POLITECNICO DI TORINO, TELECOM PARIS

Master Double's Degree in Software Engineer/Data Science AI Image





Master Double's Degree Thesis

Life Cycle Assessment in Corporate Sustainability: Principles, Regulations, and Methodological Implementations

Supervisors

Prof. LUCA ARDITO

Prof. CHARLOTTE LACLAU

Candidate

LUCA ERBÍ

October 2025

Summary

Climate change and sustainability have become central concerns in both policy and corporate strategy. As regulatory frameworks and public expectations evolve, companies are under increasing pressure to measure, report, and reduce their environmental impact. Life Cycle Assessment (LCA) has emerged as a standardized method to evaluate the environmental effects of products, services, and processes throughout their entire life cycle, from raw material extraction to end-of-life disposal.

This report is carried out in collaboration with Greenly, a carbon management platform that empowers companies to simplify and automate their climate strategies. The company supports more than 3000 organizations with tools for tracking emissions, LCA modeling, supply chain analysis, and regulatory compliance, including aligning with the GHG Protocol [1], ISO 14040/44 [2, 3], and more.

The work begins by exploring the theoretical and regulatory background of LCA, followed by a deep dive into the architecture of the platform. My contributions include back-end developments to support new LCA features, such as multi-material tracking, real-time emissions simulations, and improvements in schema management and reporting standards. I worked in both the Node.js backend, using TypeORM [4] and TSOA [5], and the Python backend. In the end, I also participated in the front-end migration to React of the new platform.

Acknowledgements

First, I express my gratitude to my superior at Greenly for his guidance and support throughout this work. I am also very grateful to all the people at the company who welcomed me during my internship. Their support, patience, and willingness to share their knowledge made this experience not only productive but truly enriching. Working with such a dedicated team taught me far more than I could have imagined, and I am thankful for every lesson learned.

To my family, my parents, my brother, and my sister, thank you for being my constant source of strength. Your unwavering support of my choices and your encouragement throughout my studies have meant everything to me. Knowing that you are always there gives me the confidence to pursue my goals.

Finally, to my friends: thank you for always being available for a chat, listening and being there whenever I needed you. Your friendship has been a crucial anchor throughout this journey.

To all of you, my deepest thanks.

"You cannot get through a single day without having an impact on the world around you. What you do makes a difference and you have to decide what kind of a difference you want to make."
Jane Goodall

Table of Contents

List of Figures VII

Acronyms IX

1 Introduction 1

2 LCA background 2
- 2.1 What is an LCA . . . 2
- 2.2 LCA history . . . 2
- 2.3 ISO 14000 series . . . 4
  - 2.3.1 Life Cycle Assessment: Methodology . . . 4
- 2.4 International and European frameworks . . . 6
  - 2.4.1 GHG protocol . . . 6
  - 2.4.2 PEF and EPD . . . 6
  - 2.4.3 CFP/PCF: Product Carbon Footprint . . . 7

3 Implementation of LCA Analysis 9
- 3.1 Existing DB overview . . . 9
- 3.2 Node.js Backend . . . 12
  - 3.2.1 LCAProduct entity modifications . . . 12
  - 3.2.2 Enum modifications . . . 14
  - 3.2.3 Modifications to variable schema export, import, validate . . . 16
  - 3.2.4 Dryrun e2e for emissions simulation . . . 17
  - 3.2.5 Update by query e2e . . . 18
  - 3.2.6 Automated ISO Report Generation . . . 19
  - 3.2.7 LCAProductHasTag Implementation . . . 20
  - 3.2.8 Adding of impact title, indicator and unit to enriched LCA activity impact view . . . 22
- 3.3 Python Backend . . . 22
  - 3.3.1 Multi-material support for product purchase . . . 22
- 3.4 React Frontend . . . 26

3.4.1 LCAProduct general-info summary page 26
3.4.2 Methodology drawer component 27
3.4.3 Variables config manager 28

4 Conclusions 32

Bibliography 33

List of Figures

2.1 The life cycle approach: different phases, from [10] 3
2.2 Stages of the Life Cycle Assessment (LCA) methodology, from [6]. . 5
2.3 Visual organization the main international standards and frameworks related to Life Cycle Assessment (LCA) and environmental declarations. It highlights the relationships between general principles, product-specific rules, and regional methodologies. 8

3.1 Screenshots from an example of generated report. 21
3.2 Flow diagram of the multimaterial product creation 23
3.3 Screenshots of the form used to edit LCAProduct general infos . . . 27
3.4 Screenshots of the drawer used to visualize the emissions calculation 28
3.5 Screenshots of the variable config manager tab 29
3.6 Screenshots of the variables diagram generation 29
3.7 Screenshots of the variables importer trough OneSchema 30
3.8 Screenshots of the variables table while importing from the shared spreadsheet 31

Acronyms

LCA
Life Cycle Assessment

ISO
International Organization for Standardization

EPD
Environmental Product Declaration

PCR
Product Category Rule

LCI
Life Cycle Inventory

LCIA
Life Cycle Impact Assessment

GHG
Greenhouse Gas

WRI
World Resources Institute

WBCSD
World Business Council for Sustainable Development

PEF
Product Environmental Footprint

PEFCR

Product Environmental Footprint Category Rule

CFP

Carbon Footprint of Products

PCF

Product Carbon Footprint

AI

Artificial Intelligence

e2e

End-to-End

ORM

Object-relational mapping

API

Application Programming Interface

SWR

Stale-While-Revalidate

MUI

Material-UI

Chapter 1

Introduction

Environmental sustainability has emerged as one of the defining challenges of our era. As industries, governments and consumers increasingly recognize the imperative to reduce ecological impact, the need for rigorous, systematic methods to evaluate environmental performance has never been more critical[6]. Products do not exist in isolation: their environmental consequences extend across complex supply chains, manufacturing processes, usage patterns, and disposal pathways. Understanding these multifaceted impacts requires methodologies capable of capturing the full spectrum of a product's environmental footprint from raw material extraction to end-of-life[7][8].

This recognition has driven the evolution and standardization of environmental assessment practices over the past several decades. International organizations have developed comprehensive frameworks to guide practitioners in measuring, reporting and comparing environmental impacts with consistency and transparency[2][3]. These frameworks represent decades of scientific consensus, regulatory development, and practical refinement across diverse industrial contexts[1][9].

Yet standardized methodologies alone are insufficient. The complexity and data intensity of comprehensive environmental assessments demand sophisticated technological infrastructure. Modern organizations require digital platforms that can integrate vast databases of emission factors, process dynamic calculations across multiple impact categories, manage compliance with evolving reporting standards, and present findings in accessible, actionable formats. The challenge lies not only in understanding environmental assessment principles but in translating them into functional, scalable software systems that practitioners can effectively use. The following chapters address both dimensions of this challenge. They examine the established scientific and regulatory foundations that define how environmental impacts should be assessed, and explore the technical architecture required to implement these principles within a modern software platform serving real-world sustainability practitioners.

Chapter 2

LCA background

2.1 What is an LCA

Life cycle assessment, also called life cycle analysis, ecobalance, and cradle-to-grave analysis, is a method to evaluate the environmental effects of a product at every stage of its life, from the extraction of raw materials through manufacturing, distribution, use, and end of life, as shown in Figure 2.1. Since its inception in the 1960s, the LCA idea has become a crucial tool for comprehending and resolving the environmental effects of human activities. As society has become more aware of the need for sustainable practices and prudent resource management, it has gained importance in academics, business, and public policy. The main goal of LCA is to provide a comprehensive view of environmental impact assessment, expanding beyond a restricted focus on a single element of a product's life cycle. LCA strives to present a complete picture of how systems and goods affect the environment by taking into account variables that include energy consumption, greenhouse gas emissions, water use, resource depletion, and ecological effects.[8][7]

2.2 LCA history

Life Cycle Assessment (LCA) originated from a 1969 Coca-Cola study on bottle materials, but only gained broader application in the late 1980s. The concept was formally proposed by the Society of Environmental Toxicology and Chemistry (SETAC) in 1990 and was later integrated into the ISO 14000 Environmental Management Standards in 1993. ISO defines LCA as a systematic method for evaluating the environmental impacts of a product throughout its life cycle.

LCA has since become a key tool in environmental management and is widely used in sectors such as manufacturing, energy, and transportation. However, traditional LCA approaches have several limitations, including challenges in defining

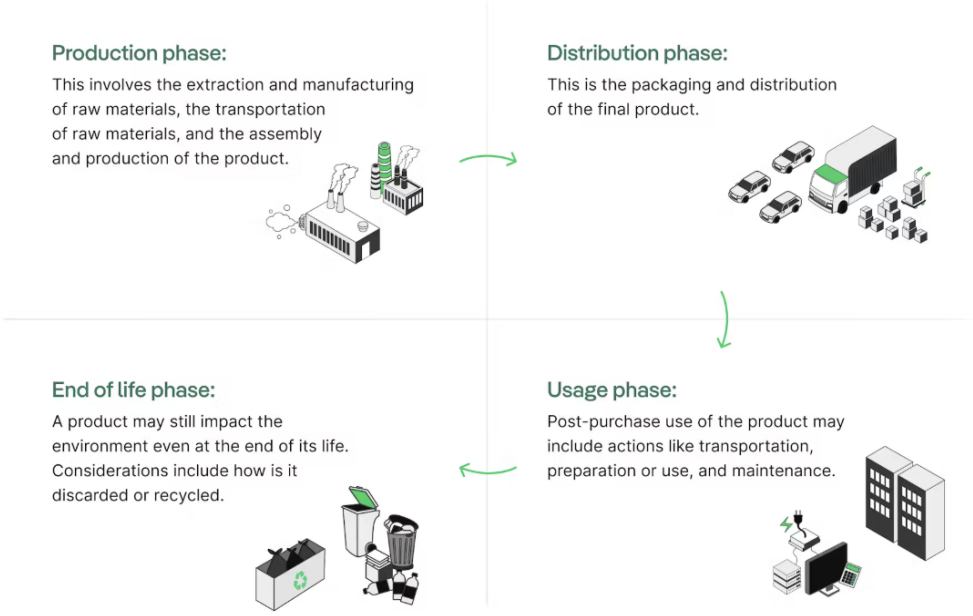


Figure 2.1: The life cycle approach: different phases, from [10]

functional units, selecting system boundaries and impact methods, ensuring data quality, and incorporating temporal, spatial, and socioeconomic factors. These assessments can also be resource intensive and less suitable for evaluating emerging technologies.

