# Behandling av personuppgifter i Lärio

Teknisk och organisatorisk redogörelse enligt dataskyddsförordningen (EU) 2016/679, med referenser till programmets källkod.

| | |
|---|---|
| Dokumenttyp | Teknisk beskrivning, underlag för personuppgiftsansvarig och dataskyddsombud |
| Avser | Lärio, lokalt installerat rättnings- och bedömningsstöd för Windows |
| Rättslig referensram | Förordning (EU) 2016/679 (GDPR), särskilt artiklarna 4, 5, 9, 25, 28, 30, 32 och 44–49. Lagen (2018:218) med kompletterande bestämmelser till EU:s dataskyddsförordning. Skollagen (2010:800) 3 kap. om betyg och dokumentation. |
| Avser programversion | 2.16.0 och senare. Versionen visas i programmets nedre högra hörn. |
| Senast ändrad | 9 september 2026 |
| Verifierbarhet | Samtliga påståenden i avsnitt 5–7 är kopplade till namngivna filer och funktioner i källkoden och till automatiska prov som körs vid varje ändring. Se avsnitt 9. |

## 1 Dokumentets syfte och avgränsning

Detta dokument beskriver hur programvaran Lärio behandlar personuppgifter, vilka tekniska åtgärder som vidtagits enligt artikel 32 i dataskyddsförordningen, och hur dessa åtgärder kan verifieras av en utomstående granskare.

Dokumentet är avsett att kunna läggas till grund för en personuppgiftsansvarigs bedömning inför införande av programvaran i en skolorganisation. Det ersätter inte huvudmannens egen konsekvensbedömning, och det utgör inte juridisk rådgivning.

Beskrivningen avser den lokalt installerade Windows-versionen. Lärio saknar serverkomponent, användarkonto och central lagring; det finns således ingen molntjänst att beskriva.

## 2 Sammanfattning för dataskyddsombud

> Elevtexten pseudonymiseras **på lärarens egen dator** innan någon del av den lämnar maskinen. Direkta identifikatorer ersätts av platshållare; särskilda kategorier av personuppgifter enligt artikel 9 **avlägsnas helt**, liksom uppgifter som pekar ut eleven utan att vara personuppgifter i sig. En spärr i nätverkslagret vägrar genomföra anropet om pseudonymiseringen skulle ha misslyckats.
>
> Termen är vald med avsikt. Europeiska dataskyddsstyrelsen är uttrycklig om att pseudonymiserade uppgifter förblir personuppgifter så länge någon med rimliga medel kan knyta dem till en person. Lärio gör därför inte anspråk på att texten efter filtret saknar personuppgifter, utan på att de identifierande

> uppgifterna är ersatta eller borttagna innan överföring, att nyckeln aldrig lämnar datorn och att den inte ens sparas.

| FRÅGA | SVAR |
|---|---|
| Finns en molntjänst eller ett konto? | Nej. Programmet körs lokalt. Det finns ingen inloggning, ingen server och ingen central databas. |
| Vilka mottagare finns? | En enda: Google (Gemini API), för själva språkanalysen. Inga analysverktyg, inga typsnittstjänster, ingen felrapportering till tredje part. |
| Vad överförs till mottagaren? | Den avidentifierade elevtexten, uppgiftens instruktion och de källor läraren själv bifogat. Inga namn, inga filnamn, inga kontaktuppgifter, inga känsliga uppgifter. |
| Var lagras elevmaterialet? | I en mapp bredvid programfilen på lärarens dator. Leverantören har ingen åtkomst till den. |
| Kan leverantören läsa elevtexter? | Nej. Det finns ingen teknisk väg för det: programmet har ingen kanal till leverantören. |
| Vad händer om skyddet går sönder? | Avidentifieringen har ett golv. De delar som kan fallera är byggda så att de bara kan lägga till maskeringar, aldrig ta bort en; går en av dem sönder kopplas den bort ensam och grundskyddet står kvar. Läraren ser vilket läge som gällde. |
| Går skyddet att kontrollera? | Ja. Se avsnitt 9. Programmet innehåller automatiska prov som mäter vad som faktiskt lämnar processen. |

## 3 Roller och ansvar

Skolhuvudmannen är **personuppgiftsansvarig** för behandlingen av elevernas uppgifter. Läraren behandlar uppgifterna inom ramen för sin anställning.

Leverantören av Lärio är **inte personuppgiftsbiträde** i förhållande till elevmaterialet, eftersom leverantören varken tar emot, lagrar eller på annat sätt behandlar detta material. Programvaran levereras som en produkt, inte som en tjänst. Något biträdesavtal mellan huvudmannen och leverantören avseende elevtexter är därför inte tillämpligt.

