Lager- und Bestandsverwaltungssystem

Pflichtenheft\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Version 1.0

**Inhaltsverzeichnis**

[1 Beschreibung der Projektidee 3](#_Toc92739470)

[1.1 Ausgangssituation 3](#_Toc92739471)

[1.2 Zielsetzung 3](#_Toc92739472)

[2 Muss- und Kann-Kriterien 4](#_Toc92739473)

[2.1 Muss-Kriterien 4](#_Toc92739474)

[2.2 Kann-Kriterien 4](#_Toc92739475)

[3 Use-Case Diagramme 5](#_Toc92739476)

[4 Projektplan 6](#_Toc92739477)

[5 Produktumgebung 6](#_Toc92739478)

[5.1 Definitiv enthalten: 6](#_Toc92739479)

[5.1.1 Visual Studio 6](#_Toc92739480)

[5.1.2 C# 6](#_Toc92739481)

[5.1.3 MSSQL 6](#_Toc92739482)

[5.1.4 ASP.NET Core 6](#_Toc92739483)

[5.1.5 Entity-Framework 6](#_Toc92739484)

[5.1.6 Caliburn.Micro 6](#_Toc92739485)

[5.2 Möglicherweise enthalten: 7](#_Toc92739486)

[5.2.1 SWAGGER 7](#_Toc92739487)

[5.2.2 Dapper 7](#_Toc92739488)

[5.2.3 Automapper 7](#_Toc92739489)

[5.2.4 xUnit 7](#_Toc92739490)

[5.2.5 MahApps.Metro 7](#_Toc92739491)

[6 GUI 8](#_Toc92739492)

[6.1 Login 8](#_Toc92739493)

[6.2 Bestand 8](#_Toc92739494)

[7 DB-Entwurf 9](#_Toc92739495)

[8 GIT-Repository 9](#_Toc92739496)

[9 Testplan 10](#_Toc92739497)

# Beschreibung der Projektidee

## Ausgangssituation

Das Programm ist für die Ausführung auf Windowsrechnern konzipiert und soll in einer Datenbank zentral verwaltet werden, um den Zugriff von mehreren Nutzern simultan zu gewähren.  
Mittels API-Controller wird die Authentifizierung einzelner Nutzer gewährleistet und dies als Erweiterungspunkt konzipiert, um eventuell zusätzliche Endpunkte für den Gebrauch der Software vorzubereiten, damit das Hinzufügen weiterer Zugangspunkte in der Zukunft einfacher umzusetzen ist.

Die Anwendung soll in einer Firmenumgebung als Lager und Bestandsverwaltungstool eingesetzt werden. Dies soll es ermöglichen Abgänge, sowie Zugänge zu Verbuchen und sämtliche Änderungen zu verwalten.

## Zielsetzung

Zielsetzung ist es den Arbeitsaufwand der Mitarbeiter zu verringern um Kosten zu sparen.  
Dies wird durch zentralisierte Speicherung der Daten im digitalen Format ermöglicht.  
Zudem soll durch eine Benutzerfreundliche Oberfläche ein einfaches Verwalten und Einsehen aller bestandbezogenen Informationen möglich sein.  
Durch die Zentralisierung wird der Zugriff von mehreren Endgeräten von verschiedenen Nutzern zur gleichen Zeit realisiert.  
Mittels Authentifizierung wird ebenfalls der Zugriff und die Berechtigungen der Arbeiter reglementiert und gegebenenfalls eingeschränkt, sofern dies in einzelnen Bereichen erforderlich ist.

Sämtliche Buchungen werden zeitlich festgehalten um vergangene Änderungen nachvollziehen zu können und belegen zu können.  
Ebenfalls erlaubt dies fehlerhafte Buchungen und mangelhafte Ware zurückzuverfolgen um die Integrität des Lagers/Lagerinhaltes zu gewährleisten.

