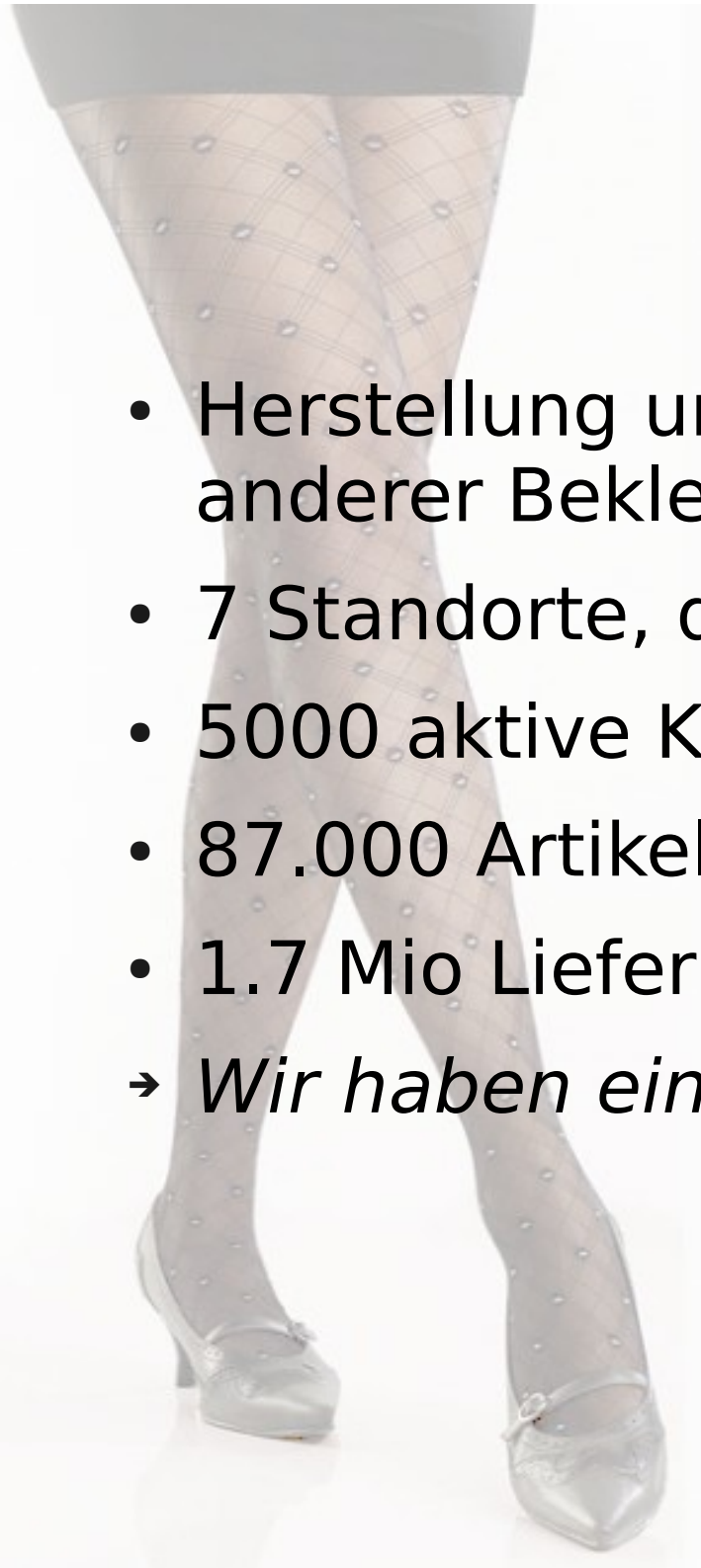
Esdarp

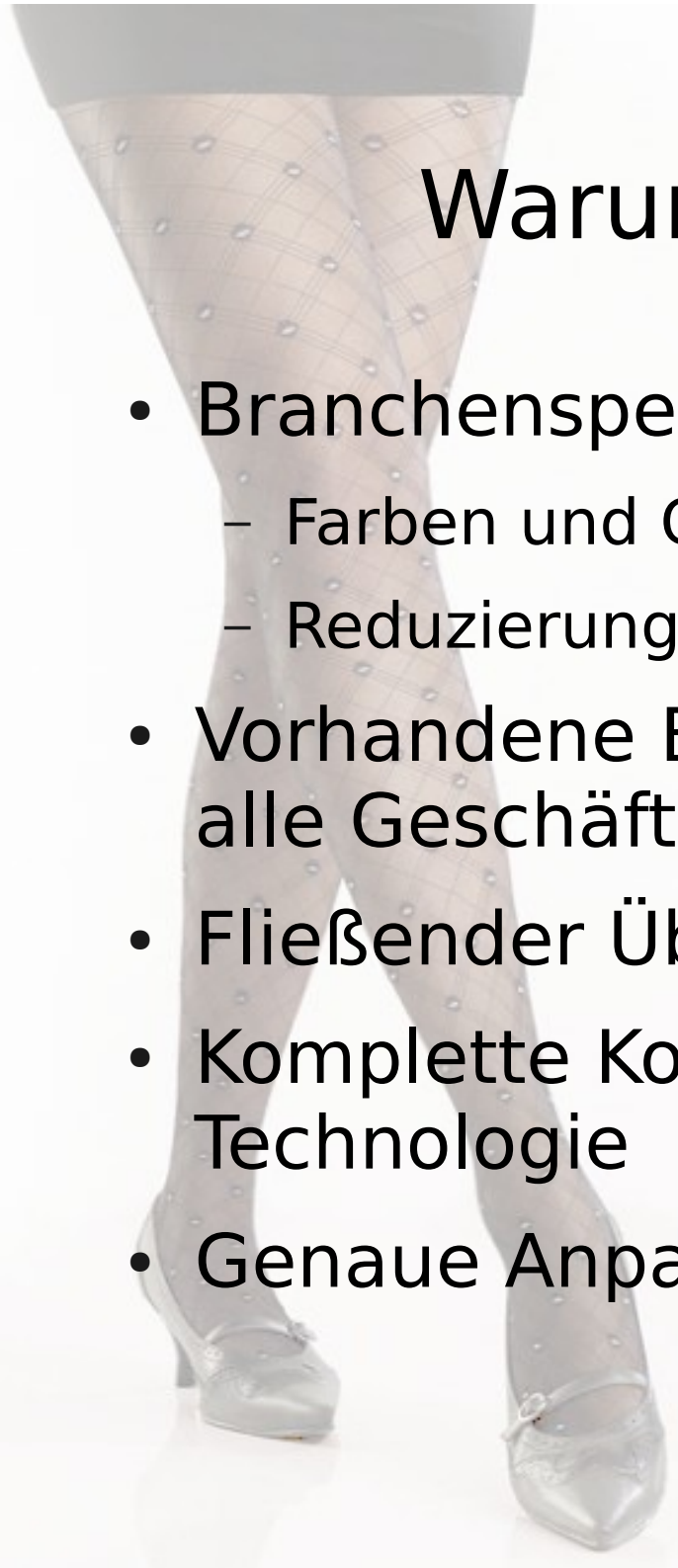
Esda
ROGO

Inhouse-Warenwirtschaft der Esda-Gruppe wird OpenSource



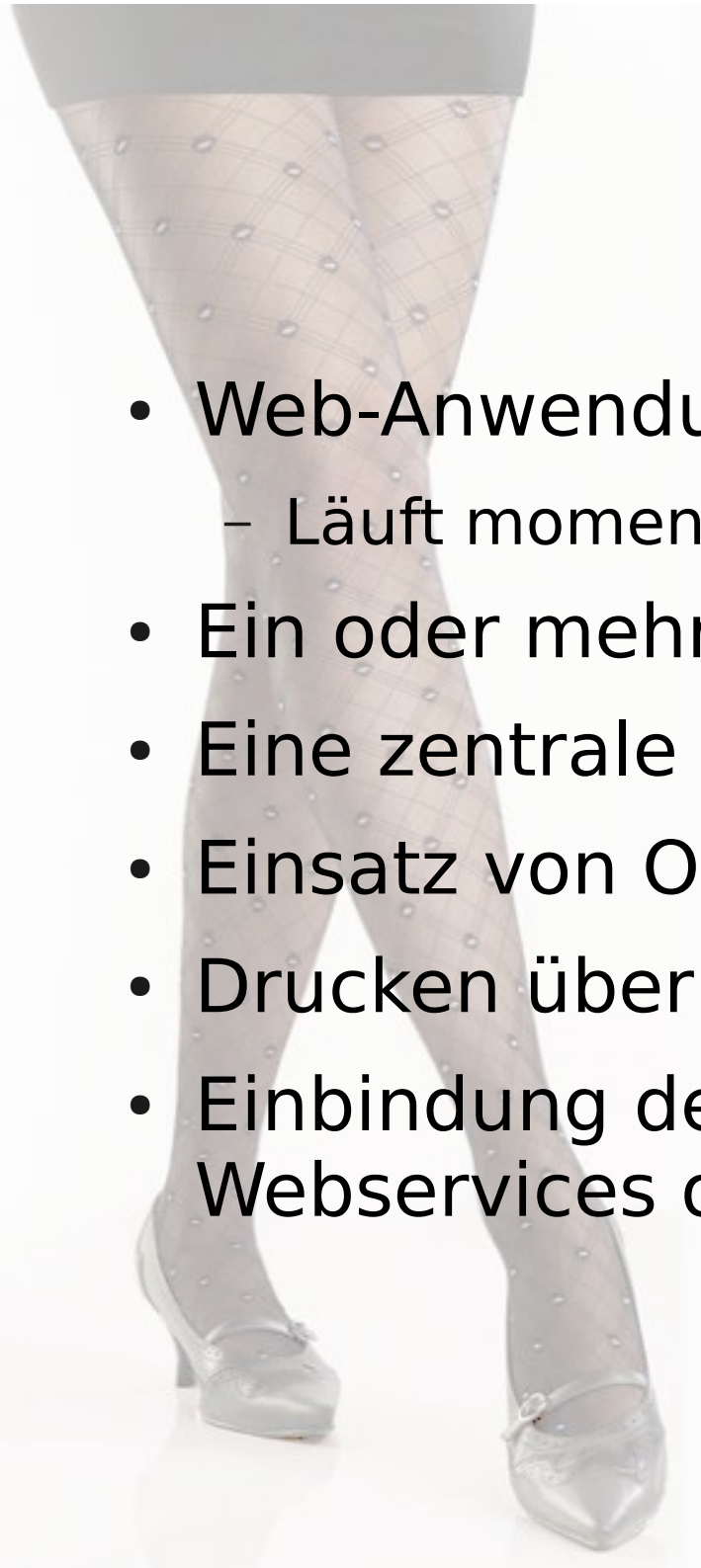
Esda-Gruppe

- Herstellung und Vertrieb von Strumpfwaren und anderer Bekleidung
- 7 Standorte, davon 5 in Deutschland
- 5000 aktive Kunden
- 87.000 Artikel
- 1.7 Mio Lieferpositionen in 2008
- ➔ *Wir haben ein Mengenproblem!*