Emerging technologies like big data analytics and AI are increasingly being integrated with LCA to tackle environmental challenges. Meanwhile, life cycle sustainability assessment and social LCA are moving beyond theoretical frameworks toward practical application. In general, LCA methods have gradually evolved from traditional static evaluations to more dynamic approaches. [6].

For example, Greenly relies on several recognized life cycle assessment (LCA) databases to ensure the consistency, transparency, and scientific robustness of its evaluations. Among the most commonly used are Ecoinvent [11], one of the world's most comprehensive LCI databases covering thousands of processes across industries and geographies, Agribalyse [12], which provides detailed agricultural and food sector inventories developed by ADEME and its partners, and the broader ADEME datasets [13], which support environmental reporting and footprint calculations across multiple domains. In addition, the company leverages corporate big data and AI models to improve the accuracy of emission calculations and streamline the overall reporting and assessment process, facilitating more efficient and precise life-cycle analyzes.

2.3 ISO 14000 series

The backbone of LCA is established by the ISO 14000 series, published by the International Organization for Standardization (ISO), the relation :

- **ISO 14025:2006** ("Environmental labels and declarations - Type III environmental declarations — Principles and procedures") defines the principles and procedures for developing Type III environmental declarations, also known as Environmental Product Declarations (EPDs)[9]. These declarations are based on Life Cycle Assessment (LCA) data and provide quantified environmental information for products using predetermined parameters. The standard emphasizes third-party verification, comparability between products within the same category, and the use of Product Category Rules (PCR) to ensure consistency and transparency [14].
- **ISO 14040:2006** ("Environmental management—Life cycle assessment— Principles and framework") describes the general principles and framework for conducting LCA, including the four main phases, reporting, critical review and the relationship between the LCA phases[2][8].
- **ISO 14044:2006** ("Environmental management—Life cycle assessment— Requirements and guidelines") specifies technical requirements and provides detailed guidance for performing an LCA, including data quality, allocation procedures, impact assessment and reporting[3][8].
- **ISO 14067:2018** ("Greenhouse gases - Carbon footprint of products — Requirements and guidelines for quantification") sets out principles, requirements and procedures for quantifying and reporting the carbon footprint of products. It is consistent with ISO 14040 and ISO 14044 and emphasizes transparency, precision, and comparability in carbon footprint studies [15].

These standards are internationally recognized and serve as the foundation for most LCA applications, ensuring consistency, transparency, and credibility in environmental assessments.

2.3.1 Life Cycle Assessment: Methodology

As we said, Life Cycle Assessment (LCA) is a systematic technique used to evaluate the environmental aspects and potential impacts associated with a product throughout its entire life cycle. The most important applications of LCA include:

- Analysis of the contribution of different stages of the life cycle to the overall environmental impact, typically aimed at identifying opportunities for improvement in products or processes.

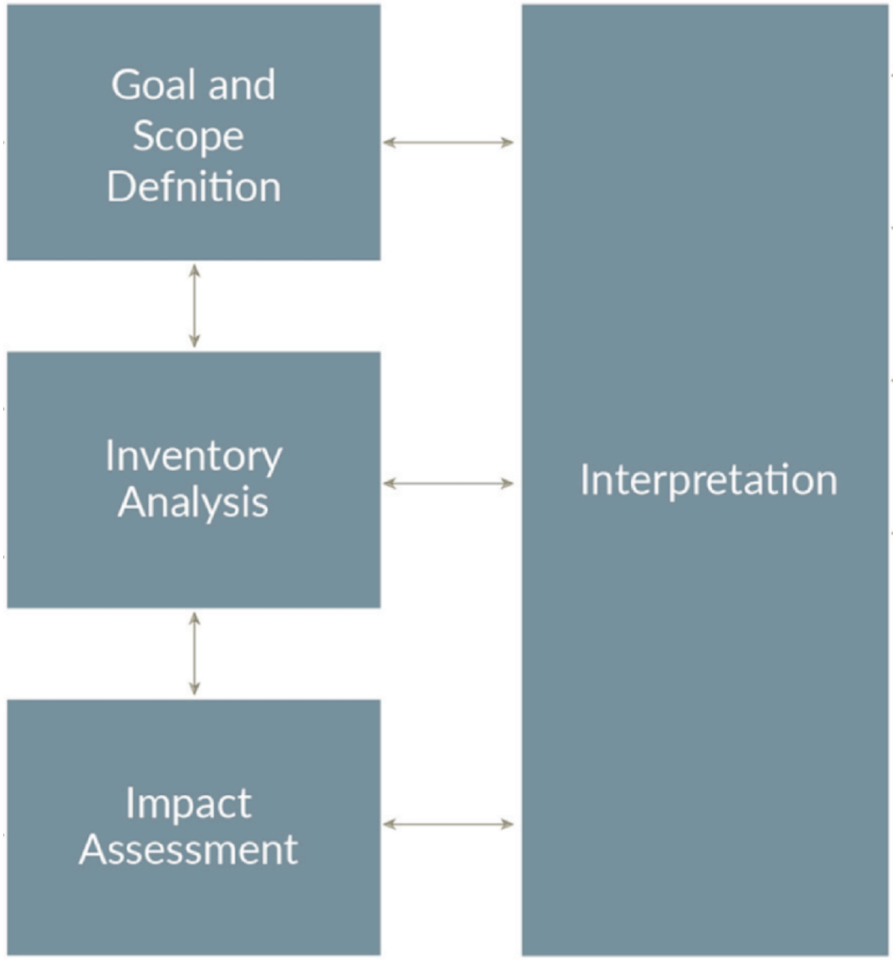


Figure 2.2: Stages of the Life Cycle Assessment (LCA) methodology, from [6].

- Internal comparison of products to support decision making within an organization.

There are four main stages involved in conducting an LCA, as illustrated in Figure 2.2:

1. **Goal and Scope Definition**: This phase defines the purpose of the study, the functional unit, system boundaries, and the criteria for product or system comparison. It determines the scale and scope of the assessment and specifies the temporal and spatial dimensions of the analysis.

2. **Inventory Analysis (LCI)**: This stage quantifies the input and output flows of materials and energy within the product system. It includes:

- *Inputs*: Raw materials, energy, water, and other resources consumed during each life cycle stage.
- *Outputs*: Emissions to air, water, and soil, along with the generation of waste throughout the product's life cycle.

3. **Impact Assessment (LCIA)**: Using the results of the inventory analysis, this phase assesses the significance of possible environmental impacts. The results are typically grouped by impact categories (e.g., climate change, acidification) and analyzed using normalization and weighting techniques to determine their relative importance.

4. **Interpretation**: This final phase involves the critical review and interpretation of the LCA results. It includes uncertainty and sensitivity analyses and aims to present clear conclusions and recommendations for decision-making. [8] [7]

2.4 International and European frameworks

2.4.1 GHG protocol

The Greenhouse Gas (GHG) Protocol is a globally recognized framework developed by the World Resources Institute (WRI) and the World Business Council for Sustainable Development (WBCSD) in the late 1990s. It provides standardized methodologies for measuring and managing greenhouse gas emissions across businesses and value chains.

A key component of the GHG Protocol is the *Product Life Cycle Accounting and Reporting Standard*, which aligns with ISO 14040 and integrates the life cycle assessment principles (LCA). This standard enables companies to evaluate the total emissions of a product from cradle to grave, covering all stages from raw material extraction to production, use, and disposal.

Unlike traditional GHG reporting, which often focuses on direct emissions (Scopes 1 and 2), life-cycle-based accounting under the Product Standard extends to indirect emissions (Scope 3), allowing for a more holistic environmental impact analysis. [1][16]

2.4.2 PEF and EPD

Environmental Product Declarations (EPDs) and the Product Environmental Footprint (PEF) represent two interrelated but distinct tools within life cycle-based environmental assessment. An EPD is a Type III environmental declaration, standardized under ISO 14025[14] and based on a verified life cycle assessment (LCA), which enables manufacturers to transparently disclose environmental performance

data. EPDs are widely used in public procurement, green building certification schemes, internal performance monitoring, benchmarking, and marketing [9].

In contrast, the PEF is a methodological framework developed by the European Commission to provide a harmonized approach based on LCA to assess the environmental impacts of products and services. It builds upon ISO 14040 and ISO 14044[2][3] principles but introduces more prescriptive modeling requirements, especially through the use of Product Environmental Footprint Category Rules (PEFCRs), to ensure comparability between products [17]. PEF studies follow a consistent sequence: definition of the objective and scope, life cycle inventory analysis, impact assessment, and interpretation, all subject to verification and strict methodological guidance.

Together, EPDs and PEF serve complementary roles: EPDs offer a standardized format for environmental communication based on LCA, while the PEF provides a unified and policy-aligned methodology for generating robust and comparable environmental performance data across European markets.

