmer und ihre Zufriedenheitsbewertungen. Die `GridView`-Elemente sind sortierbar und lassen sich in verschiedene Ansichten aufteilen.

Panels für Benutzeraktionen: Die Datei verwendet mehrere Panels (z. B. für Lesezeichen, neue Teilnehmer, Zufriedenheit), die gezielt eingeblendet werden können, um den Nutzern spezialisierte Interaktionsmöglichkeiten zu bieten. So kann der Nutzer schnell zwischen verschiedenen Aktionen wie dem Einladen neuer Teilnehmer und dem Anzeigen von Projektstatus wechseln.

Automatisches und periodisches Neuladen: Ein JavaScript-Intervall sorgt dafür, dass die Seite alle zehn Minuten automatisch neu lädt, damit der Nutzer stets aktuelle Informationen sieht.

Zusammengefasst bietet `partitio.aspx` eine umfassende Benutzeroberfläche für die Verwaltung und Interaktion innerhalb eines Projekts, wobei dynamische Server- und Client-Skripte nahtlos zusammenarbeiten, um ein flexibles und interaktives Erlebnis zu schaffen.

4.1 Page-Direktive

```
<!-- Definiert die ASP.NET-Seite mit VB als Sprache und weist die  
↳ Code-Behind-Datei zu -->  
<%@ Page  
  Language="VB"  
  AutoEventWireup="false"  
  CodeFile="partitio.aspx.vb"  
  Inherits="Partitio"  
%>
```

4.2 Header

```
<!-- Deklariert den Dokumenttyp als HTML -->  
<!DOCTYPE html>  
  
<!-- Das öffnende html-Tag definiert den HTML-Namespace und legt die  
↳ Klasse für das Styling fest -->  
<html  
  class="w3-text-dark-gray"  
  xmlns="http://www.w3.org/1999/xhtml">  
  
<head runat="server">  
  
<!-- Setzt den Content-Typ für die Seite, hier auf UTF-8  
↳ Zeichenkodierung -->  
<meta  
  http-equiv="Content-Type"  
  content="text/html; charset=utf-8"  
>  
  
<!-- Stellt sicher, dass die Seite für mobile Geräte optimiert wird  
↳ -->  
<meta  
  name="viewport"  
  content="width=device-width, initial-scale=1.0"  
>  
  
<!-- Bindet eine externe CSS-Datei mit einer Versionsnummer ein, um  
↳ Cache-Probleme zu vermeiden -->  
<link  
  href="partitio.css?v=42"  
  rel="stylesheet"  
>  
  
<!-- Verlinkt die W3.CSS Bibliothek für Layout- und Styling-Regeln -->  
<link
```

```
    rel="stylesheet"
    href="https://www.w3schools.com/w3css/4/w3.css"
  />

  <!-- Font Awesome wird für Icons genutzt, z.B. für den Hamburger-Button
  ↪ -->
  <link
    rel="stylesheet"
    href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css_
    ↪ /font-awesome.min.css"
  />

  <!-- Material Icons werden für den Schließ-Button genutzt -->
  <link
    rel="stylesheet"
    href="https://fonts.googleapis.com/icon?family=Material+Icons"
  />

  <!-- Definiert den Titel der Webseite -->
  <title>Partitio</title>

  <!-- SVG-Version des Favicons für unterstützte Browser -->
  <link
    rel="icon"
    href="partitio.svg?v=1.0"
    type="image/svg+xml"
  />

  <!-- ICO-Version des Favicons für Browser, die SVG nicht unterstützen
  ↪ -->
  <link
    rel="icon"
    href="partitio.ico?v=1.1"
    type="image/x-icon"
  />

  <!-- Shortcut-Icon, speziell für ältere Browser, die diese Methode
  ↪ verwenden -->
  <link
    rel="shortcut icon"
    href="partitio.ico?v=1.1"
    type="image/x-icon"
  />

</head>
```

4.3 JavaScript

```
<script>
:
:
:
</script>
```

4.3.1 setInterval

```
// Automatisches Neuladen der gesamten Seite alle 10 Minuten
setInterval(function () {
    // Setzt die Seite auf den aktuellen Pfad und die aktuellen
    ↪ Abfrageparameter zurück (neu laden)
    window.location.href = window.location.pathname +
    ↪ window.location.search;
}, 600000); // 600000 Millisekunden = 10 Minuten
```

4.3.2 addEventListener

```
// Schließen der Seitenleiste, wenn der Nutzer irgendwo klickt und die
↪ Seitenleiste geöffnet ist
document.addEventListener('click', function (event) {
    // Holt die Seitenleiste anhand der ClientID des ASP.NET Panels
    var Panel_seitenleiste_id = document.getElementById('<%=
    ↪ Panel_seitenleiste.ClientID %>');
    if (Panel_seitenleiste_id) {
        // Versteckt die Seitenleiste, wenn sie existiert
        Panel_seitenleiste_id.style.display = 'none';
    }
});
```

4.3.3 kopiere_in_die_zwischenablage

```
// Funktion, um den Text eines Elements in die Zwischenablage zu
↪ kopieren
function kopiere_in_die_zwischenablage(label_id) {
    // Kopiert den inneren Text des Elements mit der ID 'label_id' in
    ↪ die Zwischenablage
    navigator.clipboard.writeText(document.getElementById(label_id).inn_
    ↪ erText);
}
```

4.3.4 kamera_starten

```
// Variable für den Video-Stream
let stream = null;

// Funktion zum Starten der Kamera und Anzeigen des Video-Feeds
async function kamera_starten() {
  try {
    // Versucht, die Kamera mit der rückseitigen Kamera (Umgebung)
    ↪ zu starten
    stream = await navigator.mediaDevices.getUserMedia({ video: {
      ↪ facingMode: "environment" } });

    // Setzt den Video-Stream als Quelle für das Video-Element
    document.getElementById('videoElement').srcObject = stream;
  }
  catch (fehler) {
    // Loggt den Fehler und zeigt eine Warnung, wenn der Zugriff
    ↪ auf die Kamera nicht möglich ist
    console.log('Ein Fehler ist aufgetreten:', fehler);
    alert('Zugriff auf die Kamera nicht möglich oder abgelehnt.');  }

  // Zeigt das Panel für die Foto-Session an und scrollt zu diesem
  ↪ Abschnitt
  var panel_foto_session = document.getElementById('foto_session');
  panel_foto_session.style.display = 'block';
  panel_foto_session.scrollIntoView({ behavior: 'smooth' });

  // Versteckt den Button zum Starten der Kamera, um den mehrfachen
  ↪ Start zu verhindern
  document.getElementById('button_kamera_starten').style.display =
    ↪ 'none';
}
```