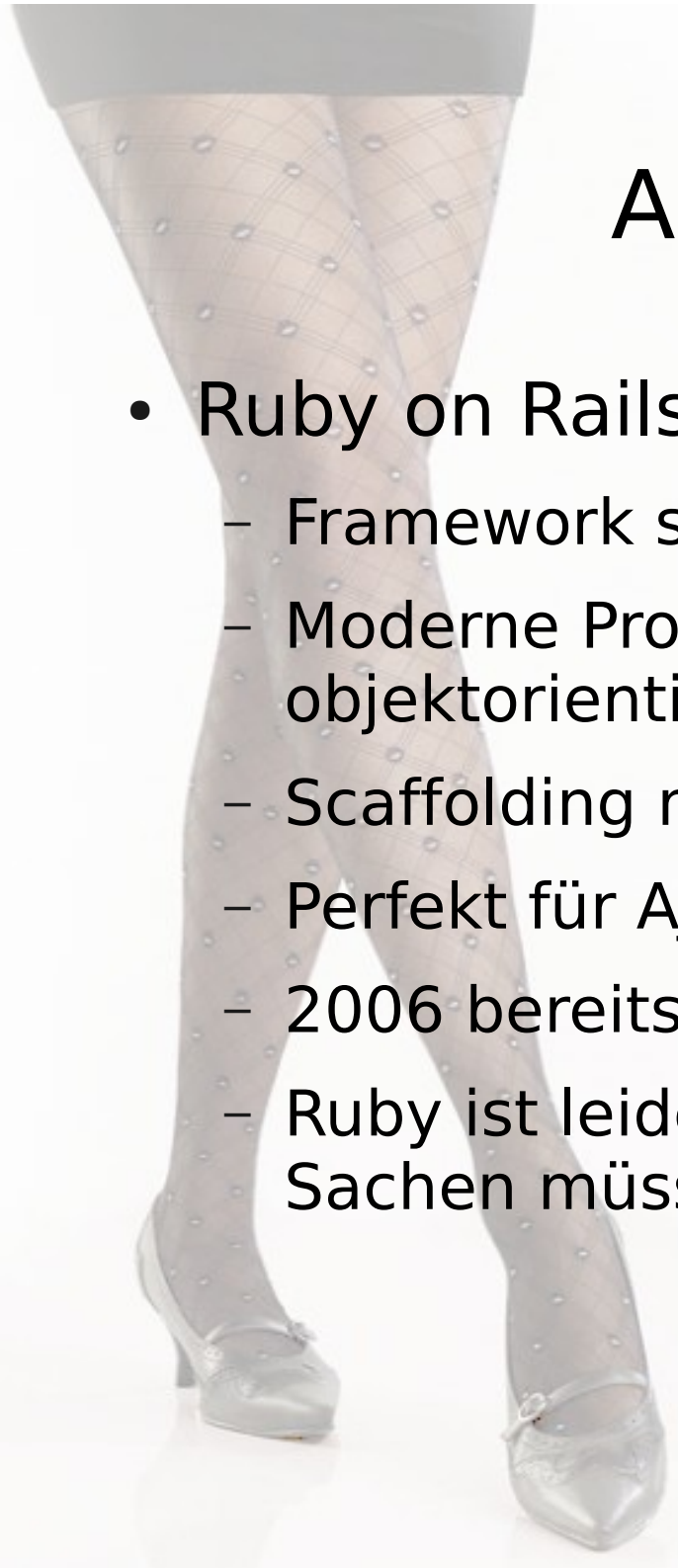
Warum Eigenentwicklung?

- Branchenspezifische Lösung nötig
  - Farben und Größen
  - Reduzierung der Anzahl Datenfelder
- Vorhandene Branchenlösungen decken nicht alle Geschäftsfälle und Produktionstiefe ab
- Fließender Übergang von Altanwendungen
- Komplette Kontrolle über Auswahl der Basis-Technologie
- Genaue Anpassung an eigene Prozesse