2.4.3 CFP/PCF: Product Carbon Footprint

The *Product Carbon Footprint (PCF)* measures the total greenhouse gas (GHG) emissions associated with a product throughout its life cycle. This includes emissions from raw material extraction, manufacturing, distribution, usage, and end-of-life treatment such as disposal or recycling. Unlike a full Life Cycle Assessment (LCA), which considers multiple environmental impacts, the PCF focuses exclusively on climate impact, expressed in CO_2 equivalent (CO_2e). By quantifying emissions at each stage, PCF assessments allow companies to identify carbon-intensive "hotspots," design effective reduction strategies, and improve transparency with stakeholders.[18] To ensure methodological consistency and comparability across industries, PCF calculations are guided by international standards such as ISO 14067:2018, which defines principles, requirements, and guidelines for quantifying and reporting product carbon footprints [15].

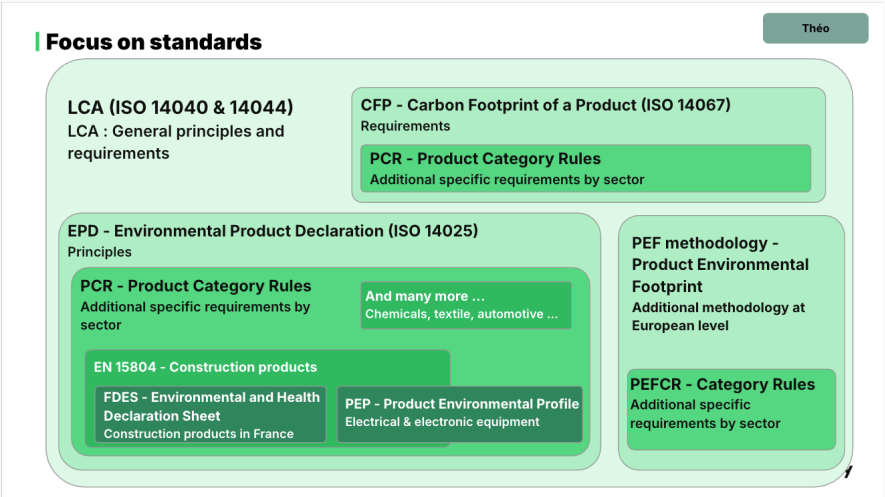


Figure 2.3: Visual organization the main international standards and frameworks related to Life Cycle Assessment (LCA) and environmental declarations. It highlights the relationships between general principles, product-specific rules, and regional methodologies.

Chapter 3

Implementation of LCA Analysis

3.1 Existing DB overview

The existing database includes a lot of tables, of which only a subset I dealt with, and for those one here is an explanation of the reasons and the main parameters:

- **company**
  Parameters:
  - `id`, `createdAt`, `lastUpdatedAt`, `companyName`
  - `industry`, `parentIndustry`, `type`, `companyLanguage`
  - `countryId`, `companyLogoUrl`

  This entity represents organizations being assessed. It stores identity, classification, and operational settings needed for reporting and linking to users, profiles, and activities.

- **user**
  Parameters:
  - `id`, `email`, `firstName`, `lastName`
  - `phoneNumber`, `jobTitle`, `role`, `password`
  - `createdAt`, `lastUpdatedAt`, `lastConnectedAt`

  Users interact with the platform, manage company data, and are associated with roles and permissions.

- **lca_activity**
  Parameters:
  - `id`, `lcaProductId`, `companyId`, `profileId`, `type`
  - `quantity`, `unit`, `emissionFactorId`, `emissionsInKg`, `quantityVariableId`

  Records specific activities of an lcaProduct contributing to emissions. It is the core input for carbon footprint calculations.

- **enriched_lca_activity_impact**
  Parameters:
  - `lcaProductId`, `companyName`, `activityId`, `emissionFactorId`
  - `carbonFootprintEmissions`, `productCountry`, `productUnitaryWeight`
  - `impact`,

  Focuses on the impact dimension, storing aggregated impact values (e.g., footprint results) linked to products and activities.

- **versioned_emission_factor**
  Parameters:
  - `id`, `name`, `UNRegion`, `countryId`
  - `methodology`, `ratioEmissionsKgPerUnit`, `unit`
  - `functionalUnit`, `year`, `confidenceScore`, `source`

  Stores standardized emission factors used to convert activities into emissions, with versioning for auditability.

- **emission_factor_has_tag**
  Parameters:
  - `emissionFactorId`, `tagId`, `value`, `type`

  Enables flexible classification of emission factors with tags.

- **conversion_factor_has_tag**
  Parameters:
  - `conversionFactorId`, `tagId`, `value`, `type`

  Similar to emission factor tags, for conversion factors.

- **versioned_conversion_factor**
  Parameters:

  - `id`, `name`, `value`, `unitInput`, `unitOutput`
  - `UNRegion`, `countryId`, `methodology`

  Defines unit conversion rules (e.g., recycled product) with version control.

- **versioned_entity_variable**
  Parameters:
  - `id`, `entityId`, `schemaKey`, `type`, `title`
  - `description`, `unit`, `boundsMin`, `boundsMax`, `options`

  Provides extensible, schema-based variables for different entities.

- **versioned_entity_variable_value**
  Parameters:
  - `id`, `entityId`, `entityVariableId`, `valueType`
  - `numericValue`, `stringValue`, `unit`, `source`

  Stores actual values for defined entity variables, supporting customization and assumptions.

- **versioned_lca_product**
  Parameters:
  - `id`, `name`, `stages`, `tags`, `impactAssessmentMethod`
  - `functionalUnit`, `systemBoundaries`, `productType`
  - `unitaryWeight`, `unitaryQuantity`, `productDescription`

  Represents products subject to LCA, describing functional units, system boundaries, and attributes.

- **tag**
  Parameters:
  - `id`, `name`, `companyId`,

  Base entity for tagging; enables flexible metadata classification across entities.

- **entity_has_tag**
  Parameters:
  - `entityId`, `tagId`, `value`, `type`

  Many-to-many link enabling tags to be applied to entities like activities, companies, or transactions.

- **external_app_data**
  Parameters:
  - `id`, `applicationId`, `companyId`
  - `entityId`, `entityType`, `appData`

  Allows integration with external applications, storing additional app-specific data linked to entities.

3.2 Node.js Backend

This backend code is implemented in TypeScript, using TypeORM[4] as the ORM layer and TSOA to generate end-to-end APIs from the controllers. TSOA is a TypeScript framework that facilitates the creation of RESTful APIs using TypeScript decorators. It enables automatic generation of OpenAPI (formerly Swagger) documentation and routing, while also supporting runtime validation of request inputs based on defined interfaces. TSOA promotes a separation of concerns by allowing developers to define the API contracts through annotations in controller files, which are then used to generate routes and validation logic automatically. This improves maintainability, type safety, and reduces boilerplate code in the backend[5]. The existing codebase includes a wide range of endpoints and predefined entities. The Node.js backend provides full CRUD (Create, Read, Update, Delete) functionality. In addition to entity management, it supports various auxiliary features and smaller functionalities required by the system. Furthermore, the architecture includes components known as UseCases, which encapsulate specific business logic used by the APIs. Each entity is accompanied by unit tests, as are its corresponding UseCases and controllers, ensuring a high level of code reliability and maintainability.

3.2.1 LCAProduct entity modifications

Added ReportingStandards column

As part of the task assigned to me, I was required to extend the LCA product model by adding a new column that captures reporting standards information. The structure of this column is as follows in the Listing3.1:

```
1    {
2        "iso14044":{
3            "isStandardRequestedByClient": true,
4            "isProductConformToStandard": false,
5            "deadline": "2024-03-02"
```

```
        },
        "EPD":{
            "isStandardRequestedByClient": false,
            "isProductConformToStandard": false,
            "deadline": null
        },
        ...
    }
```

Listing 3.1: example of ReportingStandards content

This field enables the system to record whether a given product is subject to any reporting standards, along with the relevant compliance data. To implement this functionality, I defined the following TypeScript type in the Listing3.2:

```
export type ReportingStandardsRecord =
    Partial<Record<ReportingStandardsEnum,
ReportingStandardsType>>
```

Listing 3.2: type of ReportingStandardsRecord

Here, ReportingStandardsEnum represents the list of supported reporting standards, allowing the system to maintain flexibility in handling multiple standards. Each entry in the record corresponds to a ReportingStandardsType, which encapsulates the necessary information as showed in Listing3.3:

```
export type ReportingStandardsType = {
    isStandardRequestedByClient: boolean;
    isProductConformToStandard: boolean;
    deadline: Date | null;
}
```

Listing 3.3: type of ReportingStandardsType

This data model ensures that the reporting requirements and compliance status of each product are captured in a structured and extensible manner.

Added isProductFrozen and validationStatus columns

In the course of carrying out my task, I had to introduce two new columns to the `LCAProduct` entity: `validationStatus` and `isProductFrozen`. These columns are defined as shown in Listing 3.4:

```
@Column({type: 'text', nullable: false, default: 'to_do'})
validationStatus: LCAValidationStatus;

@Column({type: 'boolean', nullable: false, default: false})
isProductFrozen: boolean;
```

Listing 3.4: Definition of `validationStatus` and `isProductFrozen`

The `LCAValidationStatus` type is an enumerated list containing the values `'in_review'`, `'in_progress'`, `'done'`, and `'to_do'` and `'error'` representing the different stages of a product's reporting validation process. The `isProductFrozen` boolean flag is used to prevent any further modifications to the product while someone is modifying it, thereby ensuring data integrity.

