Oldenburgische
Industrie- und Handelskammer

Erklärung

Bestätigung über die durchgeführte betriebliche Aufgabe[1]

(Diese Bestätigung ist als Deckblatt online einzureichen, gemeinsam mit dem Report/der Dokumentation.)

Prüfling (vollständige Anschrift und Telefonnummer)	**Ausbildungsbetrieb (vollständige Anschrift)**
Vorname, Name	Firma
Straße, Hausnr.	Straße, Hausnr.
PLZ, Ort	PLZ, Ort
Tel.Nr.:	Tel.Nr.:

Hinweis vorab: Aus Gründen der besseren Lesbarkeit wird auf die gleichzeitige Verwendung männlicher und weiblicher Sprachformen verzichtet. Sämtliche Personenbezeichnungen gelten gleichermaßen für alle Geschlechter.

Ausbildungsberuf

Bezeichnung der betrieblichen Aufgabe

Erklärung des Prüflings

Hiermit versichere ich, dass ich die betriebliche Aufgabe unter der Betreuung von

Verantwortlicher im Unternehmen

selbstständig durchgeführt und die Unterlagen selbstständig zusammengestellt habe.

Dokumente und Textpassagen, die ich nicht selbstständig erstellt habe, sind von mir gekennzeichnet.

Ort, Datum	Unterschrift des Prüflings

Bestätigung des Ausbildungsbetriebes

Wir bestätigen, dass die Angaben des Prüflings richtig sind.

Ort, Datum	Unterschrift des Verantwortlichen, der die Aufgabe betreut hat.

Ort, Datum	Unterschrift des Ausbilders

[1] Zur Vereinfachung wird einheitlich der Begriff „betriebliche Aufgabe“ verwendet. Gemeint sind die Fachaufgabe/die Projektarbeit/der betrieblicher Auftrag. Die unterschiedlichen Bezeichnungen entstehen durch die verschiedenen Berufe, die eine Aufgabe online einstellen.

Abschlussprüfung Winter 2025/2026

Fachinformatiker / Anwendungsentwicklung

Projektdokumentation

Empfehlungsmarketing-App

Webanwendung zur Empfehlungslinkerstellung und Gutscheingenerierung

Abgabetermin: Lübeck, den 10.12.2025

Prüfungsbewerber:
Benjamin Blunk
Auf dem Schild 6
23560 Lübeck

Ausbildungsbetrieb:
ananas.codes e.K.
Lindenstraße 8
23558 Lübeck

Inhaltsverzeichnis

Abkürzungsverzeichnis

AES	Advanced Encryption Standard
Cloud	Web-Server
CSRF	Cross-Site Request Forgery
CSS	Cascading Style Sheets
DOM	Document Object Model
DSGVO	Datenschutzgrundverordnung
FTP	File Transfer Protocol
Git	verteiltes Versionsverwaltungssystem
GitHub	Plattform für Git-Repositories
GUI	Graphical User Interface
HTML	Hypertext Markup Language
PDF	Portable Document Format
PDO	PHP Data Object
PHP	Hypertext Preprocessor
QR	Quick Response
SSL/TLS	Secure Sockets Layer/Transport Layer Security
SQL	Structured Query Language
URL	Uniform Resource Locator
VS Code	Visual Studio Code - Code-Editor von Microsoft
ZXing	Ausgesprochen: „Zebra Crossing“. Open-Source-Bibliothek zur Erkennung und in einigen Implementierungen auch zur Erstellung von Barcodes und QR-Codes.

1 Einleitung

1.1 Begriffsklärung

Zur Gewährleistung einer konsistenten und verständlichen Verwendung von Begriffen innerhalb dieser Projektdokumentation werden folgende Festlegungen getroffen:

- Der **Kunde**, der seinen Empfehlungslink an Interessenten weiter gibt, wird im weiteren Verlauf als **Werber** bezeichnet.
- Der **Interessent** wird im weiteren Verlauf als der **Geworbene** bezeichnet.

Durch diese Vereinheitlichung soll die Lesbarkeit und Nachvollziehbarkeit der Dokumentation verbessert werden.

1.2 Projektumfeld

- Mein Praktikumsbetrieb ist die Firma ananas.codes e.K., welche im Bereich Webentwicklung und Zeiterfassungssysteme tätig ist. Die Mitarbeiteranzahl beträgt aktuell fünf.
- Der Auftraggeber ist die Colibri Contactlinsen und Brillen GmbH, welche aktuell eine Filiale in Lübeck hat und 35 Mitarbeiter beschäftigt.

1.3 Projektziel

- Es geht um die Entwicklung einer Empfehlungsmarketing-Webanwendung.
- Das Ziel des Projekts war die Konzeption, Entwicklung und Einführung einer webbasierten Empfehlungsmarketing-Applikation für einen realen Kundenbetrieb. Die Anwendung soll es ermöglichen:
  - Kunden über einen Registrierungsprozess in ein Empfehlungsprogramm aufzunehmen.
  - Empfehlungslinks inklusive QR-Codes zu generieren und per E-Mail zu versenden.
  - Gutscheine automatisch auszustellen und Einlösungen im Mitarbeiterbereich vorzunehmen.
  - Den gesamten Prozess sicher, DSGVO konform und plattformunabhängig abzuwickeln.

  Die Entwicklung erfolgte als eigenständiges Projekt im Rahmen der Abschlussprüfung, orientiert an den Phasen des klassischen Projektmanagements (Wasserfallmodell) mit Prozessorientierung.

1.4 Projektbegründung

- Gewinnung von Neukunden durch Verteilung von Affiliate-Links durch Werber an Geworbene.
- Der Auftraggeber möchte über solch eine moderne Form der Neukundengewinnung verfügen.

1.5 Projektschnittstellen

Die Anwendung besteht aus einem webbasierten Frontend (HTML, CSS, JavaScript, Bootstrap) und einem serverseitigen Backend (PHP, Composer, MySQL). Die Kommunikation zwischen Frontend und Backend erfolgt über die `Fetch-API` mittels `HTTPS`-Requests. Daten werden dabei im `JSON`-Format ausgetauscht.

