Meilenstein 3

Dokumentation der PoCs

-Suchfunktion, Buch-Artikel finden (und Topic erstellen)

TODO

-Bibliothek und Wunschliste erzeugen und ändern (Topics)

Functions:

```
// Fill table with data
function populateWishTable() {

    // Empty content string
    var tableContent = '';

    // jQuery AJAX call for JSON
    $.getJSON( '/users/user1wish', function( data ) {

        // For each item in our JSON, add a table row and cells to the content string
        $.each(data, function(){
            tableContent += '<tr>';
            tableContent += '<td>' + this.author + '</td>';
            tableContent += '<td>' + this.titel + '</td>';
            tableContent += '<td>' + this.edition + '</td>';
            tableContent += '<td>' + this.publisher + '</td>';
            tableContent += '<td>' + this.printrun + '</td>';
            tableContent += '<td>' + this.isbn10 + '</td>';
            tableContent += '<td>' + this.isbn13 + '</td>';
            tableContent += '<td>' + this.status + '</td>';
            tableContent += '<td><a href="#" class="linkdeletefromuser1wish" rel="' + this._id +
'">delete</a></td>';
//            tableContent += '<td><a href="#" class="linkchangestatususer1wish" rel="' + this._id +
'">change status</a></td>';
            tableContent += '</tr>';
        });

        // Inject the whole content string into our existing HTML table
        $('#user1WishList table tbody').html(tableContent);
    });
};

// Add to User1Lib
function addToUser1Lib(event) {
    event.preventDefault();

    // Super basic validation - increase errorCount variable if any fields are blank
```

```
    var errorCount = 0;
    $('#addToUser1Lib input').each(function(index, val) {
        if($(this).val() === '') { errorCount++; }
    });

    // Check and make sure errorCount's still at zero
    if(errorCount === 0) {

        // If it is, compile all user info into one object
        var newEntry = {
            'author': $('#addToUser1Lib fieldset input#inputAuthor').val(),
            'titel': $('#addToUser1Lib fieldset input#inputTitel').val(),
            'edition': $('#addToUser1Lib fieldset input#inputEdition').val(),
            'publisher': $('#addToUser1Lib fieldset input#inputPublisher').val(),
            'printrun': $('#addToUser1Lib fieldset input#inputPrintRun').val(),
            'isbn10': $('#addToUser1Lib fieldset input#inputISBN10').val(),
            'isbn13': $('#addToUser1Lib fieldset input#inputISBN13').val(),
            'status': $('#addToUser1Lib fieldset input#inputStatus').val()
        }

        // Use AJAX to post the object to our adduser service
        $.ajax({
            type: 'POST',
            data: newEntry,
            url: '/users/addtouser1lib',
            dataType: 'JSON'
        }).done(function( response ) {

            // Check for successful (blank) response
            if (response.msg === '') {

                // Clear the form inputs
                $('#addToUser1Lib fieldset input').val('');

                // Update the table
                populateLibTable();

            }
            else {

                // If something goes wrong, alert the error message that our service returned
                alert('Error: ' + response.msg);

            }
        });
    }
    else {
        // If errorCount is more than 0, error out
        alert('Please fill in all fields');
        return false;
    }
}
```

```
// Add to User1Wish
function addToUser1Wish(event) {
    event.preventDefault();

    // Super basic validation - increase errorCount variable if any fields are blank
    var errorCount = 0;
    $('#addToUser1Wish input').each(function(index, val) {
        if($(this).val() === '') { errorCount++; }
    });

    // Check and make sure errorCount's still at zero
    if(errorCount === 0) {

        // If it is, compile all user info into one object
        var newEntry = {
            'author': $('#addToUser1Wish fieldset input#inputAuthor').val(),
            'titel': $('#addToUser1Wish fieldset input#inputTitel').val(),
            'edition': $('#addToUser1Wish fieldset input#inputEdition').val(),
            'publisher': $('#addToUser1Wish fieldset input#inputPublisher').val(),
            'printrun': $('#addToUser1Wish fieldset input#inputPrintRun').val(),
            'isbn10': $('#addToUser1Wish fieldset input#inputISBN10').val(),
            'isbn13': $('#addToUser1Wish fieldset input#inputISBN13').val(),