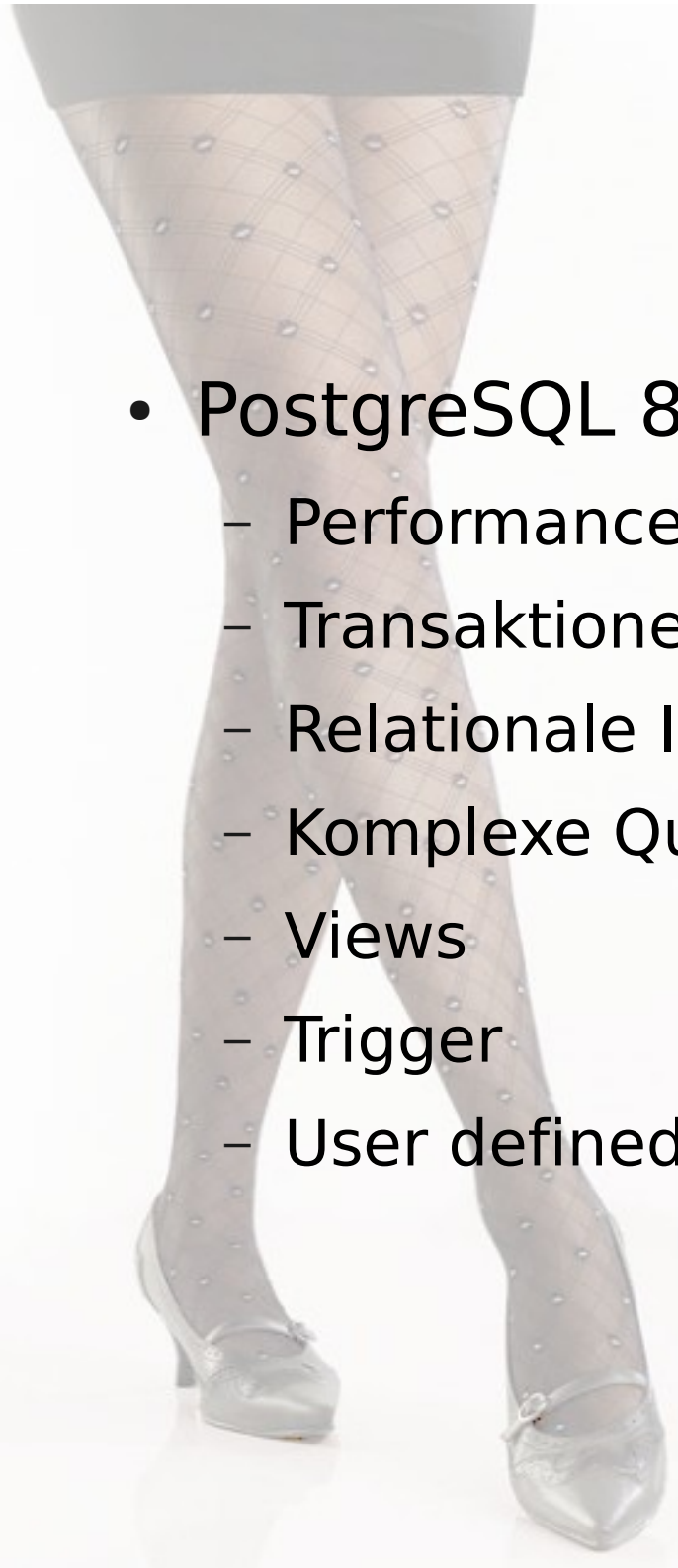
Konzept

- Web-Anwendung, keine Client-Installation
  - Läuft momentan nur mit Firefox 1.5 bis 3.x
- Ein oder mehrere Application-Server
- Eine zentrale Datenbank
- Einsatz von Opensource-Programmen
- Drucken über PDF
- Einbindung der Alt-Anwendungen über Webservices oder DB-Replikation



Application Server

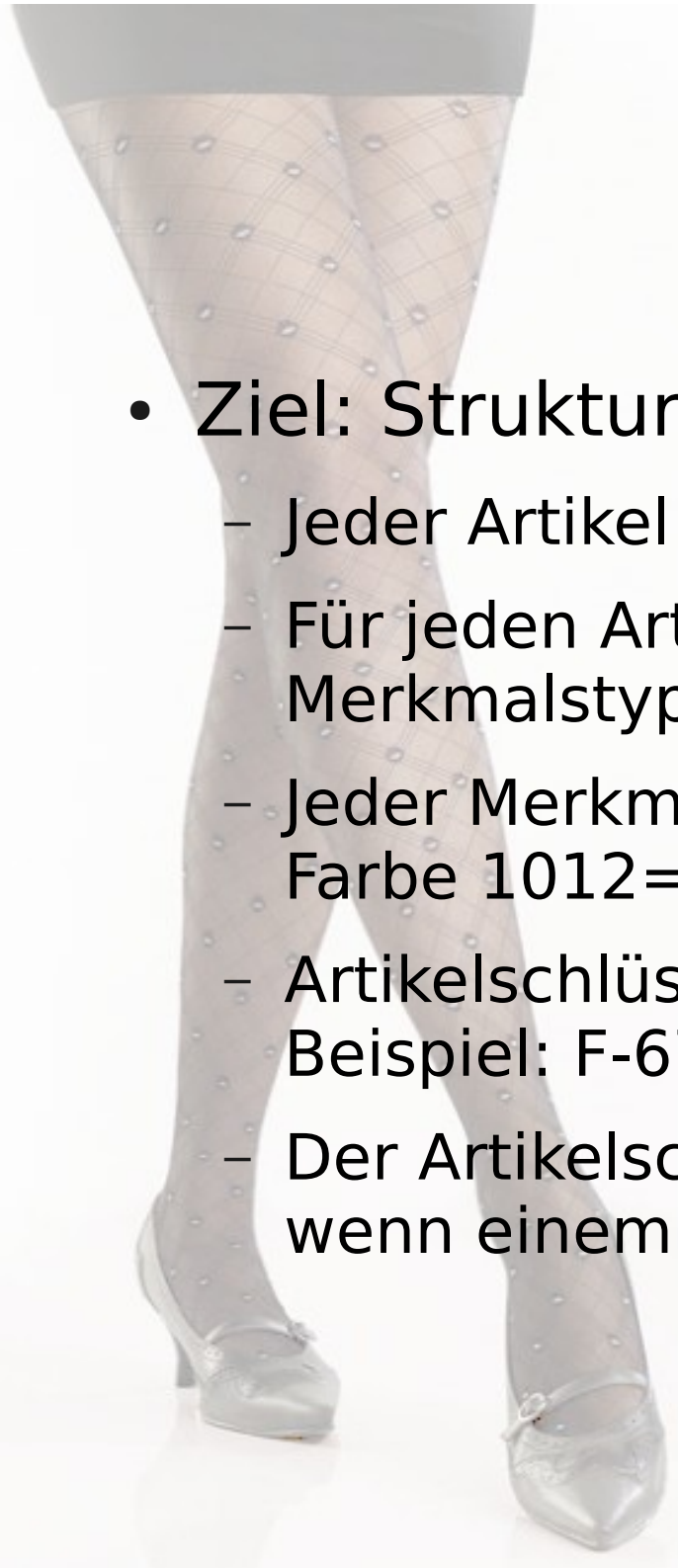
- Ruby on Rails – Anwendung
  - Framework spart viel Arbeit
  - Moderne Programmiersprache, durchgängig objektorientiert
  - Scaffolding mit eigenen Erweiterungen
  - Perfekt für AJAX geeignet
  - 2006 bereits verfügbar
  - Ruby ist leider recht langsam – performance-kritische Sachen müssen in die Datenbank



