DATA for GOOD's digital collaboration structure -

Technical documentation

1 - Getting started 3
    1.1 - About Data for Good 3
    1.2 - The Crane project 3
    1.3 - The different components of the ecosystem 3
    1.4 - Requirements to join the ecosystem 5
2 - Technical architecture 6
    2.1 - Core layers of the architecture 6
    2.2 - MitID version 8
3 - Integration 8
    3.1 - Level 1: Sending a copy of the consents to the blockchain 9
    3.2 - Level 2: Allowing users to store their own consent via SSO / Oauth 9
        3.2.1. Oauth2 (connecting accounts) 9
        3.2.2. SSO 9
    3.3 - Level 3: Moving data between services 9
4 - Performing a privacy-preserving analysis 11

1 - Getting started

1.1 - About DATA for GOOD

DATA for GOOD (DfG) is a not-for-profit organisation working to build trust in the data economy by promoting strong governance, privacy, and secure data management. In close collaboration with its main technology partner, Partisia, DfG uses cutting-edge blockchain and Multiparty Computation (MPC) technologies to deliver the infrastructure and governance needed for the ethical and secure use of data across sectors - with personal GDPR regulated data as a key focus, but with bigger ambitions.

1.2 - The Crane project

The Crane Project is a European-funded initiative aiming to revolutionize personal health data management through the creation of a secure, interoperable digital health ecosystem. Crane empowers citizens by enabling secure, consent-based control of their health information, fostering enhanced trust, privacy, and innovation in data-driven healthcare services.

1.3 - The different components of the digital collaboration structure

The DfG digital collaboration structure comprises several interconnected stakeholders and components that together enable secure, consent-driven management of personal GDPR regulated data.

- DfG Personal

This component provides citizens with control over their personal GDPR regulated data through a secure digital Personal Data Space ('wallet'). It enables connection to external data sources, consent management, and includes a Personal Data Space (PDS) for safe data storage and management.

- DfG Professional platform

This component includes dashboards specifically designed for professionals and service providers to efficiently manage user consents and a confidential computing platform that leverages Multiparty Computation (MPC). The consent dashboard is designed to ensure the privacy and security of citizen data, while the confidential computing platform can be utilized by companies, organizations, and individual analysts for performing secure, privacy-preserving data analyses.

- Service and Data Providers for Citizens

These are services that are integrated within the digital collaboration structure. Service providers provide transparency for users by storing consents in the blockchain-based ledger, leverage our SSO or Oauth2.0 for giving users ownership of their data without the need for multiple accounts, plus are enabled to share and fetch data from other service providers in the ecosystem, as well as directly through the data sources integrated with DfG Personal.

As of today, DfG has two main partners in this category: Meteda (based in Italy) and Cardiolyse (based in Finland).