# Muss- und Kann-Kriterien

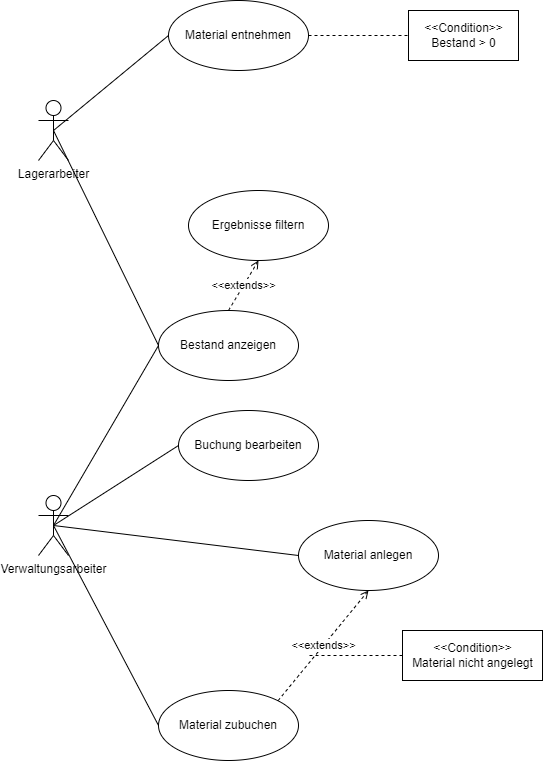
## Muss-Kriterien

|  |  |
| --- | --- |
| **Kriterium** | **Zeitaufwand** |
| Anmeldung mit Nutzername und Passwort | 5 Stunden |
| Abrufen des aktuellen Bestands und Ergebnisse filtern | 30 Stunden |
| Zu- und Abbuchung einzelner Positionen | 20 Stunden |
| Anlegen neuer Materialien | 20 Stunden |
| Korrektur von bereits abgeschlossenen Buchungen | 40 Stunden |

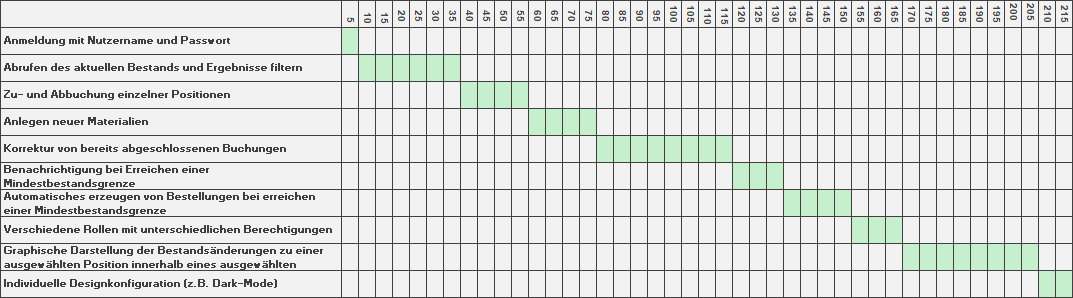
## Kann-Kriterien

|  |  |
| --- | --- |
| **Kriterium** | **Zeitaufwand** |
| Benachrichtigung bei Erreichen einer Mindestbestandsgrenze | 15 Stunden |
| Automatisches erzeugen von Bestellungen bei erreichen einer Mindestbestandsgrenze | 20 Stunden |
| Verschiedene Rollen mit unterschiedlichen Berechtigungen | 15 Stunden |
| Graphische Darstellung der Bestandsänderungen zu einer ausgewählten Position innerhalb eines ausgewählten Zeitraums | 40 Stunden |
| Individuelle Designkonfiguration (z.B. Dark-Mode) | 10 Stunden |

# Use-Case Diagramme



# Projektplan



# Produktumgebung

## Definitiv enthalten:

### Visual Studio

Die Entwicklungsumgebung von Microsoft in der der Quellcode geschrieben wird

### C#

Eine moderne, objektorientierte und typsichere Programmiersprache, die Entwicklern das Erstellen zahlreicher sicherer und robuster Anwendungen, die in .NET ausgeführt werden ermöglicht.

### MSSQL

Microsoft-Datenbank-Server

### ASP.NET Core

Ein Webframework zum Erstellen von RESTful-HTTP-Diensten

### Entity-Framework

Entity Framework Core ist ein moderner Objekt-Datenbank-Mapper für .NET.

### Caliburn.Micro

Ein Framework designt für alle XAML-Plattformen mit starker Unterstützung von MVVM Entwurfsmustern

## Möglicherweise enthalten:

### SWAGGER

Sammlung von Open-Source-Werkzeugen, um HTTP-Webservices zu entwerfen, zu erstellen, zu dokumentieren und zu nutzen.