Unique constrain on name and externalId by company

As part of my recent work, I was tasked with enforcing two unique indexes on the `products` collection: one for the combination of `name` and `companyId`, and another for `externalId` and `companyId`, specifically for products where `isLastVersion = true` and `isDeleted = false`.

To ensure a smooth deployment without disrupting the existing database, the implementation was divided into three key steps:

1. **Update Application Logic**: I modified the `LCAProduct` creation, duplication, and import logic to validate the new uniqueness constraints at the application level. If a conflict is detected, the system now returns a clear error message to prevent invalid data from being stored.

2. **Clean Up Existing Duplicates**: I developed and executed two scripts to resolve existing duplicates in the database. These scripts append a numeric suffix (e.g., "(1)", "(2)") to duplicated `name` or `externalId` fields for products within the same company.

3. **Apply Database Constraints**: After confirming that all duplicates were resolved and the application logic enforced the new rules, I applied the unique indexes directly to the database to guarantee long-term data integrity.

However, after deploying these changes, it was discovered that certain automated processes and legacy data flows were needing the not unique columns. These conflicts caused blocking errors in production, which led to the decision to temporarily revert both the application-level validations and the database-level constraints.

3.2.2 Enum modifications

ImpactAssessmentMethod enum

As a minor yet necessary modification, I was required to extend the `ImpactAssessmentMethod` enumeration to include a new key representing the

method used for conducting the Life Cycle Impact Assessment (LCIA). LCIA "is aimed at evaluating the potential environmental and human health impacts resulting from the elementary flows determined in the LCI(Life Cycle Inventory)"[19]. This method can vary depending on the specific standard or regulatory framework being followed.

The updated enumeration now includes entries such as in Listing3.5:

```
export enum ImpactAssessmentMethod {
    ...
    efMethod = 'Ef method',
    efMethodV3_1 = 'Ef v3.1',
    epd2018 = 'EPD (2018)',
    ghgProtocol = 'Greenhouse gas protocol',
    ...
}
```

Listing 3.5: example of ImpactAssessmentMethod enum

LCAImpact enum

One of the tasks I undertook involved extending the backend system by incorporating additional environmental impact categories into the `LCAImpact` enumeration. Examples of the newly added impact categories include:

- `ClimateChangeBiogenicExclCO2KgCO2Eq`: Represents greenhouse gas emissions from biogenic sources (e.g., methane or nitrous oxide from biomass) but excludes biogenic carbon dioxide (CO_2), which is often treated as climate-neutral in short-cycle biomass systems.

- `ClimateChangeDirectLandUseChangeKgCO2Eq`: Accounts for greenhouse gas emissions resulting from direct land use change (dLUC), such as deforestation, afforestation, or conversion of natural ecosystems into agricultural or industrial land, which significantly impacts carbon stock and soil emissions.

- `ClimateChangeAircraftEmissionsKgCO2Eq`: Captures emissions specifically associated with aircraft operations, considering both direct CO_2 emissions and climate-forcing effects unique to aviation, such as high-altitude emissions of NO_x and contrail formation.

To support these additions, corresponding fields were introduced into the `BaseEmissionFactorRequestBody` input type. This ensures that each relevant impact category can be assigned a value, allowing the system to capture and process the associated life cycle assessment (LCA) data. The structure now includes entries such as in Listing3.6:

```
export type BaseEmissionFactorRequestBody = {
    ...
    lcaImpactsPerUnit?: {
        ...
        ClimateChangeBiogenicExclCO2KgCO2Eq: number | null;
        ClimateChangeDirectLandUseChangeKgCO2Eq: number | null;
        ClimateChangeAircraftEmissionsKgCO2Eq: number | null;
        ...
    }
}
```

Listing 3.6: example of BaseEmissionFactorRequestBody

3.2.3 Modifications to variable schema export, import, validate

In the course of carrying out my task, I worked with three API endpoints: `exportLCAProductsParametersAsJSONSchema`, `importLCAProductsUsingSchema`, and `validateLCAProductImportData`.

The first endpoint, `exportLCAProductsParametersAsJSONSchema`, is responsible for exporting all the parameter variables defined for a specific product as a JSON Schema. An example of such a schema is shown below in Listing3.7:

```
{
    "type": "object",
    "properties": {
        "name": {
            "type": "string",
            "title": "Product Name",
            "description": "The name of the product"
        },
        "externalId": {
            "type": "string",
            "title": "Internal Identifier",
            "description": "The identifier of the
   product"
        },
        ...
        "width": {
            "type": "number",
            "title": "width (cm)",
            "description": "desc"
        },
```

```
        "depth": {
            "type": "number",
            "title": "depth (cm)",
            "description": "desc"
        }
    }
}
```

Listing 3.7: returned schema of exportLCAProductsParametersAsJSONSchema

My contribution involved two main improvements. First, I implemented a sorting mechanism for these variables based on an internal `order` attribute defined in each variable's schema. Second, I enhanced the readability of the schema for users by appending the unit of measurement (e.g., `(cm)`) to the parameter's title and by introducing a new field indicating the functional unit. This makes the data input process more intuitive and standardized.

Since this schema is used as a template to import new products with a consistent structure, these modifications also affected the behavior of the `validateLCAProductImportData` and `importLCAProductsUsingSchema` endpoints. Specifically, it was necessary to strip the appended unit (in parentheses) during validation and import, and to ensure that the newly added functional unit field was properly handled in both APIs.

3.2.4 Dryrun e2e for emissions simulation

Another task I was responsible for involved the creation of an E2E feature that, given as input a list of EntityVariableValue objects—each representing a variable and its corresponding value—first retrieves the product associated with the relevant variable schema. It then identifies the activities linked to that product.

On these activities, the system performs a dry-run update, meaning it estimates the outcome of changes (specifically emissions) based on the input values without persisting any modifications to the database. This mechanism leverages the quantityVariableId to simulate calculations and estimate emissions, allowing for the evaluation of environmental impacts in a non-destructive way.

The endpoint accepts input in the following format:

```
[
    {
        "unit": "cm",
        "value": 96,
        "key": "width",
        "variableId": "a30ad...nds92nd",
    },
```

```
    {
        "unit": "cm",
        "value": 10,
        "key": "depth",
        "variableId": "c27ad...dsd87lf",
    },
]
```

Listing 3.8: body of the dryRun endpoint

And it returns a list of activities with computed emissions:

```
[
    {
        "id": activity.id,
        "quantity": activity.quantity,
        "emissionInKg": activity.emissionInKg,
        "otherImpacts": activity.otherImpacts,
        ...
         "quantityVariableId": activity.
   quantityVariableId,
    }
]
```

Listing 3.9: returned body of the dryRun endpoint

This functionality enables dynamic impact assessment based on variable input, facilitating design exploration and decision-making without altering the underlying data.

3.2.5 Update by query e2e

As part of the work I was required to complete, I developed a new API endpoint designed to update multiple entries in the `LCA Activity` entity based on a search query. This endpoint allows the frontend to efficiently modify a set of activity records by providing two key components: a search query (using the pre-existing `EntitySearchBody` structure) and a set of attributes to update.

The API accepts a JSON request body structured as follows:

```
{
    "query": {
        "companyIds": [...],
        "conditions": {
            "combinator": "and",
```

```
            "rules": {
                "field": "id",
                "operator": "=",
                "value": id
            }
        }
    },
    "changes": {
        "overridenForVisualizationPurchaseCategoryId":
   ...,
        "name": "string",
        "shouldBeAccounted": true,
        "isVerified": false,
        ...
    },
    "companyId": ...,
    "maxUpdates": 1000
}
```

Listing 3.10: body of the updateByQuery endpoint

The `query` field specifies which activities should be selected based on filtering criteria, such as matching company IDs and custom rules. The `changes` object defines the new values to assign to the selected entries, including optional fields like `name`, `shouldBeAccounted`, `isVerified`, and `overridenForVisualizationPurchaseCategoryId`. The `maxUpdates` parameter sets an upper limit on the number of records that can be modified in a single operation, ensuring scalability and safety.

This implementation streamlines the process of bulk-editing activity data and significantly reduces the complexity of managing updates from the frontend.

3.2.6 Automated ISO Report Generation

As part of my assigned responsibilities, I implemented an automated report generation system to comply with ISO 14040 [8]. This involved creating an end-to-end (e2e) API endpoint that, upon receiving a payload containing relevant company and product data, generates a comprehensive DOCX report. The report is populated with all the necessary sections mandated by the standard, as outlined in paragraph 2.3.1: namely Goal and Scope Definition, Inventory Analysis (LCI), Impact Assessment (LCIA), and Interpretation. For this task, I utilized the `docx` library for JavaScript [20] to programmatically build the Word document. Additionally, I integrated the Luzmo APIs [21] to embed dynamic analytics dashboards within the report, enhancing its interactivity and usefulness.

Once generated, the report is saved to Google Drive through an already implemented client service. Following a successful upload, the system stores the generated file ID to create the URL, in the `external_app_data` table, enabling clients to retrieve and download the report seamlessly.

From an architectural point of view, the report generation process begins in the `LCAReportController`. The `generateReport` endpoint is exposed via a POST request to `/lca-report/generate`, secured with authentication and company-level authorization middleware. When invoked, this endpoint extracts the `companyId`, the `authorId` (derived either from the authenticated session or a request header), and the `LCAReportDataBody` containing the report content. It then triggers an asynchronous job via the `GreenlyQueueManager`, specifically enqueuing the task on the `ISOLCAReportGeneration` queue. The job payload includes all the data needed to build the report.