            'status': $('#addToUser1Wish fieldset input#inputStatus').val()
        }

        // Use AJAX to post the object to our adduser service
        $.ajax({
            type: 'POST',
            data: newEntry,
            url: '/users/addtouser1wish',
            dataType: 'JSON'
        }).done(function( response ) {

            // Check for successful (blank) response
            if (response.msg === '') {

                // Clear the form inputs
                $('#addToUser1Wish fieldset input').val('');

                // Update the table
                populateWishTable();

            }
            else {

                // If something goes wrong, alert the error message that our service returned
                alert('Error: ' + response.msg);

            }
        });
    }
```

```
    else {
        // If errorCount is more than 0, error out
        alert('Please fill in all fields');
        return false;
    }
}

// Delete From User1lib
function deleteFromUser1Lib(event) {

    event.preventDefault();

    // Pop up a confirmation dialog
    var confirmation = confirm('Are you sure you want to delete this entry?');

    // Check and make sure the user confirmed
    if (confirmation === true) {

        // If they did, do our delete
        $.ajax({
            type: 'DELETE',
            url: '/users/deletefromuser1lib/' + $(this).attr('rel')
        }).done(function( response ) {

            // Check for a successful (blank) response
            if (response.msg === '') {
            }
            else {
                alert('Error: ' + response.msg);
            }

            // Update the table
            populateLibTable();

        });

    }
    else {

        // If they said no to the confirm, do nothing
        return false;

    }

};

// Delete User From User1wish
function deleteFromUser1Wish(event) {

    event.preventDefault();

    // Pop up a confirmation dialog
```

```
    var confirmation = confirm('Are you sure you want to delete this entry?');

    // Check and make sure the user confirmed
    if (confirmation === true) {

        // If they did, do our delete
        $.ajax({
            type: 'DELETE',
            url: '/users/deletefromuser1wish/' + $(this).attr('rel')
        }).done(function( response ) {

            // Check for a successful (blank) response
            if (response.msg === '') {
            }
            else {
                alert('Error: ' + response.msg);
            }

            // Update the table
            populateWishTable();

        });

    }
    else {

        // If they said no to the confirm, do nothing
        return false;

    }

};

Clicks:
    // Add Entry to user1Library button click
    $('#btnAddToUser1Lib').on('click', addToUser1Lib);

    // Add Entry to user1WishList button click
    $('#btnAddToUser1Wish').on('click', addToUser1Wish);

    // Delete from User1library click
    $('#user1Library table tbody').on('click', 'td a.linkdeletefromuser1lib', deleteFromUser1Lib);

    // Delete from User1wishlist click
    $('#user1WishList table tbody').on('click', 'td a.linkdeletefromuser1wish', deleteFromUser1Wish);

Routes:
/*
 * POST to add to user1lib
 */
router.post('/addtouser1lib', function(req, res) {
    var db = req.db;
```

```
    var collection = db.get('user1lib');
    collection.insert(req.body, function(err, result){
        res.send(
            (err === null) ? { msg: '' } : { msg: err }
        );
    });
});

/*
 * POST to add to user1wish
 */
router.post('/addtouser1wish', function(req, res) {
    var db = req.db;
    var collection = db.get('user1wish');
    collection.insert(req.body, function(err, result){
        res.send(
            (err === null) ? { msg: '' } : { msg: err }
        );
    });
});

/*
 * DELETE to delete entry user1lib
 */
router.delete('/deletefromuser1lib/:id', function(req, res) {
    var db = req.db;
    var collection = db.get('user1lib');
    var entryToDelete = req.params.id;
    collection.remove({ '_id' : entryToDelete }, function(err) {
        res.send((err === null) ? { msg: '' } : { msg:'error: ' + err });
    });
});

/*
 * DELETE to delete entry user1wish
 */
router.delete('/deletefromuser1wish/:id', function(req, res) {
    var db = req.db;
    var collection = db.get('user1wish');
    var entryToDelete = req.params.id;
    collection.remove({ '_id' : entryToDelete }, function(err) {
        res.send((err === null) ? { msg: '' } : { msg:'error: ' + err });
    });
});
```