### Dapper

Dapper ist ein simpler Micro-ORM mit sehr hoher Performance

### Automapper

Eine simple Bibliothek, die das Mappen von einem Objekt zu einem anderen ermöglicht

### xUnit

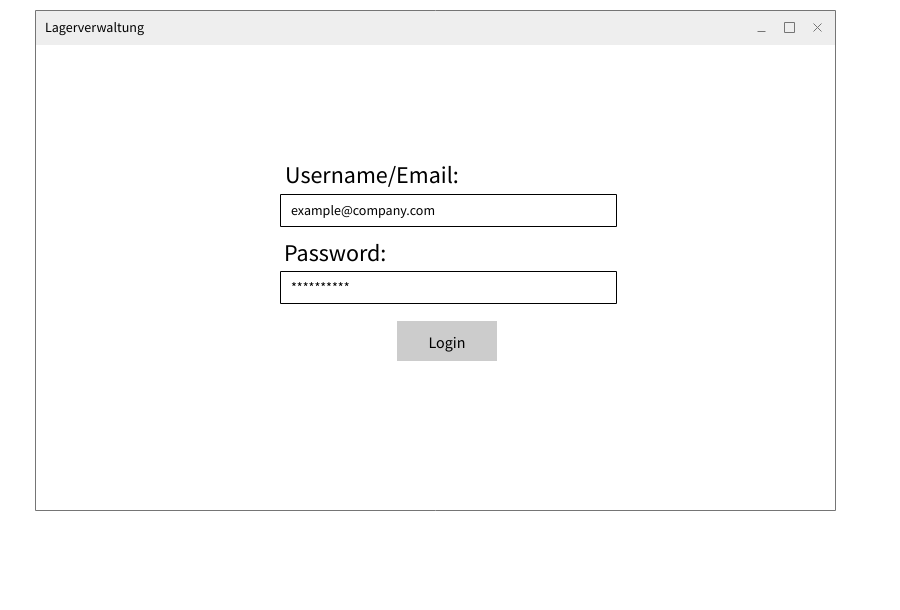
Ein kraftvolles weit verbreitetes Unit-Test-Framework

### MahApps.Metro

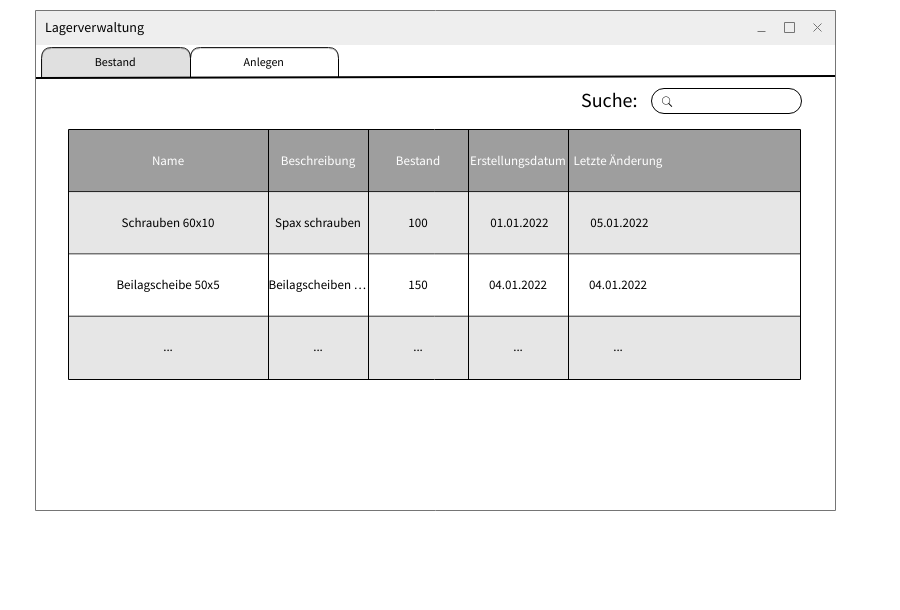
Ermöglicht Entwicklern eine Anwendung im Metro bzw. Modern-UI Stil zu designen.