The actual generation logic resides in the use case layer, under the `useCases/reportGeneration` directory. This layer processes the input data and delegates the document construction to a dedicated function such as `createDocument`. This function assembles the report dynamically by combining prebuilt, modular sections, stored in a `sections/` directory, covering various parts of the LCA process. The visual styling and formatting of the document, such as fonts, tables, headers, and pagination, are handled by utility modules like `styles.ts` and `utils.ts`, ensuring consistency and visual coherence across all generated documents. This structured flow enables scalable and maintainable report generation aligned with ISO standards.

3.2.7 LCAProductHasTag Implementation

This update introduces a new entity, `LCAProductHasTag`, to extend the system's tagging capabilities specifically for LCA products. The main changes implemented are summarized below.

Entity Model and Repository

- Introduced a new model representing the `LCAProductHasTag` entity, which defines the many-to-many relationship between LCA products and tags.

- Created a dedicated repository to manage persistence and query operations for this new entity, ensuring efficient and consistent tag handling for LCA products.

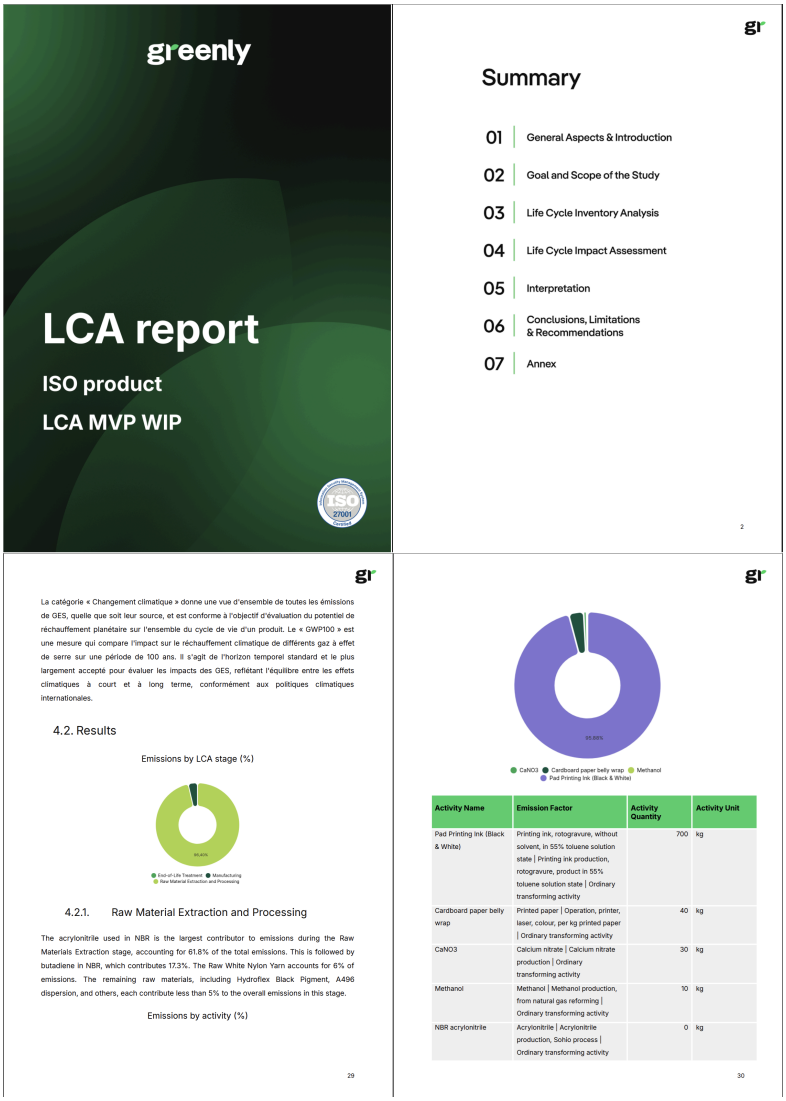


Figure 3.1: Screenshots from an example of generated report.

Business Logic Modifications

- Refactored the functions `createEntityHasTag` and `deleteEntityHasTag` to support the `LCAProductHasTag` entity, enabling tag assignment and removal using the existing generic tagging interface.
- Ensured logical consistency in tag processing across all supported entity types, now including LCA products.

In summary, this development extends the tagging system to support LCA

products, introduces the necessary backend structures and logic, and ensures correctness through comprehensive testing.

3.2.8 Adding of impact title, indicator and unit to enriched LCA activity impact view

To add a title, indicator and unit to the impacts used to create the enriched LCA activity impact view 2 pull request have been necessaries. The first one introduces a foundational overhaul of the units system by migrating the database schema to accommodate a significantly expanded set of `UnitOfMeasure`s across multiple PostgreSQL enumerations. This migration, executed via a series of `ALTER TYPE ... ADD VALUE` statements, added new codes such as `CTUE`, `KGPE`, `CTUH`, `KBQU235E`, `KGCDEFICIT`, and `KGNE`. These additions extend unit coverage across domains including activity, computed weight, volume, and emission factors, ensuring broader compatibility and more precise representation of scientific and industrial measures within both LCA and building domain models.

The second pull request builds on this foundation by enhancing both the database and application layers to improve how these units are presented and used. Specifically, it adds descriptive metadata fields (`title`, `indicator`, and `unit`) to the `enriched_lca_activity_impact_view`.

Together, these two pull requests establish a robust and extensible infrastructure for unit handling that has been leveraged to implement a multi-criteria impact analysis within the LCA process, enabling the consistent comparison of multiple environmental impact categories (e.g., climate change, land use, resource use) and supporting more holistic and transparent decision-making.

3.3 Python Backend

The backend of the Greenly codebase also incorporates Python modules that handle AI-driven categorization, process data extracted from customer-submitted templates, and support various additional functionalities.

3.3.1 Multi-material support for product purchase

As part of the task I was assigned, I modified the existing implementation related to the creation of `LCAProducts` and `LCAActivities` that occurs when purchased products are processed. In the previous version of the system, only the main material of a product was considered during this creation process, resulting in a significant approximation of the emissions.

The updated flow, illustrated in Figure 3.2, introduces a more accurate approach by taking into account at most 5 constituent materials of a product together with the recycled parts, not just the primary one:

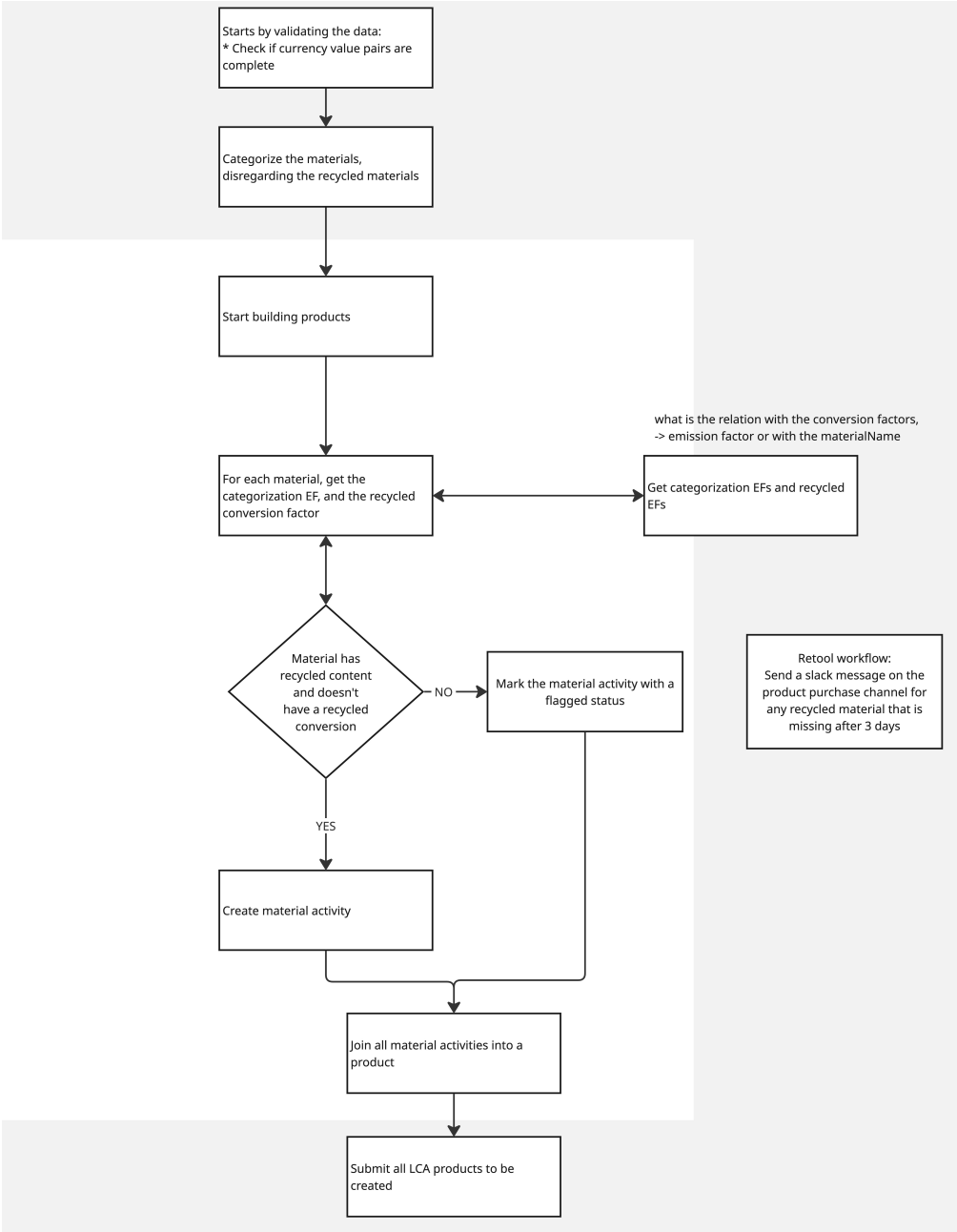


Figure 3.2: Flow diagram of the multimaterial product creation

In this revised process, all non-recycled materials are first categorized using

a pre-existing AI-based classification module. Based on the classification result, an appropriate emission factor is applied to the corresponding standard part or activity. For recycled materials, the process attempts to retrieve the corresponding conversion factor to be used together with the emission factor.