4.3.5 kamera_stoppen

```
// Funktion zum Stoppen der Kamera
async function kamera_stoppen() {
  if (stream) {
    // Beendet alle aktiven Video-Streams
    stream.getTracks().forEach(track => track.stop());
  }
}
```



```
    // Setzt den Stream zurück
    stream = null;
}

// Versteckt das Foto-Session-Panel
document.getElementById('foto_session').style.display = 'none';
// Zeigt den Button zum Starten der Kamera wieder an
document.getElementById('button_kamera_starten').style.display =
    ↪ 'block';
}
```

4.3.6 bild_aufnehmen

```
// Funktion zum Aufnehmen eines Bildes und Hochladen an den Server
async function bild_aufnehmen(projekt_id, nutzer_id) {
    // Generiert eine zufällige UUID für das Bild
    const bild_id = crypto.randomUUID();

    if (!stream) {
        // Zeigt eine Warnung, wenn kein Kamera-Stream aktiv ist
        alert('Zugriff auf die Kamera nicht möglich oder abgelehnt.');
```

```
        return;
    }

    // Holt das Video-Element
    var video = document.getElementById('videoElement');
```

```
    // Holt das Canvas-Element zum Zeichnen des Bildes
    var canvas = document.getElementById('canvas');
```

```
    // Holt den 2D-Kontext des Canvas
    var context = canvas.getContext('2d');
```

```
    // Holt die Breite des Video-Feeds
    var videoWidth = video.videoWidth;
```

```
    // Holt die Höhe des Video-Feeds
    var videoHeight = video.videoHeight;
```

```
    // Nimmt an, dass das Canvas quadratisch ist
    var canvasSize = canvas.width;
```

```
    // Berechnet den Maßstab für das Bild
    var scale = Math.max(canvasSize / videoWidth, canvasSize /
    ↪ videoHeight);
```

```
    // Berechnet die Position des Bildes auf dem Canvas (X-Achse)
    var x = (canvasSize / 2) - (videoWidth / 2) * scale;
```

```
    // Berechnet die Position des Bildes auf dem Canvas (Y-Achse)
    var y = (canvasSize / 2) - (videoHeight / 2) * scale;
```

```
// Zeichnet das Video auf das Canvas
context.drawImage(video, x, y, videoWidth * scale, videoHeight *
↪ scale);

// Wandelt das Canvas-Bild in eine Daten-URL um
const imageData = canvas.toDataURL('image/jpeg', 0.8);
// Erstellt daraus einen Blob
const blob = await fetch(imageData).then(res => res.blob());

// Erstellt ein FormData-Objekt
const daten = new FormData();
// Fügt die Bilddaten hinzu
daten.append('bild', blob, bild_id + '.jpg');
// Fügt die Projekt-ID hinzu
daten.append('projekt_id', projekt_id);
// Fügt die Nutzer-ID hinzu
daten.append('nutzer_id', nutzer_id);
// Fügt die Bild-ID hinzu
daten.append('bild_id', bild_id);

// Sendet das Bild über einen POST-Request an den Server
fetch('/partitio/bild_empfangen.ashx', {
  method: 'POST',
  body: daten
})
// Wartet auf die Antwort vom Server
.then(ergebnis => ergebnis.json())
// Loggt die Server-Antwort
.then(ergebnis_daten => console.log(ergebnis_daten))
.then(() => {
  // Simuliert einen Klick auf den Aktualisieren-Button
  var LinkButton_aktualisieren_id = '<%=
↪ LinkButton_aktualisieren.ClientID %>';
  document.getElementById(LinkButton_aktualisieren_id).click();
})
// Fehlerbehandlung beim Hochladen
.catch(fehler => console.log('Ein Fehler ist aufgetreten beim
↪ Hochladen:', fehler));

// Macht den Kamera-Start-Button wieder sichtbar
document.getElementById('button_kamera_starten').style.display =
↪ 'block';
}
```

4.4 body, form, ...

```

<!-- Beginnt den body-Tag, der den Hauptinhalt der Seite enthält -->
<body>

    <!-- Beginnt das Server-seitige Formular, das notwendig ist,
         um ASP.NET-Steuererelemente zu verwenden und Events auf dem
         ↪ Server zu verarbeiten -->
    <form
        id="form1"
        runat="server">

        <!-- Hier beginnt das ASP.NET-Panel für die Seitenleiste,
             das dazu dient, Inhalte dynamisch anzuzeigen oder zu
             ↪ verstecken -->
            :
            :
            :

        </form>

</body>

```

4.4.1 Panel_seitenleiste

```

<!-- Die Seitenleiste hat einen z-index von 1000
     (in partitio.css),
     damit sie über allem liegt. -->
<asp:Panel
    ID="Panel_seitenleiste"
    runat="server"
    CssClass="
        halbtransparent
        w3-sidebar
        w3-bar-block
        w3-card-4
        w3-animate-left"
    Visible="False">

<!-- Ein Container div mit verschiedenen CSS-Klassen für das Layout und
     ↪ Styling -->
<div class="
    w3-card
    w3-margin
    w3-round