- **Frontend ⟶ Backend:**
  - → Über `fetch()`-Aufrufe werden REST-ähnliche Endpunkte in `api.php` angesprochen.
  - → Scanner-Funktion: Im Mitarbeiterbereich wird ein QR-Code-Scanner (`@zxing/browser` Bibliothek) genutzt, um Gutscheincodes zu erfassen.
- **Backend ⟶ Datenbank / interne Systeme:**
  - → Speicherung und Validierung von Benutzer- und Gutscheindaten in einer Datenbank.
  - → Prüfung und Generierung eindeutiger Referral-Codes ohne Kollision.
  - → Hashing und Fingerprinting von E-Mail-Adressen für Datenschutz.
  - → Versand von E-Mails (inklusive QR-Code-Anhang) über ein internes Mailer-System.
- **Genehmigung:**
  - → Das Projekt wird vom Chef meines Praktikumsbetriebs genehmigt.
- **Benutzer der Anwendung:**
  - → Werber, die sich registrieren und ihren Empfehlungslink teilen, um Gutscheine zu erhalten.
  - → Geworbene, die sich registrieren um ihren Empfehlungslink und einen Gutschein zu erhalten.
  - → Mitarbeiter, die Gutscheincodes scannen und einlösen, wenn diese erfolgreich validiert wurden.
- **Präsentation der Ergebnisse:**

→ Die Projektergebnisse werden meinem Chef präsentiert.

1.6 Projektabgrenzung

- Meine Projektarbeit umfasst die Entwicklung des Frontend und Backend der Anwendung, sowie das Testen und die abschließende Abnahme durch meinen Chef. Der Rollout beim Kunden ist nicht Bestandteil meiner Projektarbeit.

2 Projektplanung

2.1 Projektphasen

Es wurde das Wasserfallmodell eingesetzt:

- **Analyse**
- **Entwurf**
- **Implementierung**
- **Test**
- **Wartung**

2.2 Abweichungen vom Projektantrag

- Detaillierte Projektbeschreibung
  - Die Angabe, dass der Geworbene nur einen Rabatt erhält wenn er eine Brille kauft ist nicht richtig. Es wird vom Auftraggeber noch entschieden, für welche Produkte eine Rabattierung möglich ist.
  - Dem Geworbenen wird ebenfalls ein Affiliate-Link zusätzlich zum Gutschein zugesandt, da dieser dann ebenfalls einen Anreiz hat diesen zu teilen um weitere Gutscheine zu erhalten.
  - Die Angabe, dass die Gutscheine ihre Gültigkeit nicht verlieren sollen beruht auf einem Missverständnis zwischen mir und meinem Chef. Die Gutscheine sollen nach der Einlösung entwertet werden.
  - Die Frontend-Library qr-scanner wird ersetzt durch ZXing für den QR-Code-Scan zum Einlösen von Gutscheinen, da diese Library häufig genutzt wird in der Softwareentwicklung und sehr zuverlässig und schnell QR-Codes erkennt.

- Projektphasen
  - Der angegebene geschätzte Zeitbedarf der einzelnen Phasen hat sich nun geändert, aufgrund meiner Neueinschätzung. Die Phase Wartung entfällt, da die Inbetriebnahme der Anwendung beim Kunden nicht Bestandteil meiner Projektarbeit sein soll (Vorgabe meines Praktikumsbetriebs).

2.3 Ressourcen und Werkzeuge

- **Frontend:** HTML5, CSS3, JavaScript, Bootstrap 5.3
- **Backend:** PHP 8.1, Composer, MySQL
- **Libraries:** vlucas/phpdotenv, PHPMailer (Mailversand), DOMPDF (PDF-Erzeugung), endroid/qr-code (QR-Codes erzeugen), ZXing
- **Datenbank:** MySQL (Cloud-gehostet)
- **Sicherheit:** PDO, Advanced Encryption Standard (AES)-256-GCM, HTTPS, CSRF-Tokens, Mitarbeiter-Login (PIN)
- **Sonstiges:** VS Code, Git/GitHub, FileZilla FTP-Client

2.4 Zeitplanung

- Analyse: 2 h
- Entwurf: 16 h
- Implementierung: 59 h
- Test/Abnahme: 3 h

3 Analysephase

3.1 Fachliches Konzept

Werber erzeugen personalisierte Empfehlungslinks (tokenisiert) inklusive QR-Codes für Geworbene. Geworbene registrieren sich über den Link. Das System erfasst die Email-Adresse und ordnet sie dem Werber zu. Nach dem Kauf bestimmter Produkte durch den Geworbenen, erhält dieser einen bestimmten Rabatt auf sein gekauftes Produkt, nachdem ein Mitarbeiter den QR-Code auf seinem Gutschein mit der App gescannt, validiert und entwertet hat. Dadurch wird automatisch ein Gutschein (PDF) erzeugt und per E-Mail an den Werber versendet.

3.2 Kosten (Beispiel)

- Entwicklungszeit (80 h) × interner Stundensatz (60 €/h) → 4 800 €,
- Hosting/Tools (monatlich), PDF-/Mail-Libraries: Open Source.

3.3 Nutzen

Erhöhung der Neukundengewinnung über Empfehlungen, transparente Nachverfolgung, minimale manuelle Aufwände durch Automatisierung.

3.4 Amortisation

Erwarteter zusätzlicher Deckungsbeitrag durch Neukunden übersteigt die initialen Entwicklungskosten nach n Monaten.

4 Entwurfsphase

4.1 Architektur

4.1.1 Schichtenmodell

Die Anwendung ist in drei Schichten gegliedert:

Präsentationsschicht (Frontend). Aufgabe: Darstellung der Benutzeroberfläche und Entgegennahme von Eingaben. Umsetzung über statische Seiten und Skripte (`public/index.html`, `public/employee.html`, `public/redeem.html`, CSS in `public/css/styles.css`, JavaScript in `public/js/app.js` und `public/js/employee.js`). Die Kommunikation erfolgt ausschließlich per `fetch` (JSON) mit der Anwendungsschicht unter `php/api.php`. Für die QR-Code-Erkennung in der Mitarbeiteransicht wird die @zxing-Library eingebunden.

Anwendungsschicht (Controller/Services). Aufgabe: Geschäftslogik und Ablaufsteuerung. `php/api.php` dient als Controller und bietet JSON-basierte Endpunkte (z. B. *register*, *validate_voucher*, *redeem_voucher*, *employee_login*). Die Logik ist in Service-Klassen gekapselt. Dazu gehören: `php/Auth.php` (Authentifizierung), `php/Csrf.php` (CSRF-Token), `php/Voucher.php` (Gutschein-Erstellung inkl. QR/PDF), `php/Mailer.php` (E-Mail-Versand und Protokollierung), `php/Crypto.php` (AES-256-GCM-Verschlüsselung) und `php/bootstrap.php` (Autoloading, `.env`, Zeitzone).