If no suitable conversion factor is available for a given recycled material, a warning tag is automatically assigned. This tag indicates that intervention is required by the Research & Methodology team, who are responsible for defining or updating the missing conversion factor for the specific material.

To determine whether a conversion factor is associated with a recycled material, it was necessary to modify the existing query that retrieves emission factors. A new field, `has_recycled_factor`, was added to flag this association. The value of this field is determined using the following query:

```
1   ef.id IN (
2         SELECT cv_has_tag.value
3         FROM conversion_factor_has_tag AS cv_has_tag
4         JOIN tag ON tag.id = cv_has_tag.tagId
5         WHERE tag.isLastVersion = TRUE
6           AND tag.isDeleted = FALSE
7           AND tag.companyId IS NULL
8           AND cv_has_tag.conversionFactorId IN (
9               SELECT inner_cv_has_tag.conversionFactorId
10              FROM conversion_factor_has_tag AS inner_cv_has_tag
11              JOIN tag AS inner_tag ON inner_tag.id =
    inner_cv_has_tag.tagId
12              WHERE inner_tag.name = 'FactorIdentifier'
13                AND inner_cv_has_tag.value = '
    ProductPurchaseRecycledFactor'
14          )
15          AND tag.name = 'ProductPurchaseEmissionFactorId'
16    ) AS has_recycled_factor
```

Listing 3.11: SQL query to identify emission factors with recycled material conversion

The input type, received by reading an Excel file, is structured as follows:

```
1  class ProductPurchaseE2EInput(CommonModuleInputs):
2      id: Annotated[UUID4, StrSerializer] | None = None
3      activityId1: Annotated[UUID4, StrSerializer] | None = None
4      ...
5      activityId5: Annotated[UUID4, StrSerializer] | None = None
6      activityId1Recycled: Annotated[UUID4, StrSerializer] | None =
     None
7      ...
8      activityId5Recycled: Annotated[UUID4, StrSerializer] | None =
     None
9      materialName1: str
```

```
    materialMassRatio1: Annotated[float, Field(ge=0, le=100)]
    materialRecycledContent1: Annotated[float, Field(ge=0, le=100)] | None = None
    ...
    materialName5: str | None = None
    materialMassRatio5: Annotated[float, Field(ge=0, le=100)] | None = None
    materialRecycledContent5: Annotated[float, Field(ge=0, le=100)] | None = None
    quantity: Annotated[float, Field(ge=0)] | None = None
    weight: Annotated[float, Field(ge=0)]
    weightUnit: Annotated[MassUnitOfMeasure, EnumSerializer]
    tab: Annotated[ProductPurchaseE2EInputTab, EnumSerializer]
    internalId: str
    supplier: str
    ...
```

Listing 3.12: Python object received as input

Given the total quantity of the product, the mass fractions of each constituent material, and their corresponding recycled content, it is necessary to calculate the specific quantity allocated to each activity. Each material is mapped to a distinct activity, and if recycled content is present, a separate activity is also defined for the recycled portion.

The products output objects are structured as follows:

```
class ProductPurchaseE2EProduct(BaseModel):
    id: Annotated[UUID4, StrSerializer] | None = None
    stages: list[ProductPurchaseE2EProductStage]
    companyId: Annotated[UUID4, StrSerializer]
    specificModuleId: Annotated[UUID4, StrSerializer]
    fileId: Annotated[UUID4, StrSerializer] | None = None
    productType: Literal["MANUFACTURED_PRODUCT", "RAW_MATERIAL"] | None = None
    externalId: str | None = None
    name: str
    unitaryQuantity: Annotated[float, Field(ge=0)] | None = None
    unitaryWeight: Annotated[float, Field(ge=0)]
    unitaryWeightUnit: Annotated[MassUnitOfMeasure, EnumSerializer]
    ...
    activities: list[ProductPurchaseE2EActivity] | None = None
```

Listing 3.13: Python object sent as output

The activities output objects are as follows:

```
class ProductPurchaseE2EActivity(BaseModel):
    activityId: Annotated[UUID4, StrSerializer] | None = None
    stage: str
```

```
    name: Translation
    subType: str
    section: Translation | None = None
    quantity: Annotated[float, Field(ge=0)] | None = None
    unit: Annotated[MassUnitOfMeasure, EnumSerializer]
    ...
    businessUnit: str | None = None
    additionalInformation: dict
    inputSupplierLabel: str | None = None
    customDimension1: str | None = None
    ...
    customDimension10: str | None = None
    targetEntityId: str | None = None
```

Listing 3.14: Python object sent as output

These are transmitted to the Node.js backend, which is responsible for creating the `LCAProduct` entity first, followed by the creation of the associated `LCAActivities`.

This improvement greatly enhances the accuracy and completeness of the life cycle assessment by accounting for the full material composition of each product, particularly in cases where recycled content significantly affects environmental performance.

3.4 React Frontend

During my internship, the company decided to migrate the frontend to a modern stack based on React [22] and Next.js [23]. This migration improves performance, modularity, and maintainability by introducing reusable components, type-safe validation, and better integration with back-end services.

3.4.1 LCAProduct general-info summary page

The modifications include the introduction of a modular, multi-step form for entering product information, leveraging React hooks and context for state management. Components such as `LCAInfoSection` and `QuestionField` were created to provide reusable UI blocks for displaying sectioned information and dynamic form fields (text, select, chip select, number), in order to provide dynamic questions depending on the type of LCA.

The setup and handling of LCA product parameters is simplified using utility functions such as `setupUpdateBodyParams`, which maps and prepares product data for updates based on user input. The flow now covers product overview, internal tracking, LCA configuration, hypotheses and boundaries, with each step isolated in

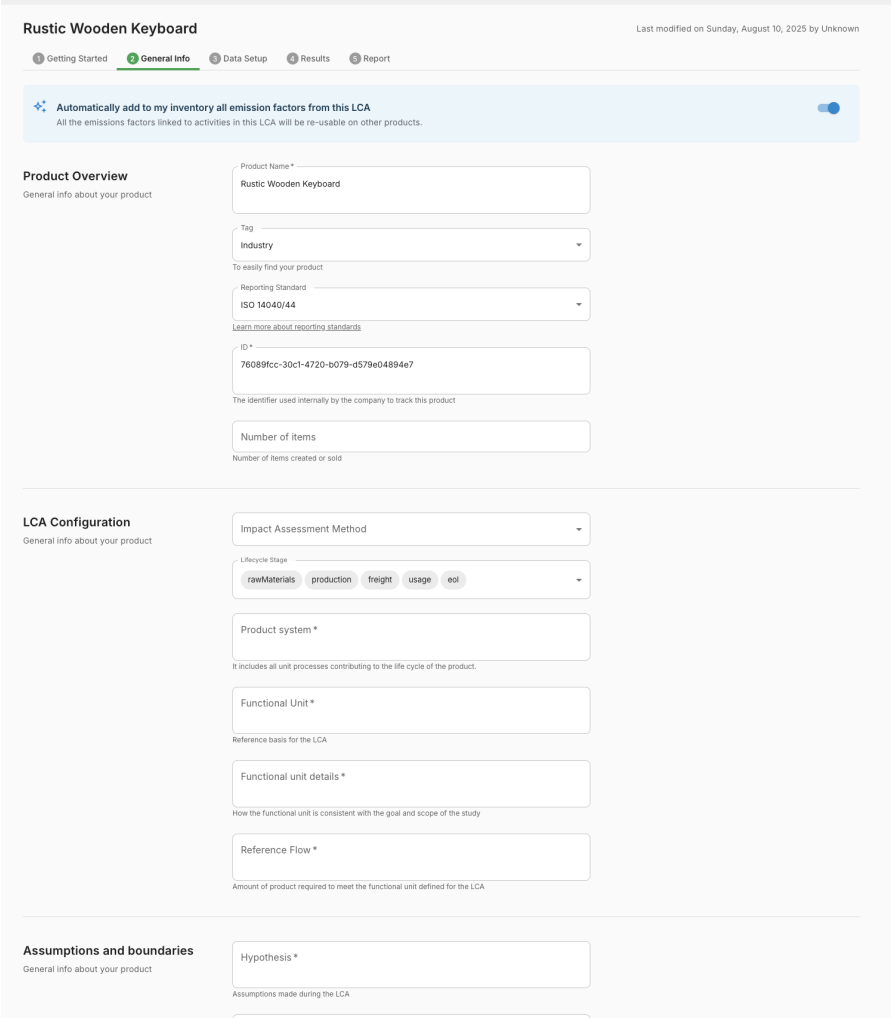


Figure 3.3: Screenshots of the form used to edit LCAProduct general infos

its own component for maintainability and clarity. Enhanced validation is provided using `zod`[24] schemas, and internationalization is supported via `next-intl`[25].

The screenshot in 3.3 shows the form used to edit the general info of the LCAProduct and save them.

3.4.2 Methodology drawer component

The main goal of this update was to migrate and enhance the methodology drawer by fetching variable schemas dynamically from the back-end to then show them.