Google är mottagare av den *avidentifierade* texten. Huvudmannen bör självständigt bedöma detta förhållande. Se avsnitt 7.

## 4 Behandlade uppgiftskategorier

| KATEGORI | LAGRAS LOKALT | ÖVERFÖRS |
|---|---|---|
| Elevtext i sakinnehåll | Ja, i avidentifierad form | Ja, avidentifierad |
| Elevens namn | Endast om läraren själv skriver in det i arkivet | **Nej** |
| Personnummer | Nej | **Nej** |
| Kontaktuppgifter | Nej | **Nej** |
| Adress, skola, klass | Nej | **Nej** |
| Filnamn på inlämnad fil | Ja | **Nej** |
| Särskilda kategorier (art. 9) | Nej | **Nej – avlägsnas helt** |
| Lärarens API-nyckel | Ja, krypterad | Ja, som autentisering mot Google |

## 5 Tekniska åtgärder enligt artikel 32

### 5.1 Pseudonymisering före överföring

Artikel 4.5 definierar pseudonymisering som behandling av personuppgifter så att de inte längre kan tillskrivas en registrerad utan kompletterande uppgifter. Lärio genomför denna behandling lokalt, innan nätverksanropet konstrueras.

Direkta identifikatorer ersätts av numrerade platshållare i stället för att raderas. Valet är medvetet: en borttagen mening förändrar textens syntax, och bedömningen skulle då avse en annan text än den eleven skrivit.

`src/services/piiShield.js`

```
// "[NAMN-1] bor i [ORT-1] och går på [SKOLA-1]."
//
// Tar man BORT text i stället för att ersätta den försvinner
// meningsbyggnaden, och bedömningen avser då en annan text än
// den eleven lämnade in.
```

Följande kategorier identifieras och ersätts:

| KATEGORI | METOD |
|---|---|
| Personnummer | Strukturell matchning, med och utan sekelsiffra och bindestreck |
| Telefonnummer | Strukturell matchning av svenska nummerformat |
| E-postadresser | Strukturell matchning |
| Gatuadresser | Mönstermatchning på gatunamnsefterled följt av nummer |
| Personnamn | Kontextregler: `jag heter`, `mitt namn är`, `min lärare`, `mvh`, samt namnrader i sidhuvud |
| Skolnamn | Mönstermatchning på `-skolan`, `gymnasiet` med flera efterled |
| Klassbeteckningar | Mönstermatchning på svenska klasskoder |
| Orter | Kontextregler kring `bor i`, `kommer från`, `hemstad` |
| IP-adresser | Strukturell matchning med giltiga oktetter, skild från telefonnummerformat |

Engelska texter bedöms i samma program och har egna kontextregler för varje kategori ovan — `my name is`, `I live in`, `my teacher`. Utan dem hade en text på engelska fått en bråkdel av det skydd en text på svenska får.

**Samma uppgift, skriven på ett annat sätt**

En regel som söker efter `990101-1234` ser inte `９９０１０１-１２３４`, och en regel som söker efter *Anna* ser inte *Anna* med ett osynligt tecken efter första bokstaven. Arton sådana varianter prövades mot spärren den 8 september 2026. Nio tog sig igenom.

Ingen av dem kräver någon avsikt att kringgå skyddet. Osynliga tecken följer med när en elev klistrar in från en webbsida, fullbreddssiffror kommer från japanskt eller kinesiskt tangentbord, och mjuka bindestreck sätts in av ordbehandlaren själv. Det behövs alltså ingen angripare för att uppgifterna ska läcka — det räcker med en elev som klistrar in.

Detektionen sker därför inte i elevens text direkt, utan i en **analyskopia** där osynliga tecken är borttagna, homoglyfer och fullbreddstecken översatta ett till ett, och isärspärrade bokstäver sammanförda.

src/services/normalisering.js

```
// ORIGINALTEXTEN ÄNDRAS ALDRIG.
//
// Normaliseringen skapar en ANALYSKOPIA som reglerna får läsa. Det
// läraren och modellen får se är fortfarande elevens egen text, tecken
// för tecken - bara med de träffade partierna utbytta mot platshållare.
//
// Skulle den normaliserade texten skickas i stället vore skadan
// uppenbar: en elev som skrivit med fel tecken skulle få sin text tyst
// omskriven, och språket skulle bedömas på något hen inte skrivit.
//
// Därför bär varje analyskopia en KARTA tillbaka till originalet.
// Position 42 i kopian pekar ut exakt vilket tecken i originalet den kom
// ifrån, så att ersättningen träffar rätt även när kopian är kortare.
```

Att kartan finns är avgörande för bedömningens riktighet, inte bara för skyddet. Utan den skulle avidentifieringen träffa fel tecken i varje text som innehåller ett osynligt tecken, och eleven skulle bedömas på ord hen inte skrivit.