Datenzugriffsschicht (Repository/Persistenz). Aufgabe: Zugriff auf die MySQL-Datenbank und Persistenz. Dies erfolgt zentral über `php/Database.php` (PDO-Initialisierung). Alle Datenbank-Operationen werden mit Prepared Statements umgesetzt (z. B. in `php/api.php`, `php/Voucher.php`, `php/Mailer.php`). Genutzte Tabellen sind `users`, `vouchers`, `redemptions` und `mail_log`. Sensible Felder werden verschlüsselt gespeichert (`users.email_enc`, `users.email_iv`, `users.email_tag`). Zur Suche wird ein Hash verwendet (`users.email_hash`).

4.1.2 Schichtgrenzen und Kommunikation

Das Frontend ruft ausschließlich JSON-Endpunkte der Anwendungsschicht auf. Direkte Datenbankzugriffe aus dem Frontend finden nicht statt. Die Anwendungsschicht greift nur über die Datenzugriffsschicht (PDO) auf die Datenbank zu. Authentifizierung und Sitzungsverwaltung liegen in der Anwendungsschicht. CSRF-Schutz wird über Token realisiert.

4.1.3 Diagramm

Zur Visualisierung des Aufbaus dient Abbildung 1.

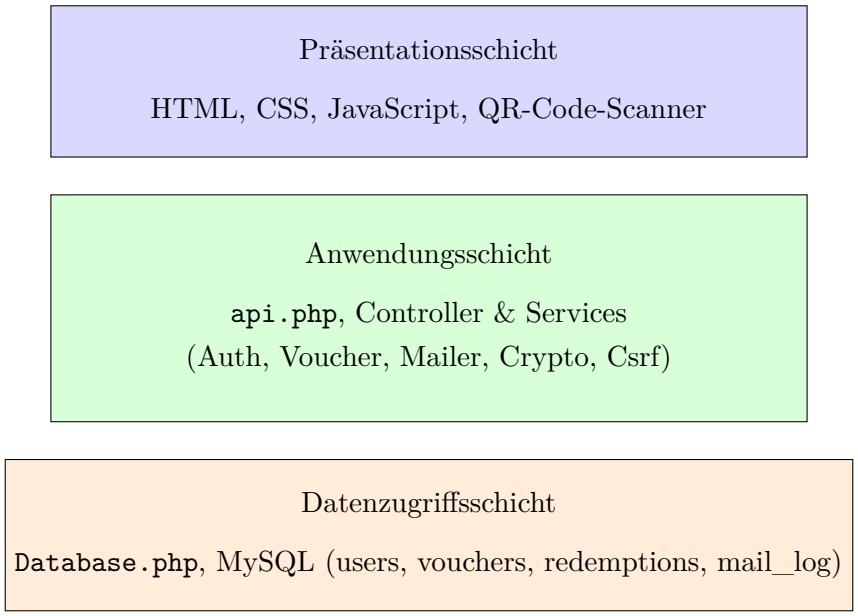

Abbildung 1: Schichtenarchitektur der Anwendung

5 Implementierungsphase

5.1 Implementierung der Datenstrukturen

Zu Beginn der Implementierungsphase habe ich die Datenbankstruktur auf Basis des in der Entwurfsphase erstellten ER-Modells in MySQL umgesetzt. Die Tabellen für Benutzer, Gutscheine, Einlösungen und das Mail-Log habe ich in einer separaten SQL-Datei (`schema.sql`) definiert und anschließend per MySQL-Client (phpMyAdmin) importiert.

```
-- Nutzer (Werber & Geworbene)
CREATE TABLE users (
  id BIGINT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
  email_enc VARBINARY(512) NOT NULL,
  email_iv VARBINARY(32) NOT NULL,
  email_tag VARBINARY(32) NOT NULL,
  email_hash CHAR(64) NOT NULL,
  referral_code CHAR(16) NOT NULL UNIQUE,
  referrer_id BIGINT UNSIGNED NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  INDEX (referrer_id),
  FOREIGN KEY (referrer_id) REFERENCES users(id) ON DELETE SET NULL
) ENGINE=InnoDB;

-- Gutscheine
CREATE TABLE vouchers (
  id BIGINT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
  user_id BIGINT UNSIGNED NOT NULL,
  code CHAR(24) NOT NULL UNIQUE,
  discount_percent INT NOT NULL,
  expires_at DATETIME NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
) ENGINE=InnoDB;

-- Einlösungen
CREATE TABLE redemptions (
  id BIGINT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
  voucher_id BIGINT UNSIGNED NOT NULL,
  employee_id BIGINT UNSIGNED NULL,
  redeemed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (voucher_id) REFERENCES vouchers(id) ON DELETE CASCADE
) ENGINE=InnoDB;

-- E-Mail-Versandlog
CREATE TABLE mail_log (
  id BIGINT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
  to_user_id BIGINT UNSIGNED NULL,
  subject VARCHAR(255) NOT NULL,
  success TINYINT(1) NOT NULL DEFAULT 0,
  error TEXT NULL,
```


```
42   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
43   INDEX idx_mail_log_to_user_id (to_user_id),
44   CONSTRAINT fk_mail_log_user FOREIGN KEY (to_user_id) REFERENCES users(id) ON DELETE
         SET NULL
45 ) ENGINE=InnoDB;
```