```

```
w3-light-grey
automatische_breite">

<!-- LinkButton für das Schließen der Seitenleiste, wird als Icon
↳ (Material Icons) dargestellt -->
<asp:LinkButton
  ID="Button_zurueck"
  runat="server"
  Text="<i class='material-icons'>close</i>"
  CssClass="
  w3-button
  w3-block
  text-left" />

<!-- Absatz (Leerzeile) -->
<p />

<!-- LinkButton zum Erstellen eines Lesezeichens -->
<asp:LinkButton
  runat="server"
  CssClass="
  w3-button
  w3-block
  text-left"
  ID="LinkButton lesezeichen erstellen"
  Text="Lesezeichen erstellen" />

<!-- Absatz (Leerzeile) -->
<p />

<!-- LinkButton zum Hinzufügen neuer Teilnehmer -->
<asp:LinkButton
  runat="server"
  CssClass="
  w3-button
  w3-block
  text-left"
  ID="LinkButton_neue teilnehmer"
  Text="Neue Teilnehmerinnen" />

<!-- Absatz (Leerzeile) -->
<p />

<!-- LinkButton für eine Zufriedenheitsabfrage -->
<asp:LinkButton
  runat="server"
  CssClass="
```

```

        w3-button
        w3-block
        text-left"
        ID="LinkButton_alle_zufrieden"
        Text="Alle zufrieden?" />

<!-- Absatz (Leerzeile) -->
<p />

<!-- LinkButton für Hilfsinformationen -->
<asp:LinkButton
    runat="server"
    CssClass="
        w3-button
        w3-block
        text-left"
    ID="LinkButton_hilfe"
    Text="Hilfe" />

</div> <!-- Ende des Containers für den Inhalt der Seitenleiste -->

</asp:Panel>

```

4.4.2 Header (w3-top)

```

<!-- Der Header (w3-top) und der Footer (w3-bottom) müssen außerhalb
↳ des eigentlichen Seiteninhalts (w3-content) stehen.
    Sie werden zweistufig definiert, um auf dem Desktop nicht bis an
↳ den Rand zu ragen.
    Auf mobilen Geräten ist diese Einschränkung irrelevant. -->
<div class="w3-top">

    <!-- Container für den Header-Inhalt mit den CSS-Klassen für das
↳ Layout -->
    <div class="w3-content w3-teal w3-padding">

        <!-- Hamburger-Icon zum Öffnen der Seitenleiste, hier als
↳ LinkButton mit einem Font Awesome Icon dargestellt -->
        <asp:LinkButton
            ID="LinkButton_seitenleiste_oeffnen"
            runat="server"
            CausesValidation="False"
            Text="<i class='fa fa-bars'></i>"
        </asp:LinkButton>

        <!-- Abstand durch nicht brechende Leerzeichen -->

```

```

    &nbsp; &nbsp; &nbsp; &nbsp; Partitio -

    <!-- ASP.NET Label für die Anzeige des Nutzernamens -->
    <asp:Label
        ID="Label_nutzer_name"
        runat="server">
    </asp:Label>

    </div> <!-- Ende des w3-content Containers für den Header -->

</div> <!-- Ende des w3-top Headers -->

<!-- Zwei Zeilenumbrüche (br-Tags), notwendig, um sicherzustellen,
     dass der Seiteninhalt nicht unter dem Header verschwindet -->
<br />
<br />

```

4.4.3 Footer (w3-bottom)

```

<!-- Footer-Bereich, der am unteren Rand der Seite verankert ist -->
<div class="w3-bottom">

    <!-- Container für den Footer-Inhalt, Layout durch w3-content,
         ↳ w3-teal und w3-padding -->
    <div class="
        w3-content
        w3-teal
        w3-padding">

        <!-- Beschriftung für das Guthaben, wird von einem Label
             ↳ angezeigt -->
        Guthaben:

        <!-- ASP.NET Label zur Anzeige des aktuellen Guthabens -->
        <asp:Label
            ID="Label_guthaben"
            runat="server">
        </asp:Label>

        <!-- LinkButton zum Aktualisieren des Guthabens, rechts im
             ↳ Footer positioniert -->
        <asp:LinkButton
            ID="LinkButton_aktualisieren"
            runat="server"
            CssClass="w3-right"
            CausesValidation="False"

```

```

        Text="Aktualisieren"
        Style="text-decoration: none;">
    </asp:LinkButton>

</div> <!-- Ende des w3-content Containers für den Footer -->

</div> <!-- Ende des w3-bottom Footers -->

```

4.4.4 button_kamera_starten

```

<!-- Schwebender Button "Kamera starten".
      Der Button startet die Kamera und bleibt fest auf der Seite
      ↪ sichtbar. -->
<button
  id="button_kamera_starten"
  onclick="kamera_starten();return false;"
  class="
    w3-button
    w3-round-xlarge
    w3-large
    w3-card
    w3-teal
    fixed-button">
  <!-- Pluszeichen als Inhalt des Buttons -->
  +
</button>

```

4.5 Eigentlicher Seiteninhalt

```

<!-- Kommentar: Beginn des eigentlichen Seiteninhalts -->
<div class="w3-content"> <!-- Container, der den Hauptinhalt der Seite
  ↪ umschließt, mit einer vordefinierten Breite und Ausrichtung durch
  ↪ w3.css -->
:
:
:
</div>

```

4.5.1 Panel lesezeichen_erstellen

```

<!-- Panel für das Erstellen eines Lesezeichens, standardmäßig
↳ unsichtbar (Visible="False") -->
<asp:Panel
  ID="Panel lesezeichen erstellen"
  runat="server"
  CssClass="w3-card w3-round w3-margin"
  Visible="False">

  <!-- Container div für den Inhalt des Panels -->
  <div class="w3-container">

    <!-- Überschrift für den Abschnitt "Lesezeichen erstellen" -->
    <h3>Lesezeichen erstellen</h3>

    <!-- Erläuterung für den Nutzer, dass ein Cookie gesetzt wurde,
         um den direkten Zugang zum Projekt zu erleichtern.
         ↳ Empfehlung, den Link zu speichern. -->
    Partitio hat einen Cookie gesetzt,
    sodass du mit partitio.de wieder direkt in dein Projekt
    ↳ gelangst.
    Trotzdem ist es sinnvoll
    (beispielsweise, wenn du ein anderes Gerät benutzen möchtest),
    den Link zu dieser Seite irgendwo zu speichern:

    <!-- Zeilenumbrüche für eine bessere Lesbarkeit des Textes -->
    <br />
    <br />

    <!-- Container für den Link zum Lesezeichen, der in blauer
         ↳ Schriftfarbe angezeigt wird -->
    <div class="w3-container w3-text-blue">
      <asp:Label
        ID="Label_link lesezeichen"
        runat="server"></asp:Label>
    </div>

    <!-- Erneuter Zeilenumbruch -->
    <br />

    <!-- Hinweis zum Erstellen eines Lesezeichens im Browser und
         ↳ zur sicheren Aufbewahrung des Links -->
    Beachte bitte:

    <ul>
      <li>Am besten erstellst du jetzt einfach ein Lesezeichen
        ↳ der aktuellen Seite in deinem Browser
        (beispielsweise mit <strong>Strg + D</strong> unter
        ↳ Windows).</li>