**Två lager, med det deterministiska som golv**

Avidentifieringen består av ett **grundskydd** och ett antal **utökade motorer**. Grundskyddet är regler i kod: det körs alltid, körs först, kräver ingen modell och tar omkring tjugo millisekunder. De utökade motorerna ser sådant som ingen enskild regel kan se — se 5.3.

Konstruktionen vilar på en enda garanti, och den är avsiktligt ensidig: en utökad motor kan **bara lägga till** maskeringar, aldrig ta bort en. Går en motor sönder — på en svag dator, vid slut minne, vid en oväntad text — kopplas just den motorn bort, medan grundskyddet står kvar oförändrat. Programmet kan därför inte hamna i ett läge där en trasig komponent ger sämre skydd än ingen komponent alls.

src/services/redactPii.js

```
// granska() är grundskyddet: regler i kod, 20 millisekunder, 67 av 67
// planterade uppgifter i provbanken. Det körs alltid och först.
//
// Ovanpå det kan en utökad motor läggas - en lokal modell som ser sådant
// regler inte kan se. Den kan bara LÄGGA TILL maskeringar, aldrig ta bort
// en. Finns ingen motor, eller går den sönder, blir resultatet exakt
// grundskyddet, och programmet fungerar som det gör i dag.
```

Läraren ser vilket läge som gällde för den enskilda texten, så att skillnaden mellan fullt skydd och enbart grundskydd inte är dold. Garantin är prövad i `test/run-utokad.js`, som bland annat låter en motor kasta fel, returnera skräp och försöka stryka en av grundskyddets träffar.

## 5.2 Särskilda kategorier av personuppgifter (artikel 9)

Artikel 9 räknar upp nio kategorier av personuppgifter som åtnjuter ett förstärkt skydd. Sådana uppgifter förekommer regelmässigt i elevtexter, eftersom eleven ofta skriver om sig själv. Samtliga nio är täckta, och ordlistorna i koden är uppställda efter förordningens egen uppräkning – en kategori per konstant, i samma ordning – så att en kategori som saknas syns som ett tomt hål i stället för att försvinna i en lång blandad lista.

| KATEGORI ENLIGT ARTIKEL 9 | KONSTANT I KODEN | EXEMPEL SOM AVLÄGSNAS |
|---|---|---|
| Etniskt ursprung | `K_ETNICITET` | ”Jag är kurd” |
| Politiska åsikter | `K_POLITIK` | ”Jag röstar på Vänsterpartiet” |
| Religiösa eller filosofiska uppfattningar | `K_RELIGION` | ”Jag är muslim” |
| Fackligt medlemskap | `K_FACK` | ”Jag är medlem i Kommunal” |
| Genetiska uppgifter | `K_GENETIK` | ”Jag är bärare av genen” |
| Biometriska uppgifter | `K_BIOMETRI` | ”Jag har lämnat fingeravtryck” |
| Hälsa | `K_HALSA` | ”Jag har adhd” |
| Sexualliv | `K_SEXUALLIV` | ”Jag är sexuellt aktiv” |
| Sexuell läggning | `K_LAGGNING` | ”Jag är homosexuell” |

Utöver de nio kategorierna behandlas asylskäl och uppehållsstatus på samma sätt ( `K_ASYL` ). De omfattas inte av artikel 9 men är minst lika utpekande i en elevtext.

För dessa uppgifter tillämpas **inte** pseudonymisering utan **radering**. Innehållet ersätts av en markering som visar att något avlägsnats, utan att avslöja vad. En platshållare av typen `[DIAGNOS-1]` skulle röja just det kategorin är till för att skydda.