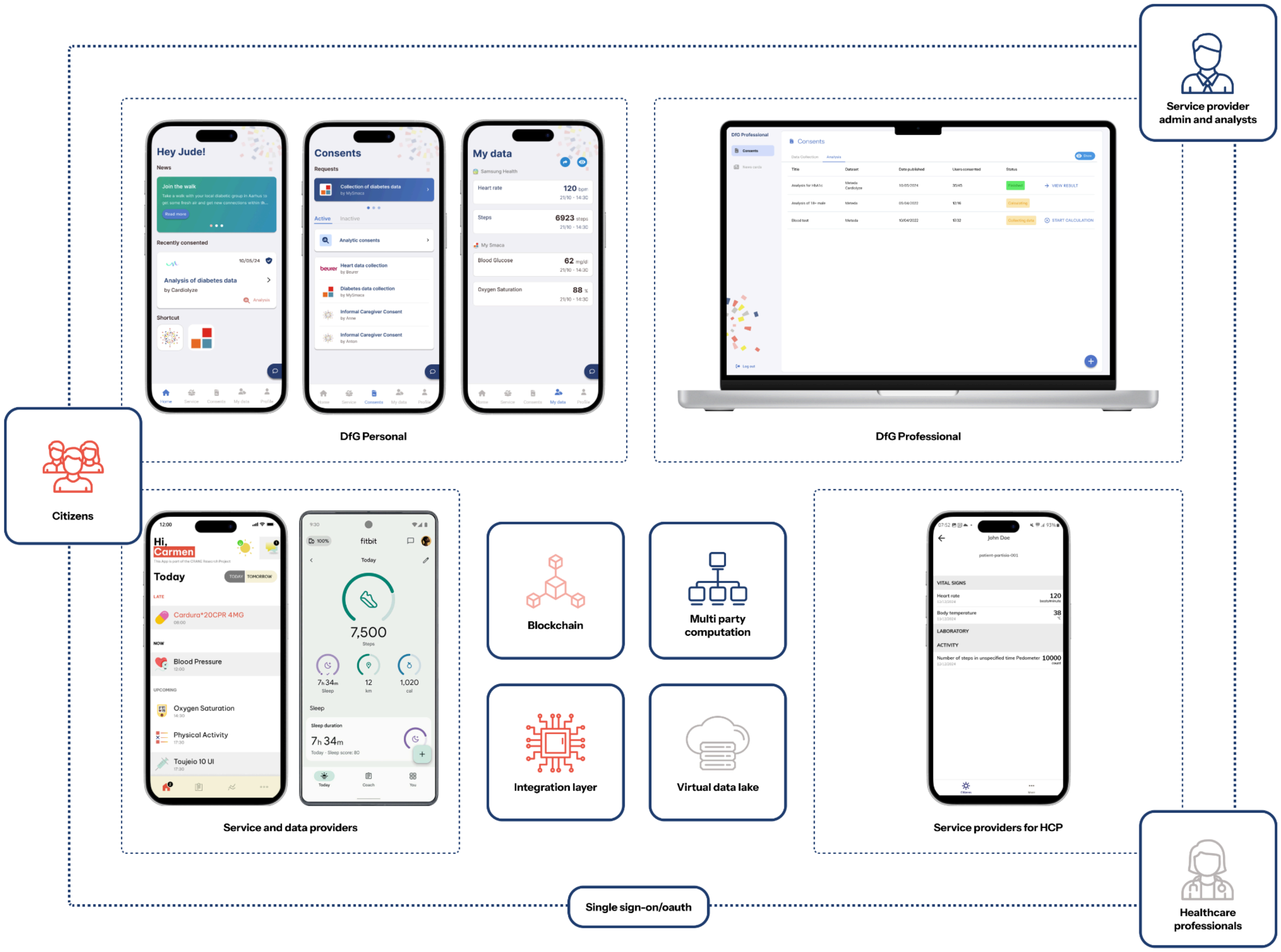


Figure 1. The DfG digital collaboration structure at a glance

- Service Providers for HealthCare Professionals (HCPs)

These include dedicated platforms and tools designed for healthcare professionals to optimize chronic disease management, clinical decision-making, and patient monitoring, enhancing overall patient care through comprehensive data integration and analytics.

As of today, DfG has two main partners in this category: Enversion (based in Denmark) and Meteda (based in Italy).

1.4 - Requirements to join the digital collaboration structure

Integrating with the DfG digital collaboration structure comes with different requirements depending on the level of integration a collaborating partner is aiming for. We currently offer three different integration levels:

- **Level 1**: Send consent records to the blockchain to provide transparency for the users of the specific Service.
- **Level 2**: Allow users to manage and store their own consents using SSO/OAuth, effectively putting them in control of their data.
- **Level 3**: Enable data transfer between services (full service provider integration) for enhancing the experience of users.

Regardless of integration level, general technical requirements include:

- A backend capable of interacting with the blockchain-based consent APIs.
- The ability to send blockchain transactions (no gas fees required). If your team doesn't have prior blockchain experience, our team will gladly support you through your first blockchain experience.
- Ideally, compliance with OAuth2 standards for user identity and consent management.