Listing 1: SQL-Definition der zentralen Tabellen

Die Verbindung zur Datenbank wird über eine zentrale `Database`-Klasse hergestellt, die PDO als Schnittstelle nutzt. Wichtig war mir, die Verbindung parametrisiert über Umgebungsvariablen (`.env`-Datei) zu gestalten, um den Host, Port oder Socket, SMTP-Zugangsdaten, Secret usw. flexibel ändern zu können und die sensiblen Daten zu schützen, da ich gleichzeitig eine Serverkonfigurationsdatei (.htaccess) erstellt habe, die den Zugriff auf die Datei mit den Umgebungsvariablen per URL verhindert (HTTP 403 - Forbidden Access Error bei Zugriffsversuch). Ebenso habe ich PDO im Exception-Modus aktiviert, um Fehler frühzeitig zu erkennen.

```
1  class Database {
2      private PDO $pdo;
3
4      public function __construct() {
5          $charset = $_ENV['DB_CHARSET'] ?? 'utf8mb4';
6          $host = $_ENV['DB_HOST'] ?? 'localhost';
7          $dsn = sprintf(
8              'mysql:host=%s;port=%s;dbname=%s;charset=%s',
9              $host,
10             $_ENV['DB_PORT'] ?? '3306',
11             $_ENV['DB_DATABASE'],
12             $charset
13         );
14         $options = [
15             PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
16             PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC
17         ];
18         $this->pdo = new PDO(
19             $dsn,
20             $_ENV['DB_USERNAME'],
21             $_ENV['DB_PASSWORD'],
22             $options
23         );
24     }
25 }
```

Listing 2: Konstruktor der Database-Klasse

5.2 Implementierung der Benutzeroberfläche

Die Benutzeroberfläche besteht aus drei HTML-Seiten (`index.html`, `employee.html`, `redeem.html`), die ich mit Bootstrap und einem eigenen Stylesheet gestaltet habe. Das Corporate Design ist

schlicht, mit klaren Flächen und gut lesbaren Schriften. Farblich habe ich mich für eine dezente Hintergrundfarbe (`whitesmoke`) und klare Kontraste bei Buttons und Formularen entschieden. Durch eigene CSS-Klassen konnte ich z. B. die Darstellung des Videoausschnitts im Mitarbeiterbereich optimieren.

```
body {
  padding-top: 24px;
  background-color: whitesmoke;
}
.card {
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.85);
  border: 0;
  background-color: whitesmoke;
}
.qr-preview {
  max-width: 280px;
}
```

Listing 3: Auszug aus styles.css

Die Interaktivität im Frontend habe ich mit JavaScript umgesetzt. Hierzu gehört z. B. die Anzeige von Rückmeldungen in einer Bootstrap-Alert-Box. Für die API-Kommunikation verwende ich die `fetch`-Funktion mit asynchronen Funktionen.

```
async function fetchJSON(url, options = {}) {
  const res = await fetch(url, options);
  let data = {};
  try {
    data = await res.json();
  } catch {
    data = {};
  }
  return { status: res.status, data };
}
```

Listing 4: Hilfsfunktion zur API-Kommunikation

5.3 Implementierung der Geschäftslogik

Die Geschäftslogik habe ich in einzelne PHP-Klassen aufgeteilt, um den Code übersichtlich zu halten. Ein wichtiger Bestandteil ist die Klasse `Voucher`, die die Erstellung von Gutscheinen übernimmt. Dabei wird ein eindeutiger Code generiert, in der Datenbank gespeichert und optional ein Ablaufdatum gesetzt. Zusätzlich wird aus den Gutschein-Daten direkt ein PDF mit QR-Code erzeugt, das per E-Mail an den Werber bzw. Geworbenen gesendet wird.

```
public static function create(PDO $pdo, int $userId, int $discountPercent, ?DateTimeImmutable $expires =
    null): array {
  $code = self::generateCode(24);
```

```
    $stmt = $pdo->prepare(
        'INSERT INTO vouchers (user_id, code, discount_percent, expires_at) VALUES (?, ?, ?, ?)'
    );
    $stmt->execute([$userId, $code, $discountPercent, $expires?->format('Y-m-d H:i:s')]);
    return ['id' => (int)$pdo->lastInsertId(), 'code' => $code];
}
```

Listing 5: Auszug aus der Voucher-Klasse

Das QR-Code- und PDF-Rendering erfolgt direkt in PHP über die Bibliotheken `endroid/qr-code` und `dompdf/dompdf`. So konnte ich sicherstellen, dass der komplette Prozess (vom Erstellen bis zum Versenden) serverseitig abläuft.

Screenshots der Anwendung in der Entwicklungsphase mit Dummy-Daten befinden sich im Anhang *A.5*.

6 Abnahmephase

6.1 Test und Qualitätssicherung

6.2 Teststrategie

- Da es sich um ein Ausbildungsprojekt handelt und der Fokus auf der funktionalen Umsetzung lag, habe ich in dieser Projektarbeit ausschließlich manuelle Tests durchgeführt. Automatisierte Tests waren nicht Bestandteil des Projektplans.

6.3 Testmethodik

Die Testfälle habe ich aus den in der Planungsphase definierten Use Cases abgeleitet. Für jeden Use Case habe ich die relevanten Eingaben und erwarteten Ausgaben definiert. Die Tests habe ich anschließend schrittweise im Browser und über direkte API-Aufrufe geprüft.

- **Registrierung**: Formular mit gültigen und ungültigen E-Mail-Adressen getestet, um Eingabevalidierung zu prüfen.
- **Login**: Sowohl korrekte als auch falsche Zugangsdaten eingegeben, um Fehlermeldungen zu validieren.
- **Gutschein-Erstellung**: Test mit verschiedenen Rabattwerten und Ablaufdaten; anschließend Überprüfung, ob der Gutschein in der Datenbank gespeichert wurde.
- **Gutschein-Einlösung**: QR-Code mit der Mitarbeiteransicht gescannt und geprüft, ob die Einlösung korrekt im Backend protokolliert wird.

- **E-Mail-Versand**: Kontrolle, ob nach erfolgreicher Gutschein-Erstellung eine E-Mail mit PDF-Anhang und QR-Code beim Nutzer ankommt.

6.4 Testergebnisse

Alle in der Testphase definierten Use Cases konnten erfolgreich und ohne kritische Fehler durchlaufen werden. Kleinere Darstellungsprobleme im Firefox wurden durch Anpassung des CSS behoben. Die Anwendung erfüllt damit die funktionalen Anforderungen aus der Planungsphase.