-Benachrichtigung bei Übereinstimmung an Clients versenden

TODO

Benutzermodelle (user models)

Identifizierung der primären, sekundären und tertiären Benutzer:

-Primäre Benutzer
Personen, die gerne und viel lesen und ihre Bücher gerne weitergeben.
Personen, die ihre Bücher gerne gebraucht erwerben zum Beispiel auf Flohmärkten.
Personen, die ihre Bücher ungern an Organisationen verkaufen möchten, die diese weiterverkaufen.
Personen, die ihre Bibliothek online verwalten wollen.
Personen, die sich für die Werke bestimmter Autoren interessieren.
Personen, die Kontakt zu anderen Lesern suchen.
Personen, die sich für das Experiment "Weg des Buches" interessieren, wie bei der Booksharing-Bewegung.
Personen, die beim Erwerb eines Buches Geld sparen wollen.
Personen, die aus ökologischen Gründen keine Neuware kaufen wollen.

-Sekundäre Benutzer
Personen, die Bücher, die sie selbst nicht lesen wollen (z.B. Geschenk) gerne tauschen möchten.
Personen, die ein bestimmtes Buch gerne weitergeben möchten.

-Tertiäre Benutzer
Kommerzielle Anbieter, deren Angebote angezeigt werden.

Charakterisierung der Benutzer:
Zuordnung relevanter Merkmale zu den Benutzern:

Merkmal/ Person	Demographie	Beruf	Qualifikation	Computerkenntnis	Fähigkeiten	Verfügbare Technologien	Produkterfahrung	Motive
1	Nicht speziell	Berufe mit viel Freizeit	Lesen/ Schreiben	gering bis Durchschnitt	Tastatur/Maus/Bildschirm benutzen	PC mit bel. Betriebssystem	Online kauf	Weitergeben der Bücher
2	mittleres Alter	Wochenende frei	Lesen/ Schreiben	Nicht speziell	Tastatur/Maus/Bildschirm benutzen	PC mit bel. Betriebssystem	Onlinekauf booksharing	Gebraucht kaufen
3	Ab mittleres Alter	Nicht speziell	Lesen/ Schreiben	Nicht speziell	Tastatur/Maus/Bildschirm benutzen	PC mit bel. Betriebssystem	Onlinekauf booksharing	Keine kommerzielle Handlung
4	Nicht speziell	Nicht speziell	Lesen/ Schreiben	Durchschnitt bis hoch	Tastatur/Maus/Bildschirm benutzen	PC mit bel. Betriebssystem	Onlinetools Cloud	Verwaltung
5	Nicht speziell	Nicht speziell	Lesen/ Schreiben	Nicht speziell	Tastatur/Maus/Bildschirm benutzen	PC mit bel. Betriebssystem	Nicht speziell	Sammeln
6	Nicht speziell	Nicht speziell	Lesen/ Schreiben	Nicht speziell	Tastatur/Maus/Bildschirm benutzen	PC mit bel. Betriebssystem	Foren Kontaktbörse	Kontakte
7	Nicht speziell	Nicht speziell	Lesen/ Schreiben	Nicht speziell	Tastatur/Maus/Bildschirm benutzen	PC mit bel. Betriebssystem	booksharing	booksharing
8	Nicht speziell	Berufe mit geringem Einkommen	Lesen/ Schreiben	Nicht speziell	Tastatur/Maus/Bildschirm benutzen	PC mit bel. Betriebssystem	Auktionshäuser Kleinanzeigen Tauschbörsen	Geld sparen
9	Nicht speziell	Nicht speziell	Lesen/	Nicht speziell	Tastatur/Maus/Bildschirm benutzen	PC mit bel. Betriebssystem	Nicht speziell	Umwelt

Merkmal/ Person	Demographie	Beruf	Qualifikation	Computerkenntnis	Fähigkeiten	Verfügbare Technologien	Produkterfahrung	Motive
			Schreiben					schutz
10	Nicht speziell	Nicht speziell	Lesen/ Schreiben	Nicht speziell	Tastatur/Maus/Bildschirm benutzen	PC mit bel. Betriebssystem	Auktionshäuser Kleinanzeigen Tauschbörsen	Umtausch
11	Nicht speziell	Nicht speziell	Lesen/ Schreiben	Nicht speziell	Tastatur/Maus/Bildschirm benutzen	PC mit bel. Betriebssystem	Auktionshäuser Kleinanzeigen Tauschbörsen	Erfahrung teilen
12	-	Online Buchhändler	Lesen/ Schreiben	hoch	Tastatur/Maus/Bildschirm benutzen	PC mit bel. Betriebssystem	Gesammte Domäne	Vertrieb Trendanalyse

-Primäre Benutzer
1 Personen, die gerne und viel lesen und ihre Bücher gerne weitergeben.
2 Personen, die ihre Bücher gerne gebraucht erwerben zum Beispiel auf Flohmärkten.
3 Personen, die ihre Bücher nicht an Organisationen verkaufen möchten, die diese weiterverkaufen.
4 Personen, die ihre Bibliothek online verwalten wollen.
5 Personen, die sich für die Werke bestimmter Autoren interessieren.
6 Personen, die Kontakt zu anderen Lesern suchen.
7 Personen, die sich für das Experiment "Weg des Buches" interessieren, wie bei der Booksharing-Bewegung.
8 Personen, die beim Erwerb eines Buches Geld sparen wollen.
9 Personen, die aus ökologischen Gründen keine Neuware kaufen wollen.