In addition, the drawer provides detailed explanations for quantity calculations, including contextual descriptions and breakdowns of how each quantity is derived.

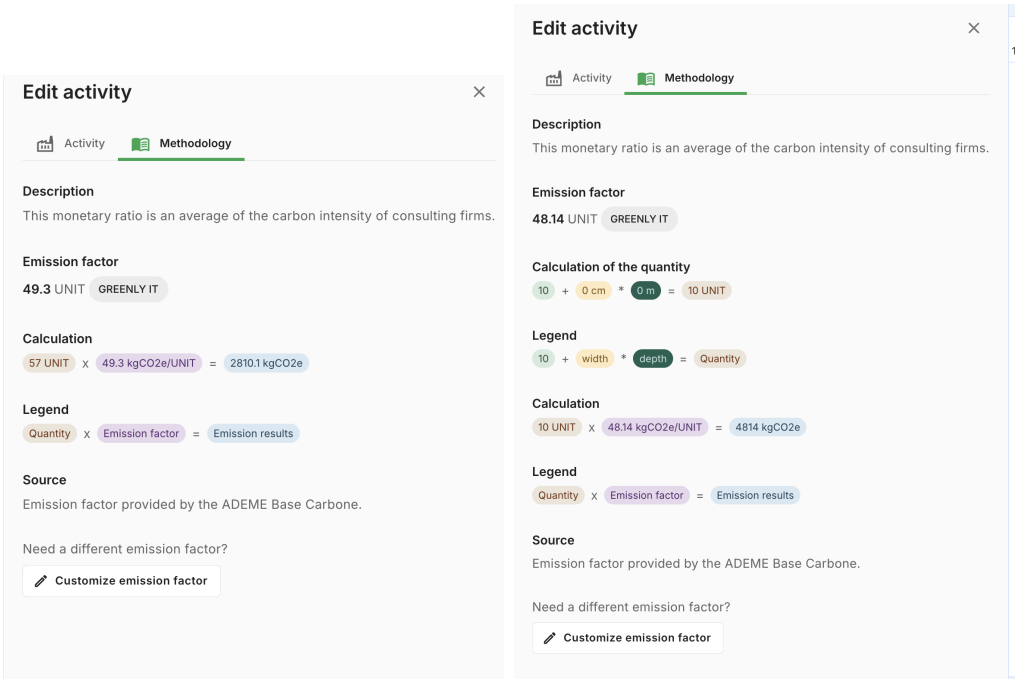


Figure 3.4: Screenshots of the drawer used to visualize the emissions calculation

Improvements also include clearer representations of how variable formulas are computed, ensuring that users can trace the relationship between input data, formulas, and resulting values.

With these enhancements, the methodology panel becomes a more transparent and informative resource, allowing users to better understand the logic and calculations behind the results of the Lifecycle Assessment (LCA). This ultimately increases user trust and facilitates more informed decision-making when interpreting environmental impact data.

The screenshot in 3.4 shows the cases where the quantity is retrieved directly from the quantity / variable quantity associated with an activity or when there is a variables formula used for that.

3.4.3 Variables config manager

The migration of the Variable Configuration Manager, to React is showed in Fig 3.5 . The `VariableConfigView` component orchestrates data handling through Material-UI's DataGrid, enabling efficient visualization, filtering, sorting, and selection of large datasets. Complementing this, the `VariableConfigToolbar` introduces import features, including template access via Google Spreadsheets and

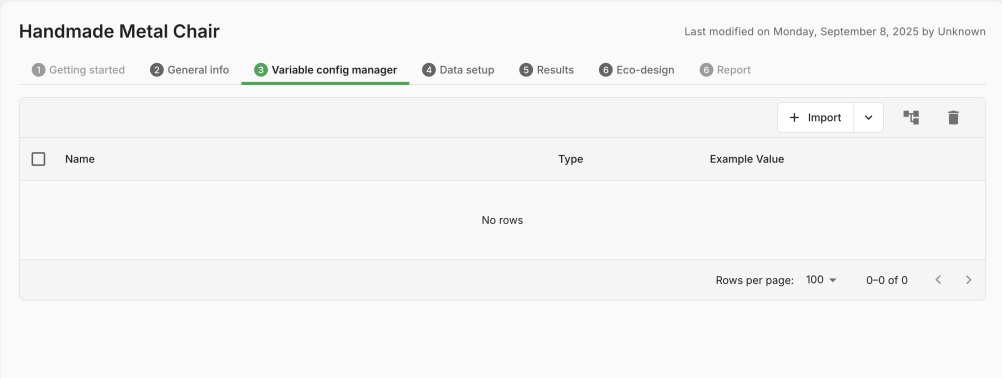


Figure 3.5: Screenshots of the variable config manager tab

CSV/Excel integration through the OneSchema service[26]. User interactions are enriched with modal-based workflows: `DiagramView` generates interactive dependency graphs via Viz.js and GraphViz (showed in Fig 3.6), `ImportFromFile` manages complex data transformations using OneSchema[26] (showed in Fig 3.7), and `ImportFromTemplate` synchronizes directly with predefined Google Spreadsheet templates. The template import process, showed in Fig 3.8, follows a structured

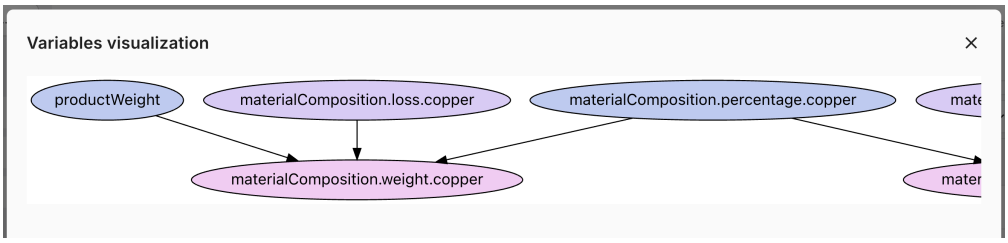


Figure 3.6: Screenshots of the variables diagram generation

three-phase pipeline: column mapping through `getColumnsMatching`, row normalization through `getMatchedRows`, and real-time data fetching from Google Sheets using spreadsheet ID. Imported data are strongly typed into `VariableResource` objects across five supported types (numberInput, calculated, textInput, assumption, enum), with validation. State management relies on React hooks and the SWR mutation system for optimistic updates, while internationalization supports five languages (English, French, Spanish, German, and Dutch) to ensure accessibility. Security is enforced by restricting access to Greenly Pro users, and error handling provides user-friendly messaging.

On the backend, the `EntityVariableController` manages schema persistence, while the new implemented `GoogleSheetsController` provides REST endpoints (e.g., `GET /cloud/google-sheets/{spreadsheetId}`) to fetch and synchronize

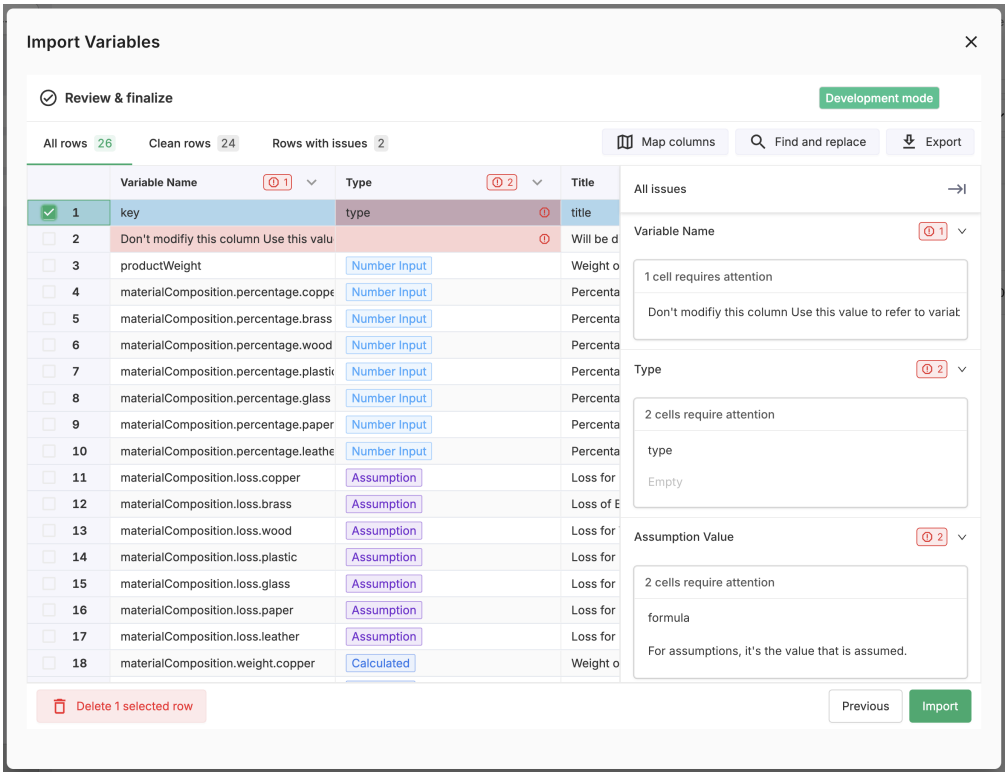


Figure 3.7: Screenshots of the variables importer trough OneSchema

structured data from Google Sheets. This is enabled by the `GoogleSheetsManager` class, which handles authenticated access to the Google Sheets API v4, error handling, and batch updates.