2 - Technical architecture

2.1 - Core layers of the architecture

The DfG digital collaboration structure is built on a layered architecture designed to ensure security, interoperability, privacy, and scalability. This structure supports consent-driven data management, secure multi-party computation, and ease of integration across stakeholders. Below is a high-level overview of the key layers:

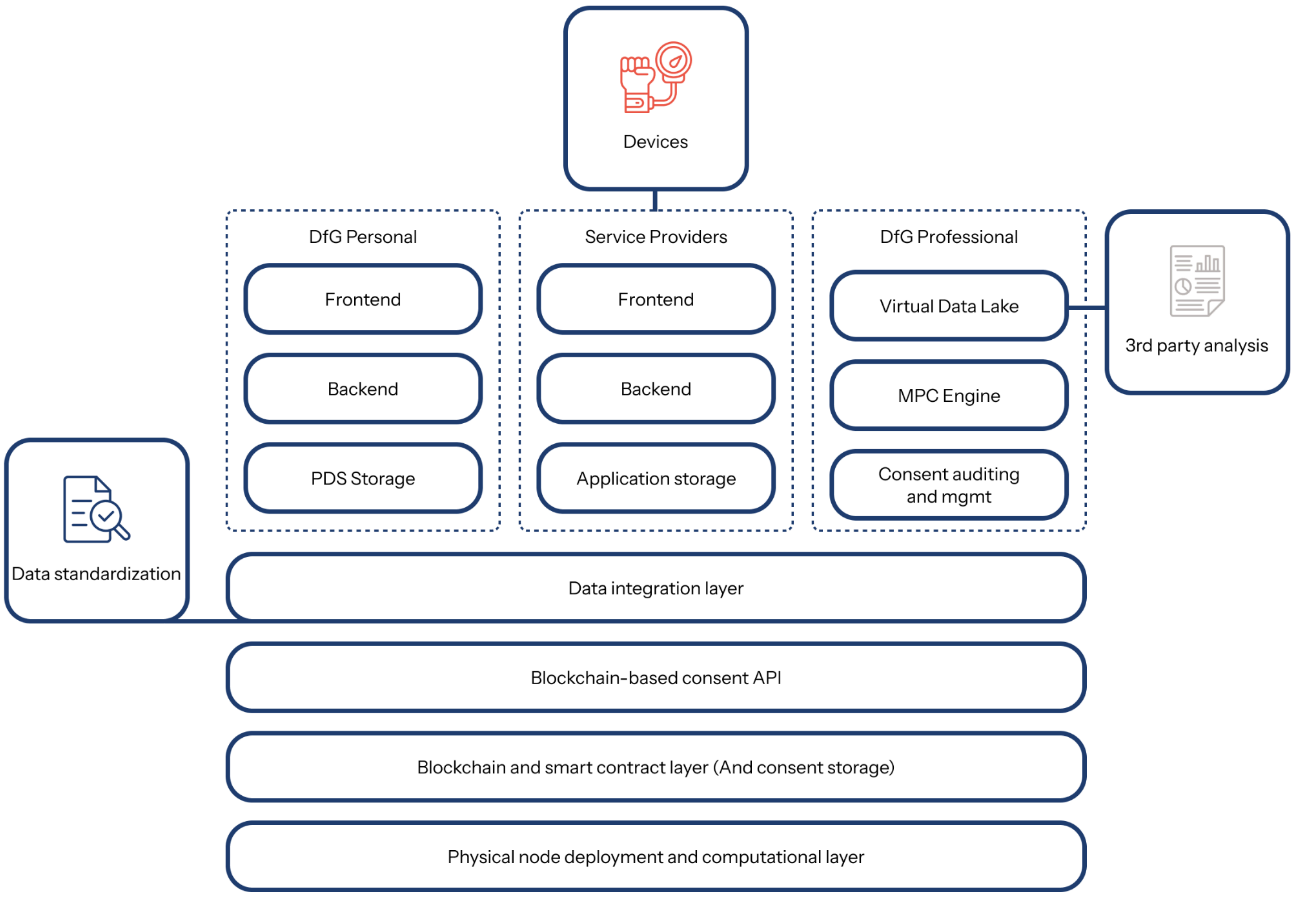


Figure 2. DfG's digital collaboration structure technical architecture

- Physical node deployment and computational layer:

This foundational layer includes physical servers (nodes) spread across providers, along with the MPC cluster for privacy-preserving analysis. These nodes manage encryption, data processing, and secure calculations without storing sensitive data centrally.

- Blockchain and smart contract layer:

Built on Partisia's blockchain technology, this layer works as a consent storage and auditing layer. Smart contracts enforce rules for consent creation, revocation, and verification. All consent transactions are recorded immutably on the blockchain, providing transparency and tamper-proof logs.