```



```

        <li>Ohne den Link kannst du nicht auf deine Objekte
        ↪ zugreifen.</li>
        <li>Wer den Link hat, kann in deinem Namen bieten. Verteile
        ↪ ihn also nicht.</li>
    </ul>

    <!-- Button zum Kopieren des Links in die Zwischenablage -->
    <asp:Button
        ID="Button_link lesezeichen"
        runat="server"
        Text="Link in Zwischenablage kopieren"
        OnClientClick="kopiere_in_die_zwischenablage('Label_link_le_
        ↪ sezeichen')"
        CssClass="w3-button w3-round w3-light-gray w3-border
        ↪ w3-border-gray w3-margin" />

    </div>
</asp:Panel>

```

4.5.2 Panel_neue_teilnehmerinnen

```

<!-- Panel für die Einladung neuer Teilnehmerinnen, standardmäßig
↪ unsichtbar (Visible="False") -->
<asp:Panel
    ID="Panel_neue_teilnehmerinnen"
    runat="server"
    CssClass="w3-card w3-round w3-margin"
    Visible="False">

    <!-- Container div für den Inhalt des Panels -->
    <div class="w3-container">

        <!-- Überschrift für den Abschnitt "Neue Teilnehmerinnen" -->
        <h3>Neue Teilnehmerinnen</h3>

        <!-- Erklärung für den Nutzer, wie neue Teilnehmerinnen
        ↪ eingeladen werden können -->
        Um neue Teilnehmerinnen zu deinem Partitio-Projekt einzuladen,
        sendest du ihnen einfach den folgenden Link per
        Whatsapp, Signal, Telegram, SMS, ... oder

        <!-- Label für die Anzeige der E-Mail-Adresse der neuen
        ↪ Teilnehmerinnen -->
        <asp:Label
            ID="Label_email_neue_teilnehmer"

```

```
        runat="server">
</asp:Label>:

<!-- Zeilenumbrüche für bessere Lesbarkeit -->
<br />
<br />

<!-- Container für den Link zum Einladen neuer Teilnehmerinnen,
     ↪ in blauer Schrift -->
<div class="w3-container w3-text-blue">
    <asp:Label
        ID="Label_link_neue_teilnehmer"
        runat="server">
    </asp:Label>
</div>

<!-- Erneuter Zeilenumbruch -->
<br />

<!-- Hinweise für den Nutzer zur Weitergabe des Links und den
     ↪ möglichen Problemen -->
Beachte bitte:

<ul>
    <li>Wer den Link hat, kann an dem Projekt teilnehmen.</li>
    <li>Viele Menschen scheuen sich sinnvollerweise,
        auf Links in Nachrichten zu klicken.
        Eine E-Mail könnte im Spam-Ordner landen.
        Vielleicht möchtest du zusätzlich anrufen?</li>
</ul>

<!-- Button zum Kopieren des Einladungslinks in die
     ↪ Zwischenablage -->
<asp:Button
    ID="Button_link_neue_teilnehmer"
    runat="server"
    Text="Link in Zwischenablage kopieren"
    OnClientClick="kopiere_in_die_zwischenablage('Label_link_ne_
     ↪ ue_teilnehmer')"
    CssClass="w3-button w3-round w3-light-gray w3-border
     ↪ w3-border-gray w3-margin" />

</div>
</asp:Panel>
```

4.5.3 Panel_neue_teilnehmerin

```

<!-- Panel für die Eingabe einer neuen Teilnehmerin, standardmäßig
↳ unsichtbar (Visible="False") -->
<asp:Panel
  ID="Panel_neue_teilnehmerin"
  runat="server"
  CssClass="w3-card w3-round w3-margin"
  DefaultButton="Button_neue_teilnehmerin_eintragen" <!-- Bestimmt den
↳ Standardbutton bei Enter -->
  Visible="false">

  <!-- Container div für den Inhalt des Panels -->
  <div class="w3-container">

    <!-- Überschrift für den Abschnitt "Neue Teilnehmerin" -->
    <h3>Neue Teilnehmerin</h3>

    <!-- Eingabeformular für den Namen der neuen Teilnehmerin -->
    <div class="w3-card w3-round w3-margin-top w3-margin-bottom">

      <!-- Label für das Namensfeld -->
      <span class="w3-margin">Name</span>

      <!-- TextBox für die Eingabe des Namens der neuen
↳ Teilnehmerin -->
      <asp:TextBox
        ID="TextBox_name"
        CssClass="w3-margin"
        runat="server"
        onfocus="this.select()" <!-- Markiert den aktuellen
↳ Text, wenn das Feld fokussiert wird -->
        Width="100"
        MaxLength="10">
      </asp:TextBox>

      <!-- Validator für das Namensfeld: Name darf nicht leer
↳ sein -->
      <asp:RequiredFieldValidator
        ID="RequiredFieldValidator_name"
        runat="server"
        ErrorMessage="<div class='w3-container'>Der Name darf
↳ nicht leer sein.</div>"
        Display="Dynamic"
        ControlToValidate="TextBox_name">
      </asp:RequiredFieldValidator>

      <!-- CustomValidator, um zu prüfen, ob der Name bereits
↳ verwendet wird -->

```

```
<asp:CustomValidator
  ID="CustomValidator_name"
  runat="server"
  ErrorMessage="<div class='w3-container'>Der Name wird
  ↪ schon genutzt (s.u.).</div>"
  Display="Dynamic"
  ControlToValidate="TextBox_name">
</asp:CustomValidator>

<!-- Zeilenumbruch für eine saubere Trennung zwischen den
  ↪ Elementen -->
<br />

<!-- Button zum Eintragen der neuen Teilnehmerin -->
<asp:Button
  ID="Button_neue_teilnehmerin_eintragen"
  runat="server"
  CssClass="w3-button w3-round w3-light-gray w3-border
  ↪ w3-border-gray w3-margin"
  Text="OK" />

</div>

<!-- Panel für die Anzeige aller bisherigen Teilnehmerinnen -->
<asp:Panel
  ID="Panel_alle_nutzer"
  runat="server"
  Visible="False"> <!-- Standardmäßig unsichtbar -->

  <!-- Überschrift für die Liste der bisherigen
  ↪ Teilnehmerinnen -->
  <h3>Bisherige Teilnehmerinnen</h3>

  <!-- GridView zur Anzeige der Teilnehmerinnen, ohne
  ↪ sichtbare Gitternetzlinien und ohne Kopfzeile -->
  <asp:GridView
    ID="GridView_nutzer"
    runat="server"
    GridLines="None"
    ShowHeader="False"
    CssClass="w3-table w3-bordered">
  </asp:GridView>

</asp:Panel>

</div>
```

```
</asp:Panel>
```