-Sekundäre Benutzer
10 Personen, die Bücher, die sie selbst nicht lesen wollen (z.B. Geschenk) gerne tauschen möchten.
11 Personen, die ein bestimmtes Buch gerne weitergeben möchten.

-Tertiäre Benutzer
12 Kommerzielle Anbieter, deren Angebote angezeigt werden.

Anforderungen (user needs)

Zusammenfassung und Priorisieren der Benutzer und was das System ihnen bieten muss:

1 Benutzerwunsch: Bibliotheksverwaltung nutzen (abhängig von Suche).
System -> Bibliothek (anzeigen, löschen, hinzufügen (Suche (Autor, Titel, ISBN))

2 Benutzerwunsch: "Tausch"-Funktion nutzen (abhängig von der Bibliotheks- und Wunschlistenverwaltung).
System -> Bibliothek und Wunschliste (anzeigen, löschen, hinzufügen), Benachrichtigung bei Übereinstimmung (Tausch anbieten)

3 Benutzerwunsch: "Stöbern"-Funktion nutzen (abhängig von der Bibliotheks- und Wunschlistenverwaltung).
System -> Bibliothek (anzeigen, löschen, hinzufügen), Stöbern (anzeigen der Übereinstimmungen, zur Wunschliste hinzufügen anbieten, Wunschliste (hinzufügen), Benachrichtigung bei Übereinstimmung (Tausch anbieten))

Benutzungsmodelle (task models)

Use case : Start der Anwendung

Benutzer	System
1.Start Applikation, Verbinde mit buchtausch.de	
	2. Identifikation fordern
3. Passwort und Benutzername aus lokalen Speicher an System	
	4. Verbinde mit Datenbank benutzerdaten und Überprüfung der Benutzerdaten
	5. Hole Datensatz benutzername bibliothek und benutzername wunschliste und benutzername übereinstimmung und benutzername angebote und RETURN HTML , oder notification „keine Benutzerdaten gespeichert“ und weiter mit use case : Anmelden
6. Optionen: hinzufügen(Bibliothek oder Wunschliste)/löschen(Bibliotheks- oder Wunschlisteneintrag)/lösche Liste(Bibliothek oder Wunschliste)/Tausch anbieten(für jeden Eintrag in Übereinstimmung)/Tausch annehmen(für jeden Eintrag in angebote)/suchen/stöbern/schließen	
	7. weiter mit use case : hinzufügen/suchen, oder use case : löschen, oder use case : stöbern, oder use case : tauschangebot, oder use case : Tausch annehmen , oder schließe Anwendung

Use case : Anmelden

Benutzer	System
1. GET buchtausch.de/anmelden	
2. Optionen: Eingabefelder ausfüllen und CheckBox 'Mit diesem Gerät immer automatisch anmelden'	
3a. (CheckBox ist checked) speichere LogInn Daten lokal und weiter	
3b. (CheckBox ist unchecked) weiter	
	4. Checke, ob Eingabe-Felder ausgefüllt sind

	5. Wenn ja: weiter mit use case : Start der Anwendung (4.) Wenn nein: notification 'Bitte LogInn Daten ausfüllen' und weiter mit 2.

Use case : hinzufügen(Bibliothek oder Wunschliste), suchen

Benutzer	System
1a. hinzufügen(Bibliothek oder Wunschliste)	
1b. suchen	
	2. GET buchtausch.de/suche/autor
(Optional:) 3. Optionen: titel-suche/ISBN-suche	
	(Optional:) 4. GET buchtausch.de/suche/isbn bzw. /titel und RETURN HTML
5. Optionen: Eingabe-Feld ausfüllen und suchen/back/close	
	6. Checke, ob Eingabe-Feld ausgefüllt ist , oder GET und RETURN HTML , oder schließe Anwendung
	7. Wenn ja: Verbinden mit externer Datenbank und Suche nach Übereinstimmung in Datenbank und füge übereinstimmenden Eintrag zu ErgebnisSuche[] hinzu und RETURN HTML Wenn nein: notification 'bitte Eingabe-Feld ausfüllen' und weiter mit 5.
7a. Optionen: hinzufügen(für jedes zurückgegebene Buch)/back/close	
7b. Optionen: hinzufügen zu Bibliothek(für jedes zurückgegebene Buch)/hinzufügen zu Wunschliste(für jedes zurückgegebene Buch)/back/close	
	8. checke, ob Eintrag vorhanden in Datenbank benutzername bibliothek bzw. wunschliste Wenn ja: RETURN confirmation „Eintrag bereits vorhanden, erneut hinzufügen?“ Wenn nein: weiter mit 10. , oder GET und RETURN HTML, oder schließe Anwendung
9. Optionen: 'Ja, Eintrag erneut hinzufügen(Buch doppelt)', 'Nein, Eintrag nicht	