src/services/piiShield.js

```
const K_ETNICITET = 'kurd|kurdisk|same|samisk|rom|romsk|assyrier|arab|somalier|…';
const K_POLITIK   = 'socialdemokrat|moderat|vänsterpartist|kommunist|feminist|…';
const K_RELIGION  = 'muslim|kristen|jude|hindu|buddhist|sikh|katolik|ateist|gud|…';
const K_FACK      = 'fackförbund|facket|if metall|lärarförbundet|medlem i kommunal|…';
const K_GENETIK   = 'gentest|genetisk|arvsanlag|kromosom|ärftlig|bärare av genen|…';
const K_BIOMETRI  = 'fingeravtryck|ansiktsigenkänning|irisskanning|biometrisk|…';
const K_HALSA     = 'autism|dyslexi|depression|ångest|epilepsi|diabetes|cancer|…';
const K_SEXUALLIV = 'sexliv|sexuellt aktiv|preventivmedel|p-piller|könssjukdom|…';
const K_LAGGNING  = 'homosexuell|bisexuell|lesbisk|queer|transperson|hbtq|…';

// ORDGRÄNSEN är avsiktligt inte \b. \b i JavaScript är ASCII-baserad, så
// \bångest matchar aldrig någonting: varken mellanslaget före eller "å"
// räknas som ordtecken. Lookaround på \p{L} gäller hela alfabetet.
const KANSLIGA_ORD = new RegExp(
  '(?<![\\p{L}\\p{N}])(?:' + [K_ETNICITET, K_POLITIK, /* … */].join('|') + ')'
  + '(?![\\p{L}\\p{N}])', 'iu');
```

Träffen kräver **två** saker: en bärare – skribenten eller någon i hens närhet – och ett verb som uttrycker tillhörighet, tillstånd eller övertygelse. Båda behövs, och det är en avsiktlig avvägning mot bedömningens kvalitet: ett ensamt känsligt ord är ett *ämne*. Meningen ”Adhd är vanligt i skolan” står kvar, eftersom en elev måste kunna skriva en text om adhd, islam eller hbtq utan att texten plockas isär. Ett skydd som gör ämnet oskrivbart är inte ett skydd.

src/services/piiShield.js

```
const KANSLIGT_RE = new RegExp(
  '\\b(?:jag|vi|min|mitt|mina|vår|vårt|våra)\\s+'
  + '((?:[\\p{L}-]+\\s+){0,2}?' + VERB_SV + SATS, 'giu');

// Bäraren får stå i tredje person: "min mamma är …", "min lillebror har …".
// Motvikten är två stopplistor - "min TEXT är om islam" och "jag är
// INTRESSERAD AV islam" säger ingenting om skribenten och står kvar.
for (const re of [KANSLIGT_RE, KANSLIGT_EN_RE]) {
  let hm;
  while ((hm = re.exec(kalla)) !== null) {
    const sats = hm[1];
    if (!sats || !arKansligUppgift(sats)) continue;
    lagg(hm.index + hm[0].indexOf(sats), sats.length,
         '[UPPGIFT OM ELEVEN BORTTAGEN]', 'KANSLIGT');
  }
}
```

Meningen *”Eftersom jag är muslim fastar jag under ramadan”* lämnar alltså datorn som *”Eftersom jag [UPPGIFT OM ELEVEN BORTTAGEN].”*

Engelska texter bedöms i samma program och har ett eget mönster ( `KANSLIGT_EN_RE` ). Det är skiftlägeskänsligt: engelskans *I* skrivs med versal, medan gement *i* är svenskans preposition, och ett skiftlägesokänsligt mönster hade läst ”i skolan är muslimer vanliga” som en uppgift om skribenten.

Skyddet är bundet till förordningens uppräkning genom ett automatiskt prov, `test/run-artikel-9.js` , som körs vid varje ändring. Provet kontrollerar dels att var och en av de nio kategorierna finns kvar i koden och innehåller minst ett ord, dels att uppgifterna faktiskt avlägsnas – och dels, med lika många kontroller, att ämnestexterna står kvar orörda.

## 5.3 Indirekt identifierbarhet och kvasi-identifierare

Avsnitten ovan söker efter *typer* av uppgifter: namn, personnummer, adress, diagnos, religion. Förordningens fråga i artikel 4.1 är bredare. Den gäller om informationen, ensam eller tillsammans med annan information som rimligen kan finnas, kan hänföras till en fysisk person. Europeiska dataskyddsstyrelsen framhåller uttryckligen att även uppgifter utanför datamängden — offentliga register, sociala medier — kan göra pseudonymiserade uppgifter hänförliga till en person.

Den frågan har inget svar i en ordlista. Meningen

> *”Min pappa driver den enda syriska restaurangen i byn där vi bor.”*

innehåller inget namn, inget personnummer, ingen adress och ingen uppgift enligt artikel 9. Den pekar ändå ut en bestämd familj för var och en som känner till orten.

**Vad som gör en sådan mening utpekande**

Inte antalet uppgifter. En text kan säga ”jag är arton år”, ”jag spelar fotboll” och ”min mamma är undersköterska” utan att peka ut någon — det gäller tiotusentals elever. Att stoppa den texten vore att göra personligt berättande omöjligt att lämna in, och personligt berättande är en egen genre i svenska som andraspråk.