4.5.4 Panel_loeschen

```
<!-- Panel für die Bestätigung einer Löschaktion, standardmäßig
↳ unsichtbar (Visible="False") -->
<asp:Panel
  ID="Panel_loeschen"
  runat="server"
  CssClass="w3-card w3-round w3-margin"
  Visible="False">

  <!-- Container div für den Inhalt des Panels -->
  <div class="w3-container">

    <!-- Überschrift für den Abschnitt "Löschbestätigung" -->
    <h3>Löschbestätigung</h3>

    <!-- Label, um Informationen zum Löschvorgang anzuzeigen
    ↳ (dynamischer Text) -->
    <asp:Label
      ID="Label_loeschen"
      runat="server"></asp:Label>

    <!-- Zeilenumbruch für Abstand zwischen Label und Buttons -->
    <br />

    <!-- Button zum Abbrechen des Löschvorgangs -->
    <asp:Button
      ID="Button_loeschen_abbrechen"
      runat="server"
      Text="Abbrechen"
      CssClass="w3-button w3-round w3-light-gray w3-border
        ↳ w3-border-gray w3-margin-top w3-margin-bottom" />

    <!-- Button zur Bestätigung des Löschvorgangs -->
    <asp:Button
      ID="Button_loeschen_ok"
      runat="server"
      Text="Löschen"
      CssClass="w3-button w3-round w3-pale-red w3-border
        ↳ w3-border-gray w3-right w3-margin-top w3-margin-bottom"
        ↳ />

  </div> <!-- Ende des Containers -->
```

```
</asp:Panel> <!-- Ende des Panels -->
```

4.5.5 Panel_zufriedenheit

```
<!-- Panel zur Abfrage der Zufriedenheit der Teilnehmerinnen,
↳ standardmäßig unsichtbar (Visible="False") -->
<asp:Panel
  runat="server"
  ID="Panel_zufriedenheit"
  CssClass="w3-card w3-round w3-margin"
  Visible="False">

  <!-- Container div für den Inhalt des Panels -->
  <div class="w3-container">

    <!-- Überschrift für die Zufriedenheitsabfrage -->
    <h3>Alle zufrieden?</h3>

    <!-- Erklärung: Wenn alle Teilnehmerinnen zufrieden sind, ist
    ↳ das Projekt abgeschlossen -->
    Wenn alle Teilnehmerinnen mit dem aktuellen Zustand des
    ↳ Projekts zufrieden sind,
    ist das Projekt abgeschlossen.

    <!-- Zeilenumbrüche für Abstand -->
    <br />
    <br />

    <!-- GridView zur Anzeige der Zufriedenheit der Teilnehmerinnen
    ↳ -->
    <asp:GridView
      ID="GridView_zufriedenheit"
      runat="server"
      CssClass="w3-table w3-bordered"
      GridLines="None"
      ShowHeader="False"
      AutoGenerateColumns="False">

      <Columns>

        <!-- Spalte für den Namen der Teilnehmer -->
        <asp:BoundField DataField="Nutzer"></asp:BoundField>

        <!-- Spalte für die Angabe der Zufriedenheit -->
        <asp:BoundField
          ↳ DataField="Zufriedenheit"></asp:BoundField>
```

```
        </Columns>

    </asp:GridView>

    <!-- Button, um anzugeben, dass man zufrieden ist -->
    <asp:Button
        ID="Button_zufrieden"
        runat="server"
        CssClass="w3-button w3-round w3-light-gray w3-border
        ↪ w3-border-gray w3-margin"
        Text="Zufrieden" />

    <!-- Button, um anzugeben, dass man unzufrieden ist -->
    <asp:Button
        ID="Button_unzufrieden"
        runat="server"
        CssClass="w3-button w3-round w3-light-gray w3-border
        ↪ w3-border-gray w3-margin"
        Text="Unzufrieden" />

</div>

</asp:Panel>
```

4.5.6 foto_session

```
<!-- Div für die Foto-Session, standardmäßig unsichtbar (display: none)
↪ -->
<div
    id="foto_session"
    style="display: none">

    <!-- Zeilenumbrüche, damit die Foto-Session nicht unter dem Header
    ↪ verschwindet -->
    <br />
    <br />

    <!-- Container div für das Foto-Session-Panel mit Kartenstil und
    ↪ runden Ecken -->
    <div class="w3-card w3-round w3-margin">

        <!-- Container für das Video-Element -->
        <div id="videoContainer">

            <!-- Video-Element zum Anzeigen des Kamerabildes -->
```

```
<video
  id="videoElement"
  class="w3-round w3-margin-top"
  autoplay="autoplay"
  playsinline="playsinline">
</video>

</div>

<!-- Unsichtbare Zeichenfläche (Canvas) für die Aufnahme des
↳ Bildes -->
<canvas
  id="canvas"
  height="300"
  width="300"
  style="display: none;"></canvas>

<!-- Button zum Abbrechen der Foto-Session, ruft die
↳ JavaScript-Funktion "kamera_stoppen" auf -->
<asp:Button
  runat="server"
  CssClass="w3-button w3-round w3-light-gray w3-border
↳ w3-border-gray w3-margin"
  ID="abbruch"
  Text="Abbrechen"
  OnClientClick="kamera_stoppen(); return
↳ false;"></asp:Button>

<!-- Button zum Aufnehmen eines Bildes, ruft die
↳ JavaScript-Funktion "bild_aufnehmen" auf -->
<asp:Button
  runat="server"
  CssClass="w3-button w3-round w3-light-gray w3-border
↳ w3-border-gray w3-right w3-margin"
  ID="bild_aufnahme"
  Text="Bild aufnehmen"
  OnClientClick="bild_aufnehmen(); return
↳ false;"></asp:Button>

</div>

</div>
```