Datenbank

Esda ROGO

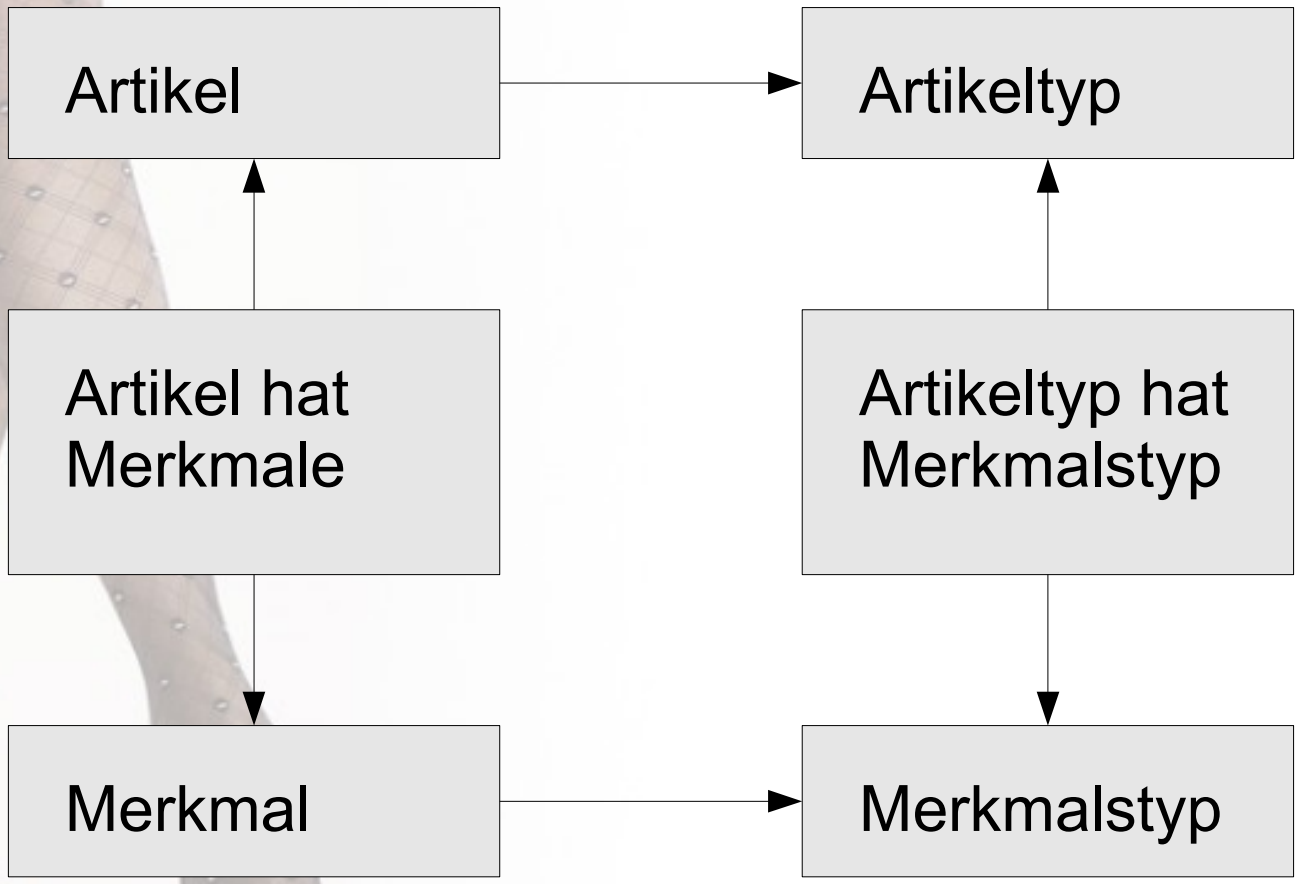
- PostgreSQL 8.2
  - Performance
  - Transaktionen
  - Relationale Integrität
  - Komplexe Queries
  - Views
  - Trigger
  - User defined Functions