Tabelle 1: Testergebnisse Registrierung und Gutschein-Erstellung

ID	**Beschreibung**	**Eingabe/Aktion**	**Erwartetes Ergebnis**	**Status**
T-001	Registrierung mit gültigen Daten	Gültige E-Mail eingeben, Formular absenden	Bestätigung im Frontend, QR-Code per Mail	bestanden
T-002	Registrierung mit ungültiger E-Mail	Ungültige E-Mail eingeben, Formular absenden	Fehlermeldung „Bitte gültige E-Mail“	bestanden
T-003	Gutschein-Erstellung ohne Ablaufdatum	Rabatt 20% auswählen, kein Ablaufdatum	Gutschein gespeichert, PDF mit QR-Code per Mail	bestanden
T-004	Gutschein-Erstellung mit Ablaufdatum	Rabatt 10% auswählen, Ablaufdatum setzen	Gutschein gespeichert mit Ablaufdatum, PDF per Mail	bestanden

Tabelle 2: Testergebnisse Einlösung, Sicherheit und Layout

ID	**Beschreibung**	**Eingabe/Aktion**	**Erwartetes Ergebnis**	**Status**
T-005	Gutschein-Einlösung durch Mitarbeiter	QR-Code scannen und bestätigen	Eintrag in DB redemptions, Anzeige „eingelöst“	bestanden
T-006	Gutschein-Einlösung mit ungültigem Code	Falschen Code eingeben	Fehlermeldung „Ungültig/abgelaufen“	bestanden
T-007	E-Mail-Versand bei Gutschein-Erstellung	Gutschein anlegen	Eintrag in DB mail_log, E-Mail erhalten	bestanden
T-008	CSRF-Schutzprüfung	Formular ohne gültigen Token absenden	Fehlermeldung im Frontend, kein DB-Eintrag	bestanden
T-009	Responsives Layout	Anwendung auf Smartphone öffnen, in allen gängigen Browsern	Inhalte lesbar, Elemente innerhalb des Viewports, QR-Scanner funktioniert	bestanden

6.5 Zusammenfassung der Testergebnisse

Alle definierten Testfälle konnten erfolgreich abgeschlossen werden. Die Tests haben gezeigt, dass die Anwendung die in der Anforderungsdefinition festgelegten Funktionen zuverlässig umsetzt. Kleinere Darstellungsprobleme im Firefox wurden während der Testphase behoben, sodass die Anwendung nun in gängigen Browsern ohne Einschränkungen nutzbar ist.

6.6 Abnahme

Die Abnahme war erfolgreich und erfolgte durch meinen Chef. Akzeptanzkriterien: Funktionale Anforderungen, Sicherheit, Nachvollziehbarkeit, PDF-Layout, E-Mail-Templates erfüllt.

7 Fazit

Das Projektziel, eine funktionsfähige, sichere und benutzerfreundliche Empfehlungsmarketing-App zu erstellen, wurde erreicht. Der Prozess von der Generierung eines Affiliate-Links und/oder Gutscheins bis zum Emailversand an den Kunden, ist automatisiert und nachvollziehbar. Die Architektur ist modular und erweiterbar. Künftige Erweiterungen (z. B. erweiterte Gutscheinregeln) sind vorgesehen.

A Anhang

A.1 Detaillierte Zeitplanung

Analysephase			**9 h**
1. Analyse des Ist-Zustands		3 h	
1.1. Fachgespräch mit der EDV-Abteilung	1 h		
1.2. Prozessanalyse	2 h		
2. „Make or buy“-Entscheidung und Wirtschaftlichkeitsanalyse		1 h	
3. Erstellen eines „Use-Case“-Diagramms		2 h	
4. Erstellen des Lastenhefts mit der EDV-Abteilung		3 h	
Entwurfsphase			**19 h**
1. Prozessentwurf		2 h	
2. Datenbankentwurf		3 h	
2.1. ER-Modell erstellen	2 h		
2.2. Konkretes Tabellenmodell erstellen	1 h		
3. Erstellen von Datenverarbeitungskonzepten		4 h	
3.1. Verarbeitung der CSV-Daten	1 h		
3.2. Verarbeitung der SVN-Daten	1 h		
3.3. Verarbeitung der Sourcen der Programme	2 h		
4. Benutzeroberflächen entwerfen und abstimmen		2 h	
5. Erstellen eines UML-Komponentendiagramms der Anwendung		4 h	
6. Erstellen des Pflichtenhefts		4 h	
Implementierungsphase			**29 h**
1. Anlegen der Datenbank		1 h	
2. Umsetzung der HTML-Oberflächen und Stylesheets		4 h	
3. Programmierung der PHP-Module für die Funktionen		23 h	
3.1. Import der Modulinformationen aus CSV-Dateien	2 h		
3.2. Parsen der Modulquelltexte	3 h		
3.3. Import der SVN-Daten	2 h		
3.4. Vergleichen zweier Umgebungen	4 h		
3.5. Abrufen der von einem zu wählenden Benutzer geänderten Module	3 h		
3.6. Erstellen einer Liste der Module unter unterschiedlichen Aspekten	5 h		
3.7. Anzeigen einer Liste mit den Modulen und geparsten Metadaten	3 h		
3.8. Erstellen einer Übersichtsseite für ein einzelnes Modul	1 h		
4. Nächtlichen Batchjob einrichten		1 h	
Abnahmetest der Fachabteilung			**1 h**
1. Abnahmetest der Fachabteilung		1 h	
Einführungsphase			**1 h**
1. Einführung/Benutzerschulung		1 h	
Erstellen der Dokumentation			**9 h**
1. Erstellen der Benutzerdokumentation		2 h	
2. Erstellen der Projektdokumentation		6 h	
3. Programmdokumentation		1 h	
3.1. Generierung durch PHPdoc	1 h		
Pufferzeit			**2 h**
1. Puffer		2 h	
Gesamt			**70 h**

A.2 Lastenheft (Auszug)

Es folgt ein Auszug aus dem Lastenheft mit Fokus auf die Anforderungen:

Die Anwendung muss folgende Anforderungen erfüllen:

1. Benutzeroberfläche (GUI)
   - 1.1. Die Webanwendung muss über ein GUI verfügen, über das sich sowohl Kunden, als auch Interessenten mit ihrer Email-Adresse registrieren können.
   - 1.2. Das Design des GUI soll schlicht und modern gehalten werden.
   - 1.3. Außerdem soll es einen Mitarbeiterbereich mit Login geben, in dem Mitarbeiter die Möglichkeit haben den QR-Code von Gutscheinen zu scannen und die Gültigkeit zu überprüfen, sowie den Gutschein zu entwerten.
2. Responsives Design
   - 2.1. Die Webanwendung soll in allen gängigen Browsern funktionieren.
   - 2.2. Sie soll auf Monitoren aller Größen über den PC nutzbar sein.
   - 2.3. Außerdem soll sie über mobile Geräte nutzbar sein.
3. Sicherheit und Datenschutz
   - 3.1. Die Kundendaten (Email-Adresse) sollen in verschlüsselter Form in der Datenbank gespeichert werden, um sensible Daten zusätzlich zu schützen
   - 3.2. Die Datenübertragungen sollen mit SSL/TLS gesichert sein.
   - 3.3. Die DSGVO Konformität soll gewährleistet sein.
4. Automatische Affiliate-Link und Gutscheinerzeugung und Versand (Backend)
   - 4.1. Bei Registrierung eines Kunden per Email-Adresse soll dieser automatisch einen Affiliate-Link zugesandt bekommen, welcher mit seiner user_id in der Datenbank assoziiert ist.
   - 4.2. Registriert sich ein Interessent durch Aufruf des Affiliate-Links eines Kunden, soll dem Interessenten ein Affiliate-Link und ein Gutschein im PDF-Format, inklusive QR-Code zum Einlösen des Gutscheins, automatisch zugesandt werden.
   - 4.3. Wird der Gutschein eines Interessenten durch einen Mitarbeiter durch Scannen des QR-Codes auf dem Gutschein eingelöst, soll dem Kunden (Werber) ebenfalls ein Gutschein im PDF-Format erzeugt und zugesandt werden.