4.5.7 Panel_detail

```
<!-- Panel zur Anzeige und Bearbeitung der Details eines Objekts,
↳ standardmäßig unsichtbar (Visible="False") -->
<asp:Panel
  ID="Panel_detail"
  runat="server"
  Visible="False"
  class="w3-card w3-round w3-margin"
  DefaultButton="Button_speichern">

  <!-- Bild des ausgewählten Objekts -->
  <asp:Image
    ID="Image_objekt"
    runat="server"
    Height="200px"
    Width="200px"
    CssClass="w3-round-large w3-margin" />

  <!-- LinkButton zum Abbrechen der Detailansicht -->
  <asp:LinkButton
    ID="LinkButton_abbrechen"
    runat="server"
    CausesValidation="False"
    Text="<i class='material-icons'>close</i>"
    CssClass="w3-button w3-right" />

  <!-- Zeilenumbruch -->
  <br />

  <!-- Textbox zur Eingabe oder Anzeige des Objektnamens -->
  <asp:TextBox
    ID="Textbox_objekt_name"
    runat="server"
    CssClass="w3-margin-left"
    onfocus="this.select()"
    Width="200px">
  </asp:TextBox>

  <!-- Besitzer-Informationen zum Objekt -->
  <div class="w3-margin-left w3-margin-top">
    Besitzer:

    <!-- Label zur Anzeige des Besitzers des Objekts -->
    <asp:Label
      ID="Label_besitzer"
      runat="server">
    </asp:Label>
  </div>
</asp:Panel>
```

```
</div>

<!-- Gebotsinformationen für das Objekt -->
<div class="w3-margin-left w3-margin-top">
  Gebot:
  <!-- Textbox zur Eingabe des Gebots -->
  <asp:TextBox
    ID="Textbox_objekt_gebot"
    runat="server"
    MaxLength="3"
    onfocus="this.select()"
    Width="50px">
  </asp:TextBox>

  <!-- RangeValidator, um das Gebot auf einen bestimmten
  ↳ Wertebereich zu beschränken -->
  <asp:RangeValidator
    ID="RangeValidator_gebot"
    runat="server"
    ControlToValidate="Textbox_objekt_gebot"
    MinimumValue="0"
    MaximumValue="999"
    Display="Dynamic"
    Type="Integer"
    CssClass="w3-text-red">
  </asp:RangeValidator>

  <!-- RequiredFieldValidator, um sicherzustellen, dass ein Gebot
  ↳ eingegeben wird -->
  <asp:RequiredFieldValidator
    ID="RequiredFieldValidator_gebot"
    runat="server"
    ControlToValidate="Textbox_objekt_gebot"
    Display="Dynamic"
    CssClass="w3-text-red">
  </asp:RequiredFieldValidator>

  <!-- Zeilenumbruch -->
  <br />

</div>

<!-- Button zum Speichern der Änderungen -->
<asp:Button
  ID="Button_speichern"
  runat="server"
  Text="Bestätigen"
```

```

        CssClass="w3-button w3-round w3-light-gray w3-border
        ↪ w3-border-gray w3-margin" />

<!-- Button zum Löschen des Objekts -->
<asp:Button
    ID="Button_loeschen"
    runat="server"
    Text="Löschen"
    CssClass="w3-button w3-pale-red w3-round w3-border
    ↪ w3-border-gray w3-right w3-margin" />

</asp:Panel>

```

4.5.8 GridView_objekte

```

<!-- GridView zur Anzeige von Objekten, mit sortierbaren Spalten und
    ↪ DataKeyNames zur Identifizierung der Objekte -->
<asp:GridView
    ID="GridView_objekte"
    runat="server"
    DataKeyNames="ID"
    AutoGenerateColumns="False"
    CssClass="w3-table w3-bordered"
    GridLines="None"
    AllowSorting="True"
    OnSorting="GridView_objekte_Sorting">

    <Columns>

        <!-- TemplateField für die Anzeige eines Bildes, das jedes
            ↪ Objekt darstellt -->
        <asp:TemplateField>
            <!-- HeaderTemplate definiert den Inhalt im Tabellenkopf
                ↪ -->
            <HeaderTemplate>
                <!-- Kein expliziter Header, da nur Bilder dargestellt
                    ↪ werden -->
            </HeaderTemplate>
            <!-- ItemTemplate zeigt in jeder Zeile ein Bild, das mit
                ↪ dem Objekt verknüpft ist -->
            <ItemTemplate>
                <asp:ImageButton
                    CssClass="w3-round"
                    ID="Button_detail"
                    runat="server"
                    CausesValidation="False"