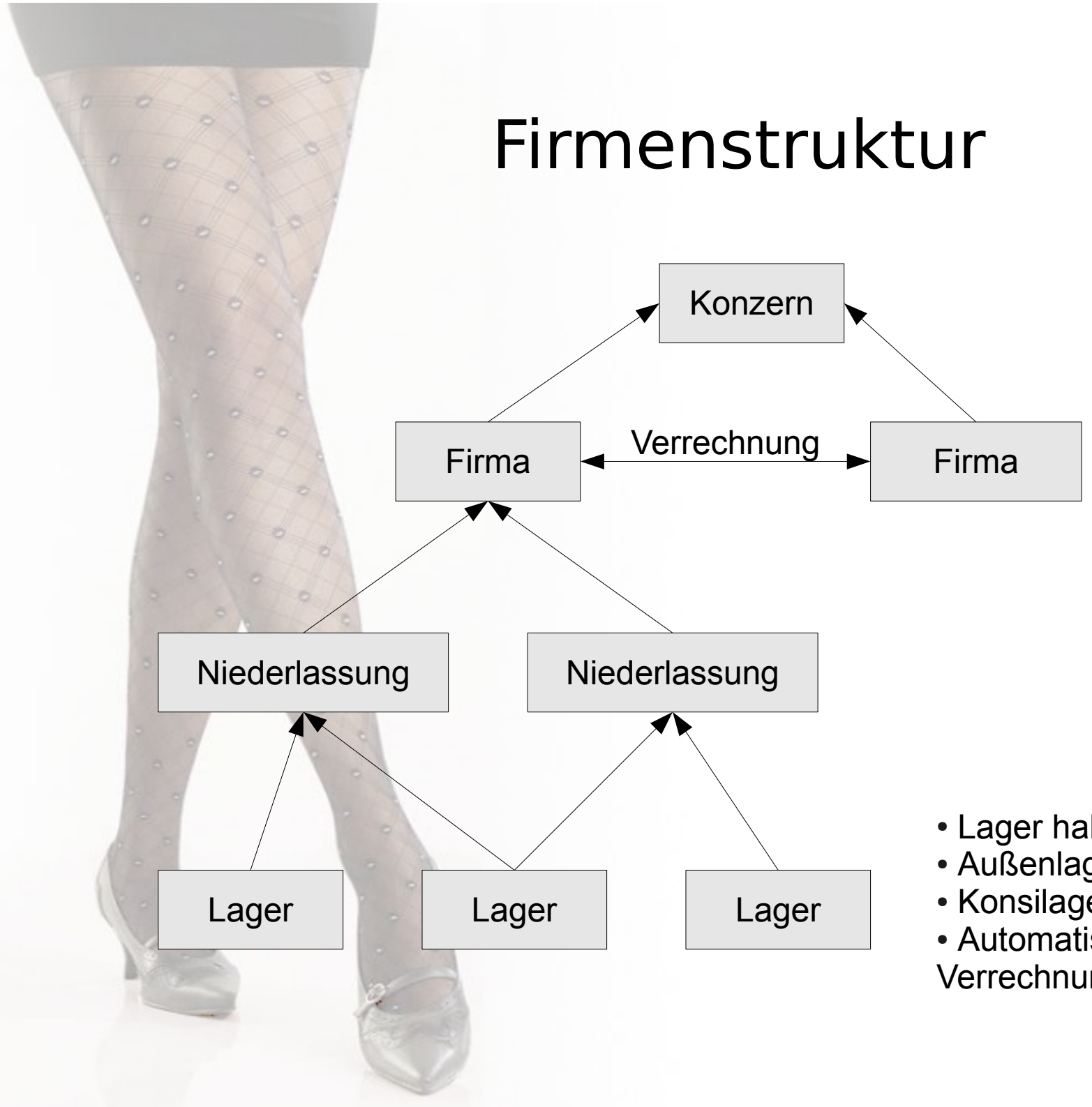
Artikelstamm

- Ziel: Struktur und Flexibilität
  - Jeder Artikel gehört zu einem Artikeltyp
  - Für jeden Artikeltyp gibt es eine geordnete Liste von Merkmalstypen, z.B. Produkt-Farbe-Größe
  - Jeder Merkmalstyp hat viele Ausprägungen, z.B. Farbe 1012=schwarz oder Größe 001=38/40
  - Artikelschlüssel: Typkürzel-Merkmal1-Merkmal2... Beispiel: F-674001Ü-1012-001
  - Der Artikelschlüssel wird durch DB-Trigger gebildet, wenn einem Artikel alle Merkmale zugeordnet sind

