Company Structure

Company / Unit	Location	Explanation
Hubert Burda Media KG	Head quarters in Offenburg, Germany but also in Munich and many other places across Europe.	Holding company of v many magazines and
Cyberport GmbH	Dresden, Germany	Daughter company of customers. The core p
CyberSolutions GmbH	Munich, Germany	Was founded as a subc Computer Universe (p
CyberSolutions Tech	Cluj-Napoca, Romania	Romanian subdivision outsourcing timeshare
Computer Universe GmbH	Friedberg, Germany	Computer Universe ht Cyberport, although b

Team Structure

Status Dec 2022, if this doesn't reflect reality, feel free to update this page.

Resource Type	Pricing Team	Sourcing Team	Integration Team	Other
PO (Lead: Hannah Winnes)	Aleksander Meier (DE)	Justus Mulli (DE)	Ana Maria Coman (RO)	Timea Barth
Backend Development Team (Lead: Horatiu Maiereanu)	Lead: Nicolae Natrapeiu (RO) Additional members: see PS Team Information	Lead: Peter Horvath (RO) Additional members: see DFS Team Information	Lead: Mihai Mocanu (RO) Additional members: see https://cyber-solutions.atlassian.net/wiki/spaces/IS/pages/553058385	no dedicated
Frontend Development Team (Lead: Sebastian Rus)	-	-	see EC Team Information	-

QA Team (Lead: Mircea Talau)	Lead: Vasile Cotofan (RO)	Lead: Mircea Talau (RO)	Lead: Virgil Trif (RO)	no dedicated
Design Team (Lead: Andrei Moldovan)	-	-	Andrei Moldovan, Madalina Merlas	-
DevOps (Lead: Florin Lupean)	see CST organizational chart			
Architect	Cristi Scoarta (RO)			

Frequently used Abbreviations

Frequently used abbreviations can be found in our Acronym Glossary - feel free to extend it during the onboarding if you find acronyms you don't know the meaning, just add it and leave the explanation empty.

Required Permissions

Who?	You should have access to:
Every team member	• Cybersolutions email address • Bitbucket / Jira / Confluence • Lucidchart • ECOMM frontends • AWS console • SSH keys to access data bases • New relic • ELK • Microservice data bases (passwords) • Argo • Metrics influx • Django Admin (only if relevant)

	• Cst vacation calendar in Outlook • Timetastic - for people in Cluj
	• Teams Channels
	• ARIS
	• Concourse
Developers	
	• Read access Exasol
	• Admin to the CS Confluence + Jira (copy PO's permissions if possible)
	• Cyberport Cybertracker • Cyberport Confluence - devconfluence • Cyberport Jira - devjira • Optional: Cyberparts
POs	• Write access Exasol • Tableau Dashboards
	• Optional for German speaking only: IT-Störungs-Mails (Incident mails) in Exasol and other external systems
	• Optional for employees in Germany only: Cytric Travel & Expense Tool

We are following a Scrum or Kanban process, depending on the team with 2 weeks sprints during which we have daily stand-ups, bi-weekly backlog refinement(grooming), bi-weekly sprint reviews(demos), retrospectives and planning meetings.

Our company focuses on creating micro services for e-commerce companies. These services are split in three streams: **Pricing, Sourcing** and **Integration Services.**

An overview of all our services and how they interact can be seen here: CST Services

The CyberSolutions Tech team handles the entire life-cycle of these projects:

more waiting for us on the roadmap ahead: Global Backlog Microservices + Data Science.

Our projects are designed as products that can serve multiple clients, even that in the current phase they are targeted for two companies from the Burda group: Cyberport and Computer Universe.

Pricing

Pricing Service (PS)

The Pricing Service imports product data, competitor and configuration information from the Data Integration Service, Price Feed Service and E-Commerce Cloud Services and calculates the prices of the product portfolio using a variety of strategies. Cyberport can use it to continuously calculate optimized prices for their webshops according to the current market situation in order to optimize their profit. Documentation can be found here Pricing Service (PS)

Calculating the price of a product in the Pricing Service is done in four steps:

1. Collection of product information from different sources (including for example information about stock, purchase price, shipping, configurations, ...)
2. Selection of the correct pricing strategy (called "price stream") evaluating the data in 1.
3. Calculating the price using the chosen stream in 2. and the data in 1.
4. Run the price through price quality checks (price change not too high, not too many price changes in a row).
5. Distributing of the price to relevant destinations (currently Exasol and Cyberparts via an FTP Server)

Price Search Engine Voucher (PSE)

The **PSE Voucher** is a microservice that computes a voucher on price search engines (PSE) for a product, based on a set of business rules. The documentation can be found here PSE Voucher Service (PSE)

- A sample of the voucher service api and how one of the clients interact with it: Computer Universe
- git repo here: https://bitbucket.org/devcst/pse-voucher/

Market Price Screening (MPS)

Several services like Pricing Service (*Auto-Pricing)* or *Price Search Engine Voucher Service* need competitor price information. For this reason, price feeds from various providers are imported on a regular basis within the Market Price Screening (MPS). It does the following:

- collects data from a given price feed data source e.g. Geizhals via http and WorkIT
- normalize and persist the data in database

- notifies PSE Voucher Service and Pricing Service when updates are available (delta changes)
- Provides this data to these and any other services via an API.
- api docs: https://market-price-screening.service.cybersolutions-tech.com/api/docs/
- git repo here: https://bitbucket.org/devcst/marketplace-pricing-service/

More details about the service here: Market Price Screening (MPS)

Marketplace Service (MS)

Th
e Marketplace Service (MS) is a product for the companies that are using Amazon Marketplace to sell their products. The purpose of the project is detailed here: Marketplace Service (MS)

The service will comprise multiple services as in this MS Component Diagrams.