```

```
        ImageUrl='<%# Eval("Bild") %>'
        CommandName="Select"
        Style="text-decoration: none;"
        Height="40px"
        Width="40px" />
    </ItemTemplate>
</asp:TemplateField>

<!-- TemplateField für die Anzeige und Sortierung nach
     ↳ Objektnamen -->
<asp:TemplateField>
    <HeaderTemplate>
        <!-- LinkButton im Header, der das Sortieren nach
             ↳ Objektnamen ermöglicht -->
        <asp:LinkButton
            ID="LinkButton_sort_objekt"
            runat="server"
            CommandName="Sort"
            CommandArgument="Objekt"
            Text="Objekt"
            Style="text-decoration: none;" />
    </HeaderTemplate>
    <ItemTemplate>
        <!-- LinkButton zur Anzeige des Objektnamens in der
             ↳ jeweiligen Zeile -->
        <asp:LinkButton
            ID="LinkButton_objekt"
            runat="server"
            Text='<%# Eval("Objekt") %>'
            Style="text-decoration: none;"
            CausesValidation="False"
            CommandName="Select"
            CommandArgument='<%# Eval("ID") %>' />
    </ItemTemplate>
</asp:TemplateField>

<!-- TemplateField für die Anzeige und Sortierung nach dem
     ↳ Gebot -->
<asp:TemplateField>
    <HeaderTemplate>
        <!-- LinkButton im Header zur Sortierung nach Gebot -->
        <asp:LinkButton
            ID="LinkButton_sort_gebot"
            runat="server"
            CommandName="Sort"
            CommandArgument="Gebot"
            Text="Gebot"
```

```

        Style="text-decoration: none;" />
</HeaderTemplate>
<HeaderStyle CssClass="text-right" />
<ItemTemplate>
    <!-- LinkButton zur Anzeige des Gebots in der
    ↪ jeweiligen Zeile -->
    <asp:LinkButton
        ID="LinkButton_gebot"
        runat="server"
        Text='<%=# Eval("Gebot") %>'
        Style="text-decoration: none;"
        CausesValidation="False"
        CommandName="Select"
        CommandArgument='<%=# Eval("ID") %>' />
    </ItemTemplate>
    <ItemStyle CssClass="text-right" />
</asp:TemplateField>

<!-- TemplateField für die Anzeige und Sortierung nach dem
    ↪ Besitzer -->
<asp:TemplateField>
    <HeaderTemplate>
        <!-- LinkButton im Header zur Sortierung nach Besitzer
        ↪ -->
        <asp:LinkButton
            ID="LinkButton_sort_besitzer"
            runat="server"
            CommandName="Sort"
            CommandArgument="Besitzer"
            Text="Besitzer"
            Style="text-decoration: none;" />
    </HeaderTemplate>
    <ItemTemplate>
        <!-- LinkButton zur Anzeige des Besitzers in der
        ↪ jeweiligen Zeile -->
        <asp:LinkButton
            ID="LinkButton_besitzer"
            runat="server"
            Text='<%=# Eval("Besitzer") %>'
            Style="text-decoration: none;"
            CausesValidation="False"
            CommandName="Select"
            CommandArgument='<%=# Eval("ID") %>' />
    </ItemTemplate>
</asp:TemplateField>

</Columns>

```

```
</asp:GridView>
```

Kapitel 5

partitio.css

In dieser CSS-Datei wurden zusätzliche, spezifische Stilklassen erstellt, um Funktionalitäten zu ergänzen, die in w3.css [5] nicht standardmäßig enthalten sind. Die w3.css-Klassen wurden für allgemeine Formatierungen und Layouts verwendet, während hier definierte Klassen besondere Aufgaben übernehmen.

Zum Beispiel deckt die Klasse `#Panel_seitenleiste` den gesamten Bildschirm ab, um Benutzerinteraktionen mit Hintergrundelementen zu verhindern. Ebenso bietet `.automatische_breite` eine flexible Breitensteuerung für Inhalte, und `.fixed-button` ermöglicht es, Buttons stets unten rechts auf dem Bildschirm zu positionieren. Für Video-Container und spezielle Textausrichtungen wurden weitere Anpassungen vorgenommen, die in Kombination mit w3.css ein gezieltes Design und Verhalten ermöglichen.

```
/*
#Panel_seitenleiste: Ein Panel, das die gesamte Breite und Höhe des
↳ Bildschirms abdeckt,
um Benutzerinteraktionen auf darunterliegenden Objekten zu verhindern.
*/
#Panel_seitenleiste {
    width: 100%;
    height: 100%;
    z-index: 1000 !important;
}

/*
.automatische_breite: Stellt sicher, dass das Element die Breite seines
↳ Inhalts annimmt,
anstatt die gesamte verfügbare Breite zu nutzen.
*/
.automatische_breite {
    display: inline-block;
    width: auto;
}

/*
.text-left: Richtet den Text linksbündig aus und erzwingt dies.
```

```
*/
.text-left {
    text-align: left !important;
}

/*
.text-right: Richtet den Text rechtsbündig aus und erzwingt dies.
*/
.text-right {
    text-align: right !important;
}

/*
#videoContainer: Ein Container für Videoelemente mit festen Abmessungen
→ und horizontaler Zentrierung.
*/
#videoContainer {
    width: 300px;
    height: 300px;
    overflow: hidden;
    position: relative;
    margin: 0 auto; /* Zentriert den Container horizontal */
}

/*
#videoElement: Passt das Video dem Container an und schneidet es bei
→ Bedarf zurecht.
*/
#videoElement {
    width: 100%;
    height: 100%;
    object-fit: cover;
}

/*
.fixed-button: Ein fester Button, der sich unten rechts auf dem
→ Bildschirm befindet.
*/
.fixed-button {
    position: fixed;
    bottom: 60px;
    right: 20px;
    z-index: 999;
}

/*
.w3-table td: Erzwingt eine mittlere vertikale Ausrichtung von
→ Tabellenzellen.
```



```
w3-table setzt standardmäßig vertical-align: top.
*/
.w3-table td {
  vertical-align: middle !important; /* w3-table setzt defaultmäßig
  ↳ vertical-align: top*/
}

/*
.monospace-font: Setzt eine Schriftart mit fester Breite, ideal für
↳ codeähnliche Darstellungen.
*/
.monospace-font {
  font-family: "Courier New", Courier, monospace;
}

/*
.halbtransparent: Setzt einen halbtransparenten schwarzen Hintergrund
↳ mit 50% Deckkraft.
*/
.halbtransparent {
  background-color: rgba(0, 0, 0, 0.5) !important; /* Schwarz mit 50%
  ↳ Transparenz */
}
```

Kapitel 6

`bild_empfangen.ashx`

Der HTTP-Handler `bild_empfangen` verarbeitet Bild-Uploads von Nutzern und fügt neue Objektinformationen in eine XML-Datei für ein Projekt ein. Dieser Handler funktioniert in einer ASP.NET-Anwendung und übernimmt folgende Schritte:

Datei-Upload-Verarbeitung: Die Methode `ProcessRequest` prüft, ob eine Datei unter dem Schlüssel "bild" im Anfrageobjekt hochgeladen wurde. Falls eine gültige Datei vorhanden ist, wird sie in einem bestimmten Verzeichnis (`bilder/`) auf dem Server gespeichert.

Antwort an den Client: Der Handler sendet eine JSON-Antwort zurück, die entweder den erfolgreichen Upload bestätigt oder im Fehlerfall darauf hinweist, dass keine Datei hochgeladen wurde.

Objekterstellung in Projekt-XML: Mithilfe der Methode `Neues_objekt` wird das hochgeladene Bild als neues Objekt in einer XML-Datei für das Projekt registriert. Das Objekt wird mit einer eindeutigen Objekt-ID, einer Nutzer-ID (Besitzer) und einem anfänglichen Gebotswert in der XML-Datei gespeichert.