# GUI

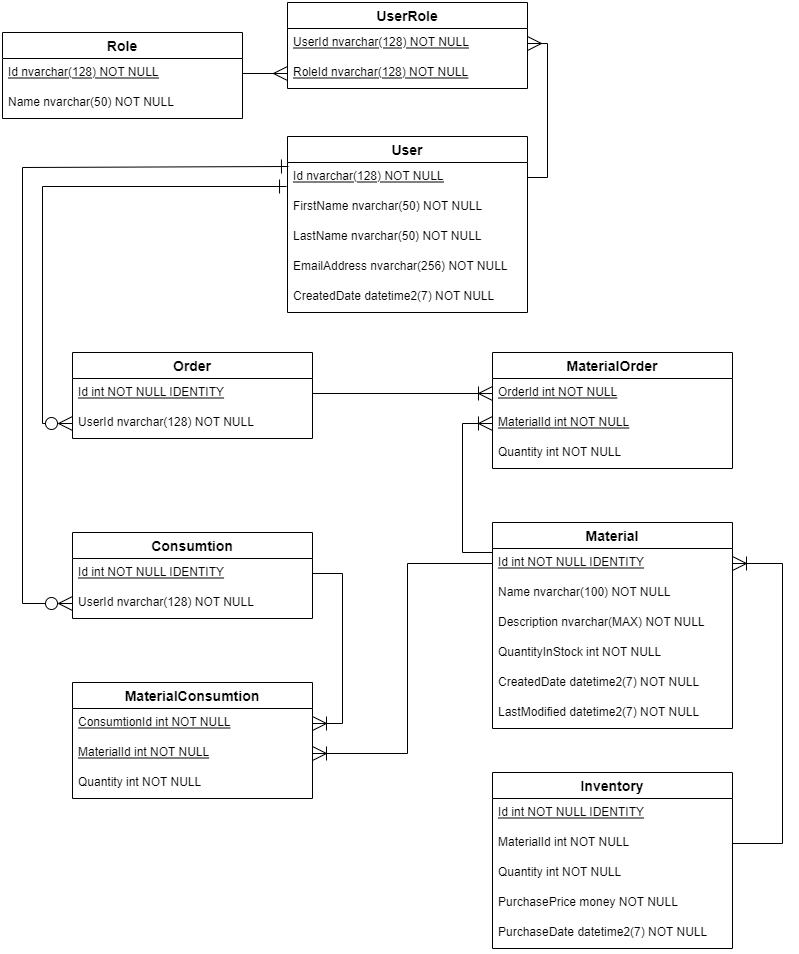
## Login



## Bestand



# DB-Entwurf



# GIT-Repository

<https://github.com/AdrianLion/Lager-und-Bestandsverwaltung>

# Testplan

|  |  |
| --- | --- |
| ID: | T01 |
| Beschreibung:  Vorbedingung:  Test-Schritte:  Erwartetes Resultat: | Einloggen  Das Programm ist im Anmeldedialog  1. Im Feld “Username/Email” wird die Email eines Users eingegeben.  2. Im Feld “Password” wird ein passendes Passwort eingegeben.  3. Der Login wird durchgeführt.  Der User ist eingeloggt und die Hauptansicht wird geladen |
| ID: | **T02** |
| Beschreibung:  Vorbedingung:  Test-Schritte:  Erwartetes Resultat: | Filtern des angezeigten Bestands  Der User ist eingeloggt  Im Suchfeld wird ein Artikel eingegeben  Die Produktansicht wird nach dem Suchkriterium eingeschränkt |
| ID: | **T03** |
| Beschreibung:  Vorbedingung:  Test-Schritte:  Erwartetes Resultat: | Entnahme von Material aus dem Lager  1. Der User ist eingeloggt  2. Der User besitzt die nötigen Rechte  3. Der Bestand ist ausreichend  1. Das zu entnehmende Material wird ausgewählt  2. Die Menge wird angegeben  3. Die Entnahme wird durchgeführt  Der Bestand verringert sich um die angegebene Menge |

|  |  |
| --- | --- |
| ID: | T04 |
| Beschreibung:  Vorbedingung:  Test-Schritte:  Erwartetes Resultat: | Neues Material anlegen  1. Der User ist eingeloggt  2. Der User besitzt die nötigen Rechte  3. Das Material ist noch nicht vorhanden  1. Die benötigten Informationen über das Material werden eingetragen  2. Das Material wird angelegt  Das neue Material wird dem Lager hinzugefügt |
| ID: | **T05** |
| Beschreibung:  Vorbedingung:  Test-Schritte:  Erwartetes Resultat: | Material zubuchen  1. Der User ist eingeloggt  2. Der User besitzt die nötigen Rechte  3. Das Material ist vorhanden  1. Das zu verbuchende Material wird ausgewählt  2. Die Menge wird angegeben  3. Die Buchung wird durchgeführt  Der Bestand erhöht sich um die angegebene Menge |