- Blockchain-based consent API layer:

This API serves as the interface for managing consents. It allows service providers to interact with the blockchain for storing, retrieving, and auditing consents. The API supports operations like consent submission and checks, ensuring compliance with GDPR and user control.

- Data integration layer:

This layer facilitates the standardisation of data between services and external data sources. For example in the health industry, we're relying on our partner Enversion to provide the Journl Stream solution, which orchestrates the process of converting health data into standardised FHIR data. You can read more about the specifics of the integration layer here.

- Application layer:

This top layer includes tools provided by the DfG digital collaboration structure: the consent dashboards and the Confidential Computing platform (for professionals), the Virtual Data Lake (for data exchange and privacy-preserving analytics), and end-user apps like DfG Personal (for citizens) and DfG Professional platform (for HCPs and analysts).

For more details, refer to our documentation on consent APIs (e.g., https://docs.partisia.com/consent/consent.html).

2.2 - MitID version

In regions like Denmark, a specialized version integrates with MitID (national digital ID) for authentication. This adds an extra layer of trust by leveraging government-backed identity verification. It includes tailored consent mechanisms but maintains the same core layers, ensuring compatibility with the broader ecosystem.

3 - Integration

Joining the DfG digital collaboration structure lets Service Providers build user trust, enable easy data sharing, and join a privacy-focused network. Integrations start simple and grow more advanced.