IsReusable-Eigenschaft: Diese Eigenschaft ist auf `False` gesetzt, was bedeutet, dass bei jeder neuen Anfrage ein neuer Handler-Instanz erzeugt wird, um eine saubere Verarbeitung zu gewährleisten.

Dieser Handler ist speziell dafür konzipiert, Bilddateien von Nutzern zu empfangen, auf dem Server zu speichern und die zugehörigen Objektinformationen für eine spätere Nutzung in einer Projekt-XML zu aktualisieren.

```
<%@ WebHandler Language="VB" Class="bild_empfangen" %> ' Deklariert
↳ einen HTTP-Handler mit der Klasse "bild_empfangen" in VB

Imports System ' Importiert grundlegende .NET-Bibliotheken
Imports System.Web ' Importiert Funktionen zur Webverarbeitung

' Definiert die Klasse "bild_empfangen" und implementiert das
↳ "IHttpHandler"-Interface
Public Class bild_empfangen : Implements IHttpHandler

    ' Hauptmethode zur Verarbeitung eingehender HTTP-Anfragen
```

```

Public Sub ProcessRequest(ByVal context As HttpContext) Implements
  IHttpHandler.ProcessRequest

  ' Zugriff auf die eingehende HTTP-Anfrage
  Dim request = context.Request

  ' Extrahiert die hochgeladene Datei mit dem Schlüssel "bild"
  ' → aus dem Anfrageobjekt
  Dim file = request.Files("bild")
  ' Erfasst die Projekt-ID aus den Formulardaten der Anfrage
  Dim projekt_id As String = request.Form("projekt_id")
  ' Erfasst die Nutzer-ID aus den Formulardaten der Anfrage
  Dim nutzer_id As String = request.Form("nutzer_id")
  ' Erfasst die Objekt-ID des Bildes aus den Formulardaten der
  ' → Anfrage
  Dim objekt_id As String = request.Form("bild_id")

  ' Prüft, ob eine Datei hochgeladen wurde und ob die Dateigröße
  ' → größer als 0 ist
  If file IsNot Nothing AndAlso file.ContentLength > 0 Then

    ' Ermittelt den Serverpfad, unter dem die Datei gespeichert
    ' → wird
    Dim dateiname = context.Server.MapPath("bilder/" &
      ' → file.FileName)

    ' Speichert die Datei am bestimmten Speicherort
    file.SaveAs(dateiname)

    ' Sendet eine JSON-Antwort, die den erfolgreichen Upload
    ' → bestätigt
    context.Response.Write("{\"status\": \"Erfolg\"}")
    ' context.Response.Write("{\"status\": \"\" & nutzer_id &
    ' → \"\"}") ' (Optionaler Code zum Debuggen, derzeit
    ' → auskommentiert)

  Else

    ' Sendet eine JSON-Antwort mit einer Fehlermeldung, falls
    ' → keine Datei hochgeladen wurde
    context.Response.Write("{\"status\": \"Fehler\",
      ' → \"message\": \"Keine Datei hochgeladen.\"}")
  End If

  ' Setzt den Content-Type der Antwort auf "application/json" für
  ' → JSON-Daten
  context.Response.ContentType = "application/json"

```

```

    ' Erstellt ein neues Objekt im Projekt mithilfe der gegebenen
    ↪ IDs und des Kontextes
    Neues_objekt(projekt_id, nutzer_id, objekt_id, context)

End Sub

' Legt fest, dass der Handler nicht wiederverwendbar ist
Public ReadOnly Property IsReusable() As Boolean Implements
    ↪ IHttpHandler.IsReusable
    Get
        Return False ' Gibt an, dass für jede Anfrage ein neuer
        ↪ Handler erstellt wird
    End Get
End Property

' Methode zum Hinzufügen eines neuen Objekts in der
↪ Projekt-XML-Datei
Private Sub Neues_objekt(
    ByVal projekt_id As String,
    ByVal nutzer_id As String,
    ByVal objekt_id As String,
    ByVal context As HttpContext)

    ' Ermittelt den Pfad zur Projekt-XML-Datei basierend auf der
    ↪ Projekt-ID
    Dim projekt_xml_datei =
        context.Server.MapPath(
            "projekte/" &
            projekt_id &
            ".xml")

    ' Lädt die Projekt-XML-Datei zur Bearbeitung
    Dim projekt_xml = XElement.Load(projekt_xml_datei)

    ' Greift auf den Abschnitt "objekte" innerhalb der Projektdatei
    ↪ zu
    Dim alle_objekte As XElement = projekt_xml.Element("objekte")

    ' Definiert ein neues Objekt-Element mit ID, Name, Besitzer-ID
    ↪ und Gebot
    Dim neues_objekt As XElement =
        <objekt>
            <objekt_id><%= objekt_id %></objekt_id>
            <objekt_name>...</objekt_name>
            <besitzer_id><%= nutzer_id %></besitzer_id>
            <objekt_gebot>0</objekt_gebot>
        </objekt>

```

```
    ' Fügt das neue Objekt der Sammlung aller Objekte in der  
    ↪ Projektdatei hinzu  
alle_objekte.Add(neues_objekt)  
  
    ' Speichert die Änderungen an der Projekt-XML-Datei  
projekt_xml.Save(projekt_xml_datei)  
  
End Sub  
  
End Class
```

Literaturverzeichnis

- [1] Wikipedia. (2024) Kō. [Online]. Available: <https://de.wikipedia.org/wiki/K%C5%8D>
- [2] OpenAI. (2024) Chatgpt. [Online]. Available: <https://chatgpt.com>
- [3] Github Copilot. (2024). [Online]. Available: <https://github.com/features/copilot>
- [4] Stack Overflow. (2024). [Online]. Available: <https://stackoverflow.com/>
- [5] W3Schools. (2024) W3.css. [Online]. Available: <https://www.w3schools.com/w3css>
- [6] J. J. Buchholz. (2024) Partitio. Hochschule Bremen. [Online]. Available: <https://m-server.fk5.hs-bremen.de/partitio/partitio.html>
- [7] Wikipedia. (2024) Asp.net. [Online]. Available: <https://de.wikipedia.org/wiki/ASP.NET>