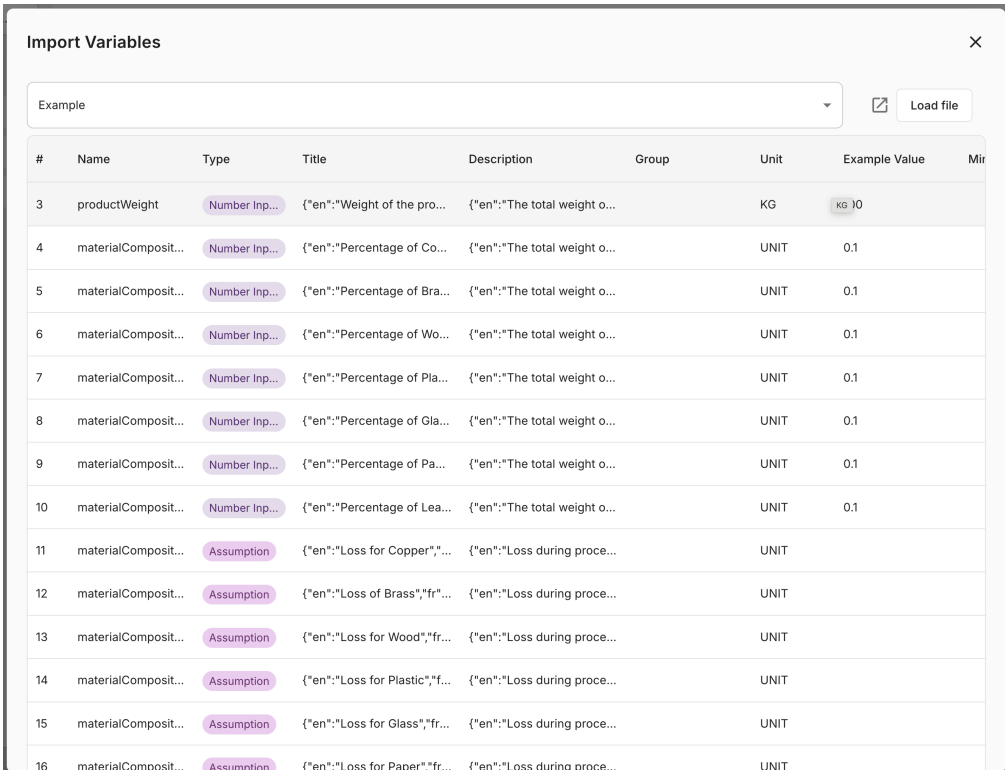


Figure 3.8: Screenshots of the variables table while importing from the shared spreadsheet

Chapter 4

Conclusions

This thesis has demonstrated how standardized Life Cycle Assessment methodologies can be effectively translated into scalable software systems that make environmental impact evaluation accessible to organizations.

Working with Greenly's platform, I implemented features across Node.js, Python, and React codebases that enhance the accuracy, usability, and compliance capabilities of LCA tools. Key contributions include multi-material tracking for product assessments, automated ISO 14040 report generation, dynamic emissions simulation, expanded impact categories beyond carbon footprint, and a modernized user interface with transparent calculation methodologies.

The primary challenges encountered were mastering the complex domain of environmental science and LCA regulations alongside Greenly's diverse technology stack, while consistently ensuring that every implementation prioritized code optimization, reusability, and maintainability to support the platform's scalability.

The solutions developed substantially reduce the manual effort required for comprehensive environmental assessments while improving their scientific rigor. As regulatory frameworks are starting to make sustainability reporting increasingly mandatory, platforms that democratize access to rigorous LCA methodologies will play a crucial role in enabling organizations of all sizes to measure, manage, and reduce their environmental impact in the collective effort to address climate change.

Bibliography

[1] World Resources Institute (WRI) and World Business Council for Sustainable Development (WBCSD). *Greenhouse Gas Protocol*. https://ghgprotocol.org/product-standard. Accessed: 2025-07-05. 2023 (cit. on pp. ii, 1, 6).

[2] *Environmental Management — Life Cycle Assessment — Principles and Framework (ISO 14040:2006)*. Last reviewed and confirmed in 2022; accessed via ISO Online Browsing Platform. International Organization for Standardization, 2006. URL: https://www.iso.org/standard/37456.html (cit. on pp. ii, 1, 4, 7).

[3] *Environmental Management — Life Cycle Assessment — Requirements and Guidelines (ISO 14044:2006)*. Last reviewed and confirmed in 2022; accessed via ISO Online Browsing Platform. International Organization for Standardization, 2006. URL: https://www.iso.org/standard/38498.html (cit. on pp. ii, 1, 4, 7).

[4] TypeORM Contributors. *TypeORM — An ORM for TypeScript and JavaScript*. Accessed: 2025-06-22. 2024. URL: https://typeorm.io/ (cit. on pp. ii, 12).

[5] TSOA Community. *TSOA — TypeScript OpenAPI Framework*. Accessed: 2025-06-22. 2024. URL: https://tsoa-community.github.io/docs/introduction.html (visited on 06/22/2025) (cit. on pp. ii, 12).

[6] Min Liu, Guangyan Zhu, and Yajun Tian. «The historical evolution and research trends of life cycle assessment». In: *Green Carbon* 2.4 (2024), pp. 425–437. ISSN: 2950-1555. DOI: https://doi.org/10.1016/j.greenca.2024.08.003. URL: https://www.sciencedirect.com/science/article/pii/S2950155524000727 (cit. on pp. 1, 3, 5).

[7] Wikipedia contributors. *Life-Cycle Assessment*. Accessed: 5 July 2025. 2025. URL: https://en.wikipedia.org/wiki/Life-cycle_assessment (cit. on pp. 1, 2, 6).

[8] Hari Srinivas. *ISO 14040:2006 – Life Cycle Assessment: Principles and Framework*. https://gdrc.org/uem/eia/ISO-14040.html. Accessed: 5 July 2025. GDRC, 2006 (cit. on pp. 1, 2, 4, 6, 19).

[9] The International EPD System. *EPD Applications*. Accessed July 6, 2025. 2024. URL: https://www.environdec.com/all-about-epds/epd-applications (cit. on pp. 1, 4, 7).

[10] Kara Anderson. «Everything You Need to Know About LCA (Life Cycle Analysis)». In: *Greenly Blog* (Oct. 2024). Accessed: 5 July 2025. URL: https://greenly.earth/en-gb/blog/company-guide/everything-you-need-to-know-about-lca-life-cycle-analysis-in-2022 (cit. on p. 3).

[11] Ecoinvent Association. *Ecoinvent Database*. Accessed: 2025-08-26. 2024. URL: https://ecoinvent.org (cit. on p. 3).

[12] ADEME. *Agribalyse Database*. Accessed: 2025-08-26. 2024. URL: https://agribalyse.ademe.fr (cit. on p. 3).

[13] ADEME. *ADEME Environmental Data and Databases*. Accessed: 2025-08-26. 2024. URL: https://base-empreinte.ademe.fr (cit. on p. 3).

[14] *Environmental labels and declarations — Type III environmental declarations — Principles and procedures (ISO 14025:2006)*. Last reviewed and confirmed in 2022; accessed via ISO Online Browsing Platform. International Organization for Standardization, 2006. URL: https://www.iso.org/standard/38131.html (cit. on pp. 4, 6).

[15] International Organization for Standardization. *ISO 14067:2018 Greenhouse gases – Carbon footprint of products – Requirements and guidelines for quantification*. 2018. URL: https://www.iso.org/standard/71206.html (cit. on pp. 4, 7).

[16] Greenly. *What is the Greenhouse Gas Protocol?* Accessed: 2025-06-28. 2022. URL: https://greenly.earth/en-gb/blog/company-guide/what-is-the-greenhouse-gas-protocol (cit. on p. 6).

[17] European Commission. *The Product Environmental Footprint (PEF) Method*. Accessed July 6, 2025. 2024. URL: https://green-forum.ec.europa.eu/environmental-footprint-methods/pef-method_en (cit. on p. 7).

[18] Kara Anderson. *Understanding a Product Carbon Footprint (PCF)*. Updated October 30, 2024; accessed 2025-09-08. 2024. URL: https://greenly.earth/en-us/blog/company-guide/understanding-a-product-carbon-footprint-pcf (cit. on p. 7).

[19] Wikipedia. *LCA — Life Cycle Analysis*. Accessed: 2025-06-22. 2024. URL: https://en.wikipedia.org/wiki/Life-cycle_assessment (visited on 06/22/2025) (cit. on p. 15).

[20] Dolan Miu. *docx: A JavaScript/TypeScript Library for Creating .docx Files*. https://www.npmjs.com/package/docx. Version 9.5.1; supports Node.js and browsers; declarative API (cit. on p. 19).

[21] Luzmo. *Luzmo: Embedded Data Analytics Platform*. `https://luzmo.com`. Accessed: 2025-07-13 (cit. on p. 19).

[22] Meta Platforms, Inc. *React – A JavaScript library for building user interfaces*. `https://react.dev`. 2025 (cit. on p. 26).

[23] Vercel Inc. *Next.js – The React Framework for Production*. `https://nextjs.org`. 2025 (cit. on p. 26).

[24] zod. *Zod Documentation*. `https://zod.dev/`. Accessed: 2025-08-10 (cit. on p. 27).

[25] next-intl. *next-intl Documentation*. `https://next-intl-docs.vercel.app/`. Accessed: 2025-08-10 (cit. on p. 27).

[26] OneSchema. *OneSchema – AI-Powered CSV, Excel, and PDF Import Platform*. `https://www.oneschema.co/`. Embeddable importer for SaaS, supports intelligent mapping, validation, and no-code transformations. Accessed: 2025-09-08. 2025 (cit. on p. 29).