Datenstruktur Artikelstamm



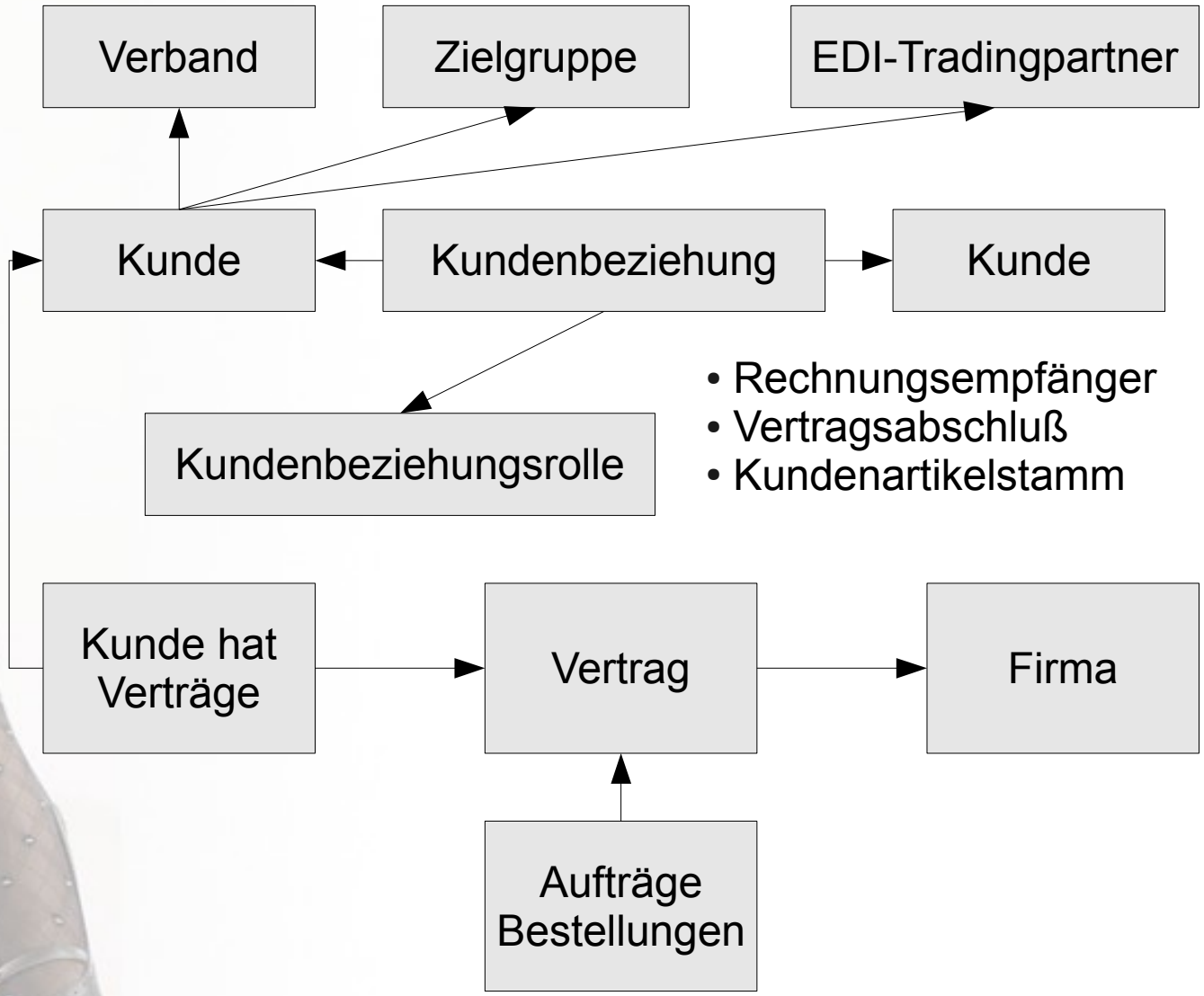
Firmenstruktur



- Lager haben Eigentümer
- Außenlager
- Konsilager
- Automatische Verrechnung

Kunden/Lieferanten

Esda ROGO



Geschäftsfälle

- NOS (never out of stock)
  - Kunde legt Mindestbestandsmengen fest
  - Kurzfristige Lieferung auf Abruf
- Konsignation
  - Lieferung an Konsilager
    - Order vom Kunden
    - Eigene Disposition
  - Abrechnung der Entnahme aus Konsilager
    - Abverkaufsmeldung -> Rechnung
    - Gutschriftsverfahren
    - mit/ohne automatischer Nachorder

Geschäftsfälle 2

- Cross Docking
  - Lieferung an ein oder mehrere zentrale Lager des Kunden
  - Verpackung und Auszeichnung für Warenendempfänger (Kunden-Filiale)
  - mit/ohne Konsi
- DESADV mit NVE (Nummer der Versandeinheit)
  - Erfassung der Packstücke
  - Vergabe eindeutiger NVE, Barcode EAN-128
  - Elektronischer Lieferschein per EDI

Geschäftsfälle 3

- Vorgepackte Colli
  - Fertige Packstücke im Lager
  - Zuordnung zu Lieferscheinen
- Teillieferungen
  - Aus mehreren Niederlassungen, wenn der Lagerbestand in einer NL nicht reicht
- Sammelrechnung
  - Rechnung je Teillieferung, je Auftrag oder zusammengefaßt nach Zeitraum, Vertrag und Rechnungsempfänger

Realisierung

Esda ROGO

- MVC-Architektur a la Rails
- Eigenes Scaffolding – extrem wenig Code
- Zuerst DB-Struktur, danach Klassen/Methoden
- Sorgfältige Validierung
  - Im Browser per Javascript
  - Im Code, vorrangig in Model-Klassen
  - In der Datenbank mittels referentieller Integrität, Trigger und Kapselung in Funktionen
  - Doppelte Lagerbuchhaltung
- Alles transaktionsgesichert

Beispielcode

app/models/waehrung.rb

```
require "stammdaten"
class Waehrung < ActiveRecord::Base
    include Stammdaten
    include ScaffoldingPrivileges
    set_primary_key "waehrung_id"
    has_many :preislisten
    has_many :waehrungsumrechnungen, :class_name => \
        'Waehrungsumrechnung', :foreign_key => 'zielwaehrung_id'

    # Validators
    validates_presence_of :bez, :kuerzel
end
```

app/controllers/waehrung.rb

```
require "stammdaten"
class WaehrungController < ApplicationController
        include Stammdaten
        scaffold :waehrung
end
```