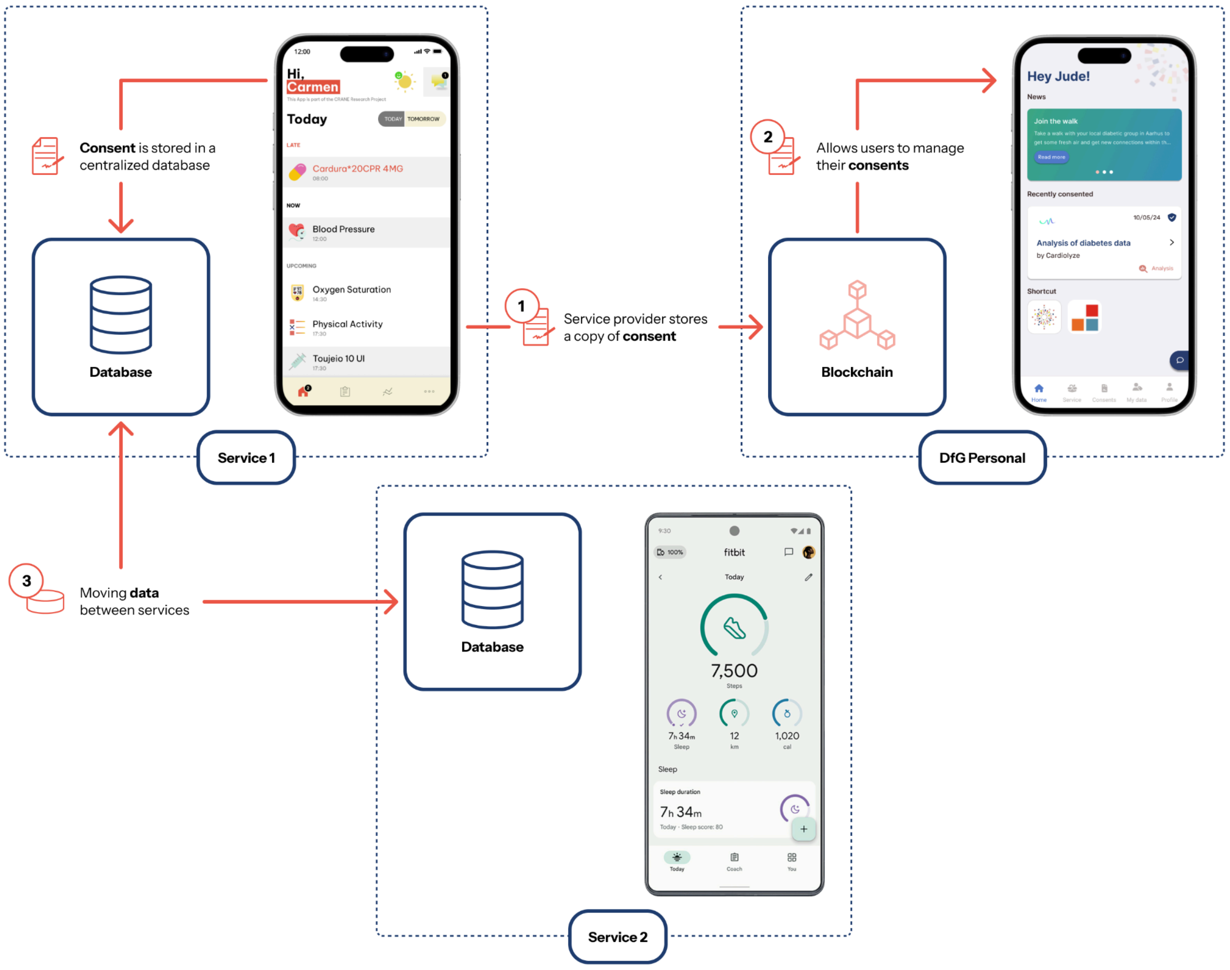

Consent is stored in a centralized database
Database
Service 1
1
Service provider stores a copy of consent
2
Allows users to manage their consents
Blockchain
DfG Personal
3
Moving data between services
Database
Service 2

Figure 3. The 3 different levels of integration with the DfG digital collaboration structure

3.1 - Level 1: Sending a copy of the consents to the blockchain

This entry-level integration provides transparency by storing consent records on the blockchain, earning a DfG Compliance Badge. Users do not need to be DfG users - your service can operate independently.

- **Steps**:
  - Integrate with the Consent API.
  - When a user gives consent in your app, send a transaction to store a copy (or hash) on the blockchain (see https://docs.partisia.com/consent/how-to-build-a-transaction.html).
- **Requirements**: Basic API access; no user authentication changes needed.
- **Benefits**: Immutable audit log; display DfG badge for trust.

- **Example**: A well-being app sends consent proofs to demonstrate ethical handling of data.

3.2 - Level 2: Allowing users to store their own consent via SSO / Oauth

This level puts users in control, allowing them to manage consents directly via DfG Personal. Users will become DfG users upon connection.

3.2.1. Oauth2 (connecting accounts)

Use OAuth2 to link user accounts between your service and DfG.

- **Steps**: Implement OAuth flow. DfG gets the user's ID for consent syncing.
- **Requirements**: Your backend must handle user ID exchange through OAuth2.
- **Example**: User logs into your app, connects to DfG, and consents are mirrored.

3.2.2. SSO

For seamless authentication, service providers can implement DfG's Single Sign-On (SSO).

- **Steps**: Follow Crane's SSO setup. Users log in once via DfG Personal or your app.
- **Requirements**: Your backend must handle JWT tokens for user data.
- **Benefits**: Simplified user journeys; enables full consent control through DfG's identity management.

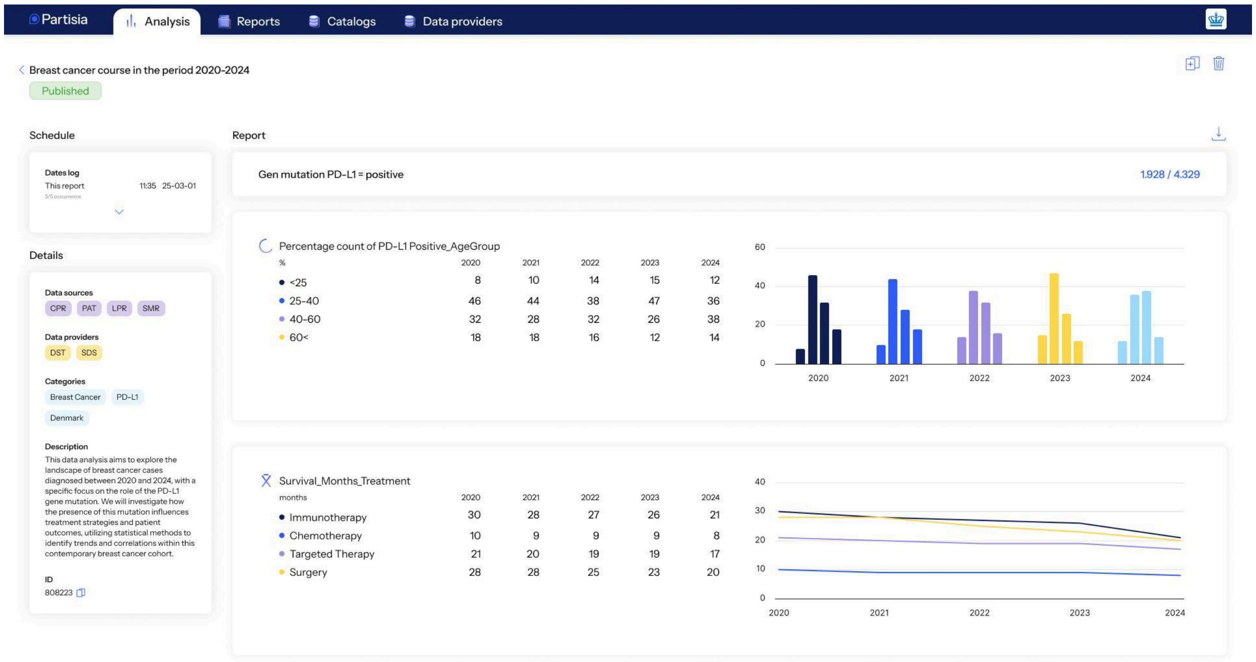
3.3 - Level 3: Moving data between services