A.3 Use Case-Diagramm

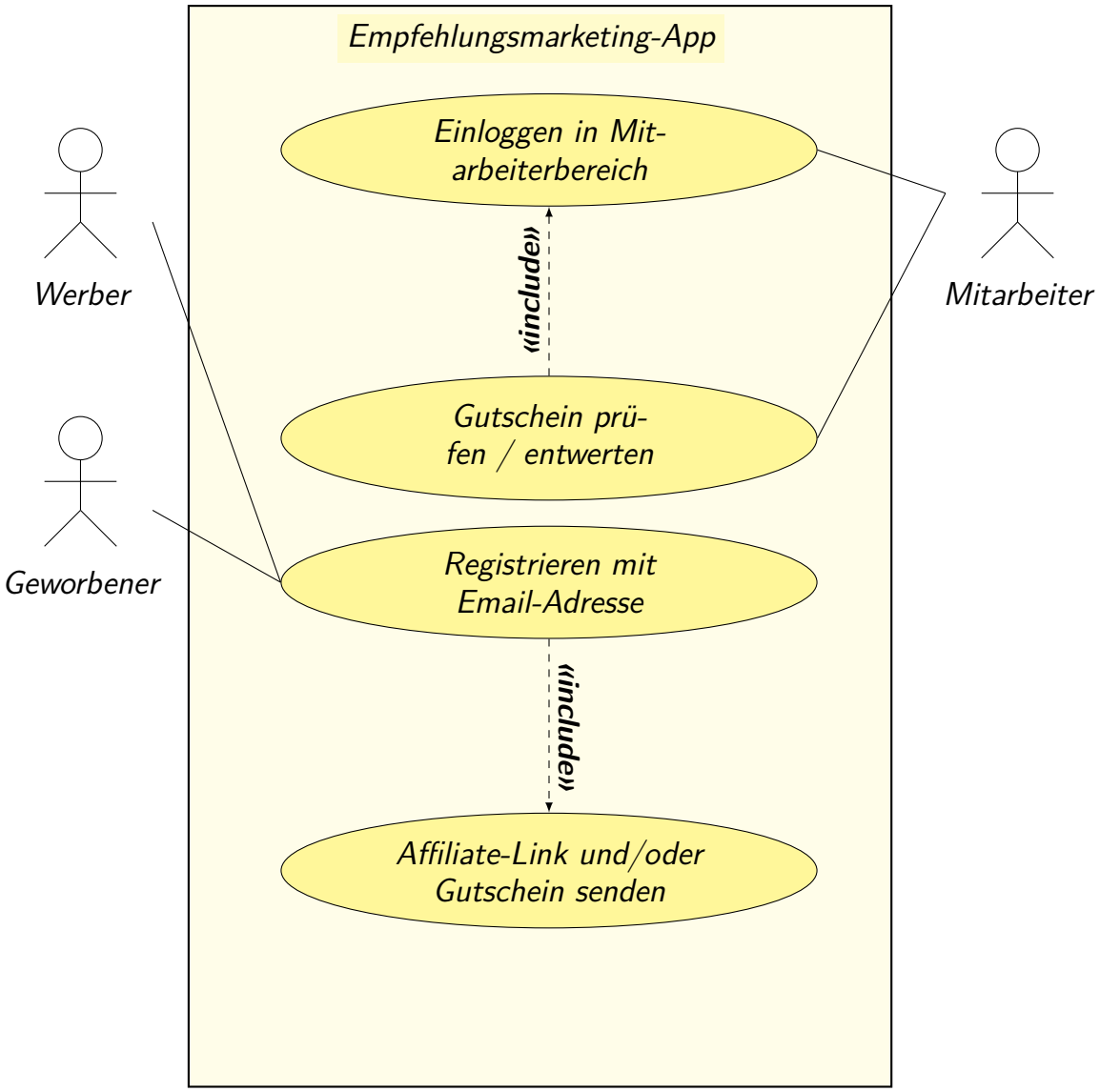

Abbildung 2: Use Case-Diagramm

A.4 Datenbankmodell

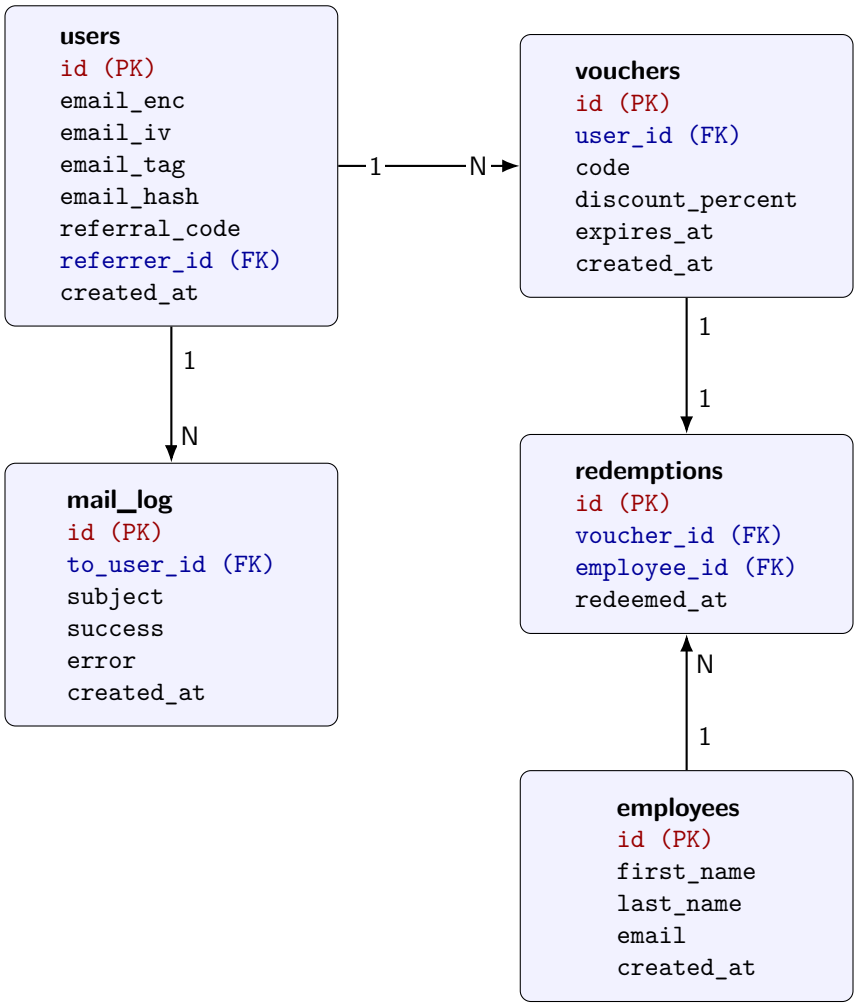

Abbildung 3: Vereinfachtes Datenbankschema (MySQL)