Weitere Komponenten

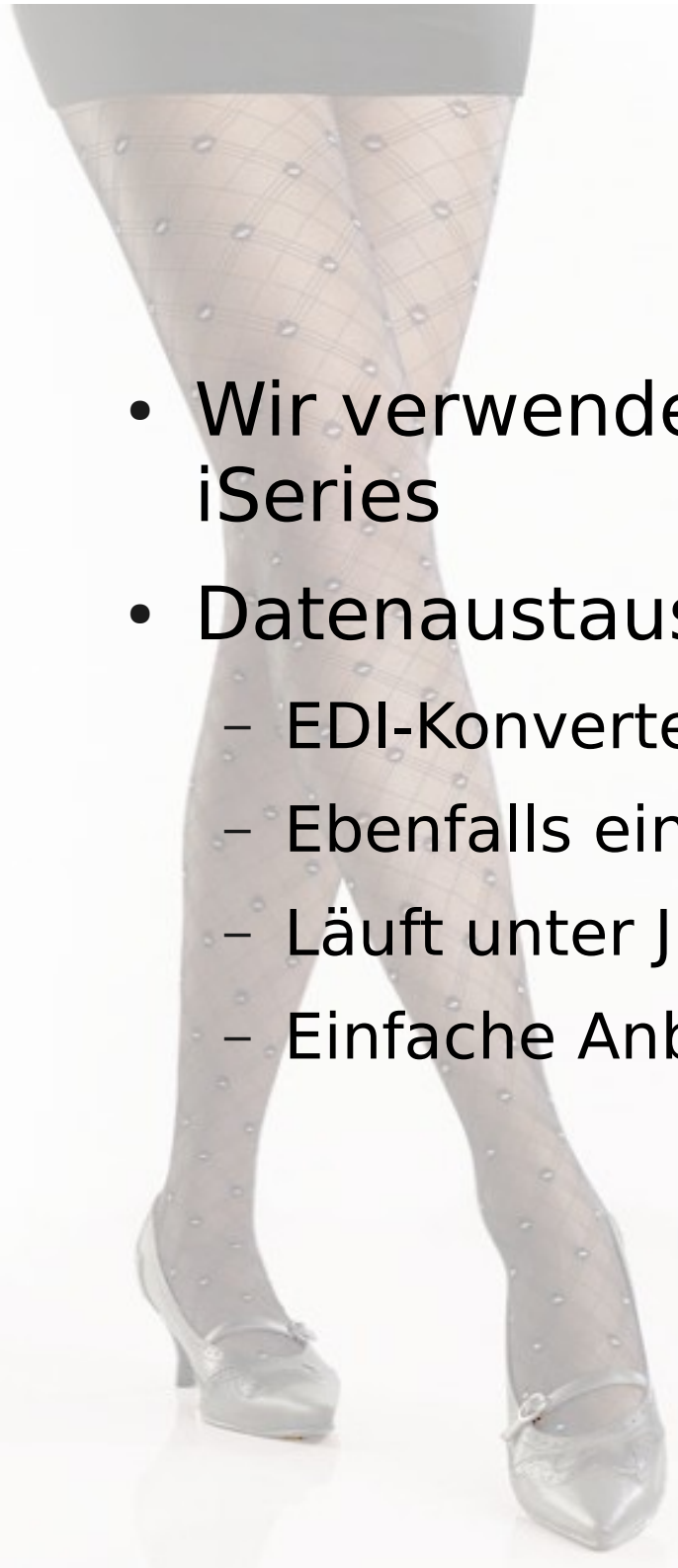
Esda ROGO

- Single Sign On
  - Funktioniert mit Kerberos 5 und Firefox
  - Alternative Passwort-Prüfung gegen POP-Server
- Reporting
  - JfreeReport in eigenem Servlet in Tomcat
  - Pentaho Report Designer zum Entwurf
  - Zusätzlich seit einigen Wochen JasperReports und iReport, da mehr Funktionalität
- Auswertungen als OpenOffice-Spreadsheet
  - Ruby-Programm schreibt direkt OpenDocument
- Vertriebs-Controlling mittels Palo

Buchhaltung

Esda ROGO

- Esdarp beinhaltet keine Buchhaltung
- Bei Esda:
  - Rechnungswesen A.REWE der Ametras Consult GmbH auf IBM iSeries (früher AS/400)
  - Esdarp hat Schnittstellen dorthin
  - Im Moment nicht geklärt, ob diese auch OpenSource werden.



EDI

- Wir verwenden EDI-Konverter Trusted Link auf iSeries
- Datenaustausch über extra SOAP-Server
  - EDI-Konverter prinzipiell austauschbar
  - Ebenfalls eine Rails-Anwendung
  - Läuft unter Jruby, der Java-Version von Ruby
  - Einfache Anbindung der iSeries-Datenbank über JDBC

ToDo

- Modularisierung
  - Ausgliederung firmenspezifischer Funktionalität in eigene Module
- Internationalisierung
  - Benutzeroberfläche nur deutsch
  - Bezeichner sind überwiegend in deutsch
  - Deutsches bzw. EU-Steuerrecht
- Portierung auf neuere Rails-Versionen
  - Setzt im Moment Rails 1.2.x voraus und bringt dieses mit

Wie weiter?

Esda ROGO

- OpenSource-Release etwa Mitte 2009
  - Vorher Heraustrennung von Esda-Spezifika
  - Ziel: mehr Entwickler – mehr Funktionen
  - Umstellung von svn auf git
  - Eigene Buchhaltung oder Schnittstellen zu anderen
  - Kostenplanung und Kostenrechnung
- Zielrichtung bei Esda
  - Technologie und Kalkulation
  - Produktionsplanung und -steuerung