This full integration enables secure data sharing across the digital collaboration structure (e.g., from your own service from/to any other service in the collaboration structure), using the integration layer for standardisation of data and relying on the blockchain-based ledger for ethical handling of data.

- **Steps**: Check consents via API endpoints. Use the standardisation layer to ensure data is interoperable between services. Data flows to/from Virtual Data Lake to services if consented.
- **Requirements**: A backend for performing consent checks and data export.
- **Benefits**: Enhanced interoperability between services; additional features can be built.
- **Example**: Share your Fitbit data with your Healthcare provider.

4 - Performing a privacy-preserving analysis

The DfG digital collaboration structure provides an Analytics-as-a-Service tool called the Confidential Computing platform - which uses a powerful encryption method named Multiparty Computation (MPC) to combine sensitive datasets in a privacy-preserving manner.



Breast cancer course in the period 2020-2024

Published

Schedule

Dates log

This report 11:35 25-03-01

5/5 occurrence

Details

Data sources: CPR PAT LPR SMR

Data providers: DST SDS

Categories: Breast Cancer PD-L1 Denmark

Description

This data analysis aims to explore the landscape of breast cancer cases diagnosed between 2020 and 2024, with a specific focus on the role of the PD-L1 gene mutation. We will investigate how the presence of this mutation influences treatment strategies and patient outcomes, utilizing statistical methods to identify trends and correlations within this contemporary breast cancer cohort.

ID

808223

Report

Gen mutation PD-L1 = positive 1.928 / 4.329

Percentage count of PD-L1 Positive_AgeGroup

%	2020	2021	2022	2023	2024
● <25	8	10	14	15	12
● 25-40	46	44	38	47	36
● 40-60	32	28	32	26	38
● 60<	18	18	16	12	14

Survival_Months_Treatment

months	2020	2021	2022	2023	2024
● Immunotherapy	30	28	27	26	21
● Chemotherapy	10	9	9	9	8
● Targeted Therapy	21	20	19	19	17
● Surgery	28	28	25	23	20

Figure 4. The Confidential Computing platform

Organizations, universities, companies, and individuals can securely analyze aggregated data sets without exposing sensitive individual data, thus, extracting insights from sensitive data while keeping it safe and private. The full documentation for the Confidential Computing product can be found here.

To perform privacy-preserving analyses, the following steps are necessary:

1) Analysts request consent through the consent dashboard, clearly defining data types and intended analysis.
2) Citizens receive consent requests via their DfG Personal application, where they can review and approve or decline consent.
3) Upon consent approval, analysts securely combine data from various sources by leveraging Multi-Party Computation and can query on the different datasets.

The available datasets come from a variety of Service Providers, such as Meteda and Cardiolyse and data sources, such as Fitbit, and include health metrics (e.g., blood pressure, ECG readings, heart rate variability), activity data from wearables, and more.

If you're interested in having access to the Confidential Computing platform, you can reach out at info@dfgfoundation.org.