A.5 Screenshots der Anwendung

A.5.1 Registrierungsformular

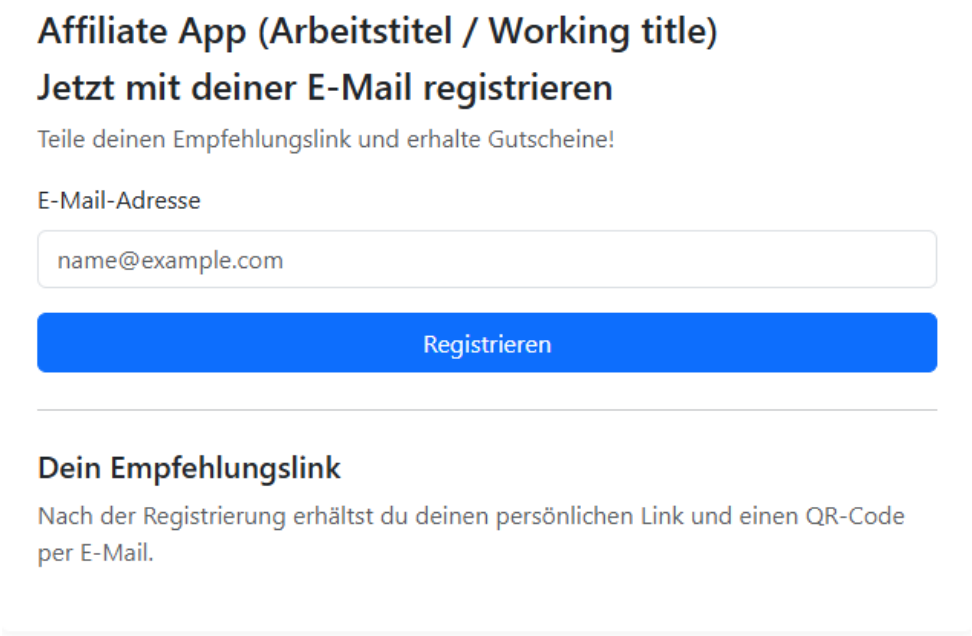

Abbildung 4: Leeres Formular

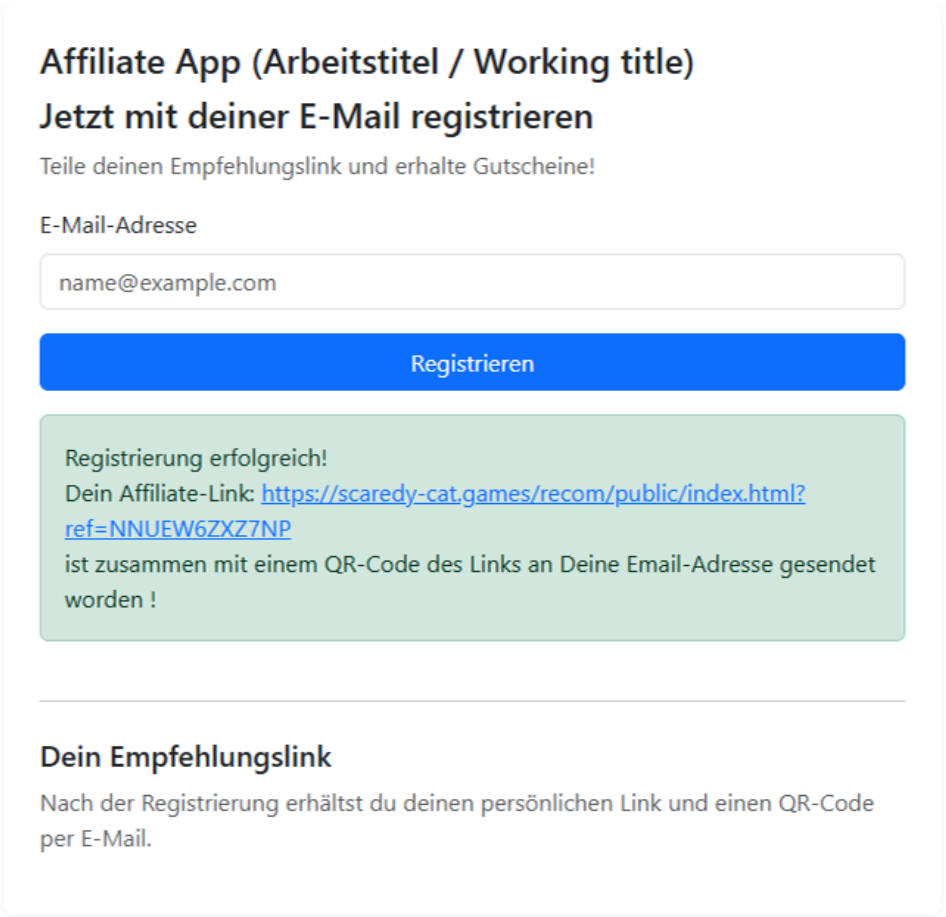

Abbildung 5: Erfolgreiche Registrierung

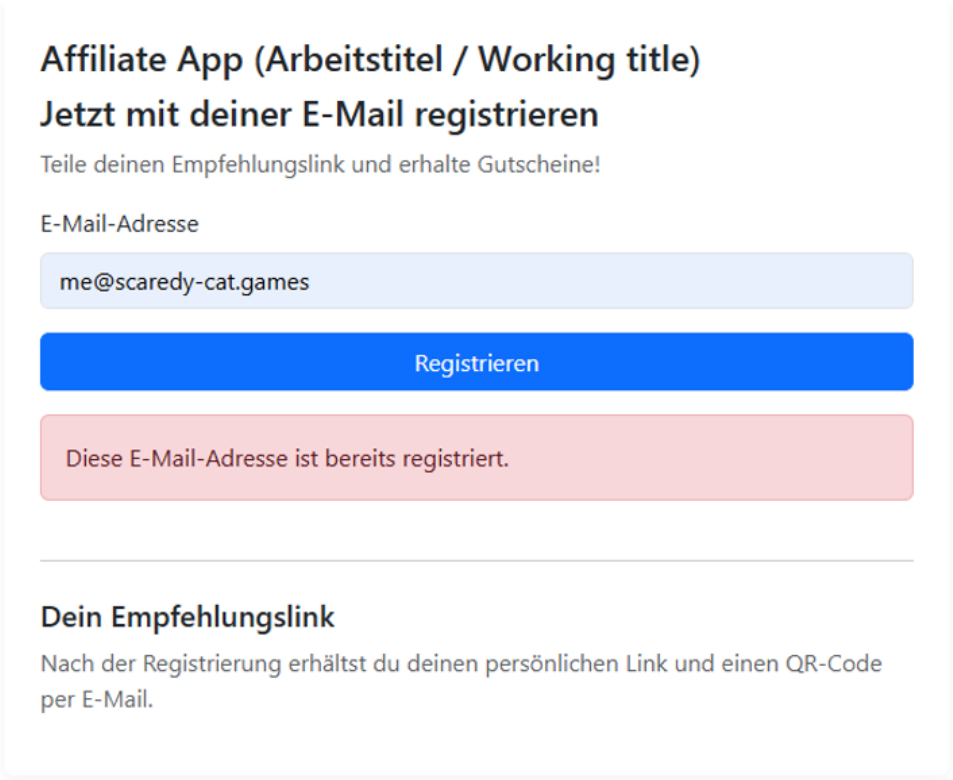

Abbildung 6: Fehlermeldung bei doppeltem Registrierungsversuch

A.5.2 Emails an Werber oder Geworbenen

Abbildung 7: Gutschein an Werber/Geworbenen

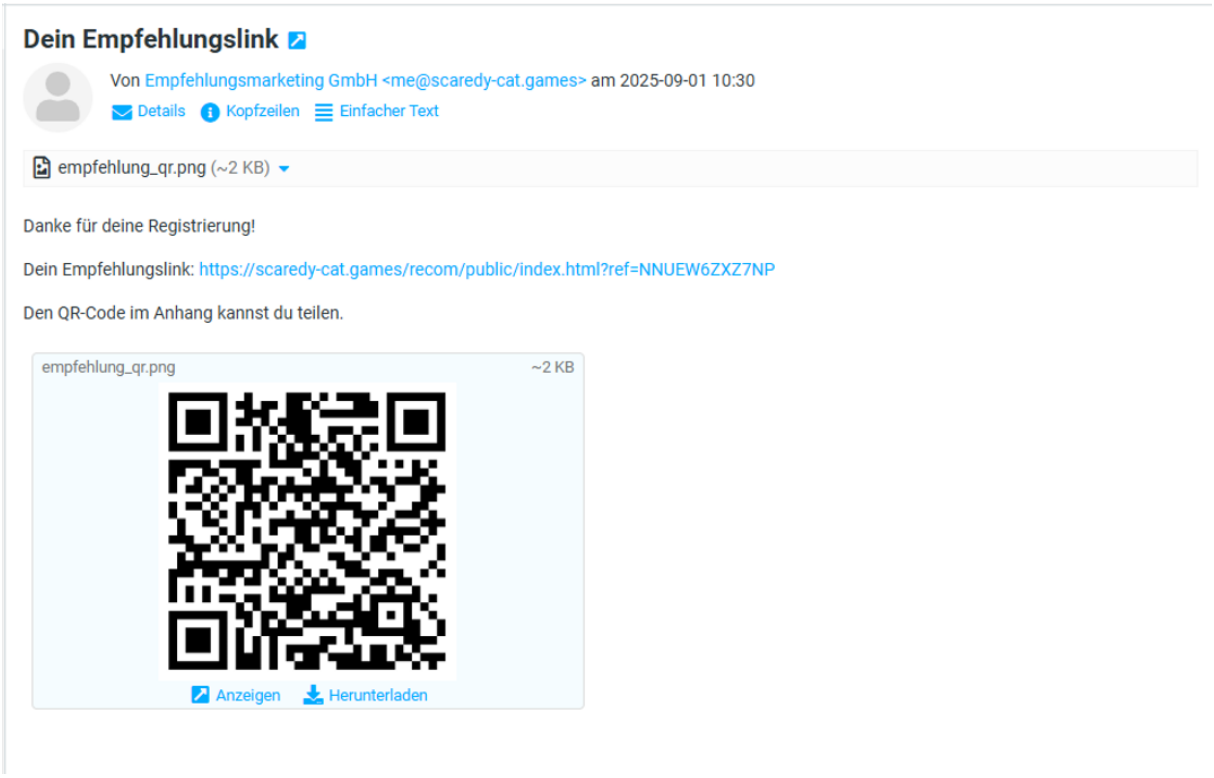

Abbildung 8: Affiliate-Link und QR-Code an Werber/Geworbenen