erneut hinzufügen'	
	10. INSERT Eintrag(itemID) in Datenbank benutzername bibliothek bzw. wunschliste und INSERT Eintrag(userID) in Datenbank alleBibliothek bzw. alleWunschliste und checke match und GET und RETURN HTML

Use case : Löschen, löschen/id

Benutzer	System
1a. löschen(Bibliothek oder Wunschliste)	
1b. löschen Eintrag(aus Bibliothek oder Wunschliste)	
	2. Confirmation
3. Optionen: 'Ja, löschen'/'Nein, lieber doch nicht'	
	4. Wenn nein: GET und RETURN HTML Wenn ja: a (Liste löschen): DELETE ((itemID) alle Einträge in Datenbank Benutzername Bibliothek bzw. Wunschliste) und DELETE (Einträge (userID) in Datenbank alleBibliotheken bzw. alleWunschlisten) und GET und RETURN HTML b (einzelnen Eintrag löschen): DELETE/id ((itemID) in Datenbank Benutzername Bibliothek bzw. Wunschliste) und DELETE (Eintrag (userID) in Datenbank alleBibliotheken bzw. alleWunschlisten) und GET und RETURN HTML

Stöbern

Benutzer	System
1. Stöbern	
	2. Vergleiche jeden Eintrag in Datenbank Benutzername Bibliothek mit mit den Einträgen in Datenbanken alleWunschliste(itemID), bei Übereinstimmung füge dem ErgebnisFürEintrag[] die Einträge aus Datenbank Benutzername Bibliothek(der zurückgegebenen anderen Benutzer aus

	alleWunschliste) hinzu und RETURN HTML buchtausch.de/benutzername/stöbern
3. Optionen: hinzufügen zur Wunschliste(für jeden Eintrag in Ergebnis-Liste)/back/close	
	8. checke, ob Eintrag vorhanden in Datenbank benutzername wunschliste Wenn Eintrag bereits vorhanden: confirmation „Eintrag bereits vorhanden, erneut hinzufügen?“ Wenn Eintrag noch nicht vorhanden: weiter mit 11. , oder GET und RETURN HTML , oder schließe Anwendung
9. Optionen: 'Ja, Eintrag erneut hinzufügen(Buch doppelt)', 'Nein, Eintrag nicht erneut hinzufügen'	
	10. Wenn ja: weiter Wenn nein: weiter mit 3.
	11. INSERT Einträge in Datenbank benutzername wunschliste und alleWunschliste und checke match und GET und RETURN HTML

Use case : Tausch anbieten

Benutzer	System
1. Tausch anbieten	
	2. Confirmation
3. Optionen: 'Ja, Tausch anbieten'/'Nein, lieber doch nicht'	
	4. Wenn nein: RETURN HTML Wenn ja: checke, ob Eintrag vorhanden Wenn Eintrag vorhanden: INSERT Eintrag in Datenbank benutzername angebot (beide Benutzer) und RETURN HTML Wenn Eintrag nicht vorhanden: notification „Buch steht leider nicht mehr zum Tausch zur Verfügung“ und RETURN HTML

Use case : Tausch annehmen

Benutzer	System
1. Tausch annehmen	
	2. Confirmation
3. Optionen: 'Ja, Tausch annehmen'/'Nein, lieber doch nicht'	
	4. Wenn Ja: checke, ob Einträge vorhanden Wenn ja: DELETE Einträge in Datenbanken benutzername wunschliste und alleWunschliste bzw. benutzername bibliothek und alleBibliothek (für beide Benutzer) und setze Status in Datenbanken Benutzername angebot auf angenommen (beide Benutzer) und hole Datensätze aus Datenbank Benutzerdaten und sende Email mit den benötigten Benutzerdaten an beide Benutzer und RETURN HTML Wenn Nein: notification „Buch steht leider nicht mehr zum Tausch zur Verfügung“ und RETURN HTML

Projektplan

siehe Projektplan3.pdf