4.1 - Multi-Party Computation (MPC) API

The following GraphQL code snippets describe the fields that service providers must fill in to build their own frontend for the Confidential Computing Platform.

GraphQL is a query language for APIs and a runtime that lets clients ask for exactly the data they need - nothing more, nothing less. Instead of many REST endpoints, a GraphQL API exposes a single endpoint backed by a strongly-typed schema.

```
None
type Query {
  analysis(id: ID!, orgId: Int): Analysis! # (Shared coordinator and node)
  analyses(analysisFilter: AnalysisFilter!): AnalysesConnection! # (Shared
coordinator and node)
  dataView(id: ID! orgId: Int): DataView!
  datasource(id: ID!): Datasource!
  catalog(id: ID!): Catalog!
  datasources(input: CatalogsAndDatasourcesQueryInput): [Datasource!]!
  catalogs(input: CatalogsAndDatasourcesQueryInput): [Catalog!]!
  binder(analysisId: ID!): Binder
  result(analysisId: ID! orgId: Int signedRequests: [[String!]!]! taskIds:
[Int!]!): Analysis!
  node(id: ID!, orgId: Int): Node # (Shared in coordinator and node)
}
```

```
None
type Mutation {
  # Coordinator:
  createCatalog(signedTransaction: String!): Catalog!
  createDatasource(signedTransaction: String!): Datasource!
  createVariable(signedTransaction: String!): Variable!
  updateCatalog(signedTransaction: String!): Catalog!
  updateDatasource(signedTransaction: String!): Datasource!
  updateVariable(signedTransaction: String!): Variable!
  createDataView(signedTransaction: String!): DataView!
  createAnalysis(signedTransaction: String!): Analysis!
  updateAnalysis(id: ID! signedTransaction: String!): Analysis!
  deleteAnalysis(signedTransaction: String!): Boolean!
  updateMergeSpecification(id: ID!, signedTransaction: String!): Analysis!
  createCalculation(signedTransaction: String!): Calculation!
  updateCalculation(id: ID! signedTransaction: String!): Calculation!
  duplicateCalculation(id: ID! signedTransaction: String!): Calculation!
  deleteCalculation(signedTransaction: String!): Boolean!
  addCatalog(analysisId: ID! signedTransaction: String!): Analysis!
  removeCatalog(analysisId: ID! signedTransaction: String!): Analysis!
  addDatasource(analysisId: ID! signedTransaction: String!): Analysis!
  removeDatasource(analysisId: ID! signedTransaction: String!): Analysis!
  sendDataView(signedTransaction: String!): DataView!
  sendAnalysis(signedTransaction: String!): Analysis!
  duplicateDataView(signedTransaction: String!): DataView!
  duplicateAnalysis(signedTransaction: String!): Analysis!
  queueAnalysisForComputation(signedTransaction: String!): Analysis!

  # CC Nodes:
  approveAnalysis(orgId: Int! signedTransaction: String!): Analysis!
  approveResults(orgId: Int! signedTransaction: String!): Analysis!
  requestResultAccess(signedAccess: String!): Binder!
}
```

```None
type Analysis implements Node {
  id: ID! @globalId
  name: String # Shared
  note: String # Shared
  analysts: [String!]!
  calculations: [Calculation!]! # Shared
  catalogs: [Catalog!]! # Shared
  datasources: [Datasource!]! # Shared
  approvals: [OrganizationApproval!]
  status: AnalysisStatus!
  locked: Boolean!
  results: [ResultsDto!]
  modulus: String! # Shared
  fixedPointPrecision: Int! # Shared
  dataViewId: Int!
  # From Nodes (not coordinator)
  createTs: TIME!
  modifyTs: TIME!
  analystName: String
  analystEmail: String!
  approvalState: ApprovalState!
  resultsApprovalState: ResultsApprovalState!
  stateStatus: AnalysisStateStatus!
  ownDataTs: TIME
  otherDataTs: TIME
  processed: Boolean!
  confidentialDataset: String
  finishedTasks: [Int!]
  offChainAuthorities: [String!]
  datasetDefinitions: [DatasetDefinition!]!
  trusted: Boolean
  mergeSpec: String!
}
```

Additional documentation of the Confidential Computing platform can be found here.