Marketplace Feed Service

- is using the AMWS (Amazon Marketplace Web Services) API, more exactly: Amazon Marketplace Endpoints and Operations, to get and the needed data and store it in a Postgres database.
- the service is polling a SQS queue for notifications from AMWS, for any changes occurring in an offer for one of the products listed by a client. These messages are parsed and saved in the database.
- data model: MS Feed Data Model
- after saving the all the new offers in the database the service sends a notification to the Marketplace Price Calculation Service through a SQS FIFO queue. The service exposes the data through a rest API.
- api docs: https://amazon-feed.service.cybersolutions-tech.com/api/docs/
- git repo: https://bitbucket.org/devcst/amazon-feed-service/

api docs: http://amazon-feed.stg.cybersolutions-tech.com/api/docs/

Marketplace Pricing Service

- is currently under development. Based on notifications coming from either AMWS Feed Service(through a SQS message) or Data Integration Service(through a SNS notification) this service will recalculate the price for a product based on an algorithm: MS Pricing Algorithm.
- the new prices will be stored in a Postgresql database and exported as CSV files, hourly and distributed to an external service.
- git repo: https://bitbucket.org/devcst/price-calculation-service/

Sourcing

Sourcing MicroServices

Demand Forecasting & Auto-Ordering

The **Demand Forecasting** project is currently just a small component of the **Auto-Ordering** project and consists only in a SQL query. Plans are to develop **Demand Forecasting** as a separate service with more elaborated algorithms and models.
Auto-Ordering architecture here: AutoOrdering describes the project's dependencies on the **Data Warehouse** (named Exasol in the diagram) and **CST Data Service**.
The **auto-ordering** algorithm relies on two kinds of data: historical data for calculating certain values (lead time, drop shipment and safety stock) and real-time data for getting the stock information. The historical data is fetched from the **Data Warehouse**, while the real-time data is fetched from the **Data Integration Service** (also referred to as Integration Service), even though in the future we hope to be able to fetch all the data through Data Integration Service.

A description of the terms used in AutoOrdering algorithm can be found here: Sourcing Glossary.

Demand Forecasting:

- api docs: https://demand-forecasting.service.cybersolutions-tech.com/api/docs/
- git repo: https://bitbucket.org/devcst/demand-forecasting/

Auto-Ordering:

- git repo: https://bitbucket.org/devcst/auto-ordering
- data model: ReOrder(Auto-Ordering) Database Model
- api docs: http://auto-ordering.service.cybersolutions-tech.com/api/docs/

Integration Services (projects used by both Pricing and Sourcing micro services)

There are three projects used by both streams: **Data Integration Service**, **Ecommerce Cloud**, and **Data Historization Service.**

Data Integration Service:

- receives data from Cyberport SAP PRO and Data Warehouse(Exasol), saves it to the database and exposes it through an API. Currently it receives 5 types of data: general product info, hierarchical product/categories info called Item info, product prices, product stock and product shipping data.
- data model: Data Integration Service Data Model
- API docs: https://data-integration.service.cybersolutions-tech.com/api/docs/

- git repo: https://bitbucket.org/devcst/data-integration/

Ecommerce Cloud

Is part of the **eCommerce Cloud** web application, an application where our customers will configure the Pricing and Sourcing microservices, as well as having access to interactive dashboards.

The **Pricing Service** needs a complex configuration. To help with this configuration we are developing a web application where users can create, edit and delete these type of configuration.

- API docs as they're implemented (staging version): https://ecomm-cloud.stg.cybersolutions-tech.com/api/docs/
- data model: EC Data Model
- UI staging: https://app-staging.cybersolutions-tech.com/pricing

Details about the service here: E-commerce Cloud (EC)

The big picture diagram of all our services can be seen here: https://app.lucidchart.com/documents/edit/8f3a76d4-684a-4f1b-8fde-acbc265c39f3/

Data Historization Service

All microservices historize their data in the Data Historization Service (DHS).

Technology stack

Frontend Development Technology Stack

Python Development Technology Stack

Technology Stack - Cloud Infrastructure & DevOps

So far, all our projects are developed using:

- Flask, Sqlalchemy, Pandas and Django
- Postgresql
- Elasticsearch

Our tests are running either in Bitbucket Pipelines or Concourse CI pipelines.

All our services are deployed in Kubernetes, using Concourse CI: https://concourse.ops.cybersolutions-tech.com/.

Our infrastructure lives in Amazon Web Services:

- AWS EKS for running the Kubernetes clusters on EC2 instances.
- AWS RDS for our Postgresql databases

- AWS SQS message queues
- AWS SNS topics
- AWS S3 buckets
- AWS Cognito for authentication and authorization

All the logs are stored in ELK: https://service.eu.sumologic.com/ui and we are monitoring our apps with NewRelic: https://rpm.eu.newrelic.com/accounts/2666361.