Det som gör kombinationen identifierande är **entydigheten**: orden ”den enda”, ”ortens enda”, ”först i”, ”ensam om”. De förvandlar en vanlig uppgift till en unik. Lärio söker därför efter entydighetsmarkören, inte efter mängden vanliga fakta — och markören räknas bara när den är förankrad i en plats eller en grupp som är liten nog att räkna medlemmar i.

| KATEGORI | BEHANDLING |
|---|---|
| Entydighetsanspråk förankrat i en plats | Avlägsnas |
| Exakt födelsedatum (dag, månad, år) | Avlägsnas |
| Användarnamn i sociala medier | Avlägsnas |
| Placering i namngiven tävling | Avlägsnas |
| Exakt ålder | Avlägsnas när tre kategorier sammanfaller — se nedan |
| Förälders yrke eller arbetsplats | Avlägsnas när tre kategorier sammanfaller — se nedan |
| Liten ort, namngiven klubb, lokal händelse | Visas för läraren, avlägsnas inte |

Gränsen är dragen efter var kostnaden ligger. En missad uppgift kan lämna datorn. En felaktigt borttagen mening gör bedömningen sämre. För de fyra första väger det första tyngre; för de svagare gör det inte det.

De svagare uppgifterna **blockerar alltså ingenting**. En spärr som slår på ”jag är arton år” och ”min mamma är undersköterska” skulle stoppa varje personligt berättande — en egen genre i svenska som andraspråk — och läraren skulle sluta använda programmet. I stället listas de på dataskyddssidan i programmet, under rubriken *Detta står kvar – med avsikt*, tillsammans med det stycke de hittades i. Läraren kan då avgöra själv om just den här texten säger för mycket, och redigera den för hand innan den skickas.

Avvägandet är alltså synligt i stället för tyst. Ett tyst avvägande är inget som en lärare eller ett dataskyddsombud kan granska.

src/services/kvasiidentifierare.js

```
// ORDET "enda" RÄCKER INTE. Svenskan använder det överallt: "ett enda F
// bland kriterierna", "endast om". Första versionen av regeln träffade åtta
// ställen i vår egen bedömningsmatris och spärrade varje utgående anrop -
// alltså hela programmet, inte bara de farliga texterna.
const ENTYDIG_RE = new RegExp(
  String.raw`(?<![\p{L}])(?:${ENTYDIG_MARKOR})(?![\p{L}])`
  + String.raw`[^.!?\n]{0,80}?(?<![\p{L}])(?:i|på|inom)\s+(?:${ENTYDIG_PLATS})(?![\p{L}])`
  + String.raw`[^.!?\n]{0,60}?(?=[.!?\n]|$)`, 'giu');
```

**När flera harmlösa uppgifter sammanfaller**

Entydighetsmarkören fångar den mening som säger ”den enda”. Den fångar inte det fall där ingen enskild mening säger någonting alls:

| MENING | HUR MÅNGA DEN BESKRIVER |
|---|---|
| ”Jag är 17 år.” | Tusentals elever |
| ”Jag bor i byn.” | Tusentals elever |
| ”Jag spelar målvakt i Åstorps IF.” | Hundratals |
| ”Min mamma är rektor.” | Några |

Tillsammans är det en person. Ingen enskild mening innehåller en personuppgift, och därför fångar ingen enskild regel den. Det är just det fall som gör pseudonymisering av fri text svårare än att stryka namn, och det är precis den situation artikel 4.1 beskriver: uppgifter som *tillsammans* kan hänföras till en fysisk person.

En särskild motor räknar därför kategorier på **dokumentnivå** i stället för meningsnivå. Två avvägningar styr vad den gör:

| AVVÄGNING | INNEBÖRD |
|---|---|
| **Tröskeln** | Under tre samtidiga kategorier görs ingenting alls. En ensam uppgift pekar inte ut någon, och att maskera den vore ren förlust för bedömningen. |
| **Urvalet** | Bara biuppgifter maskeras — exakt ålder och förälderns yrke. De bär sällan textens innehåll; ingen argumentation vilar på att skribenten är sjutton. Orten, klubben och den lokala händelsen står kvar, eftersom en text mycket väl kan handla om dem. |

src/services/kombinationsmotor.js

```
const TROSKEL = 3;

// De kategorier som får maskeras när tröskeln nås. Urvalet är avsiktligt
// smalt. Ålder och förälderns yrke är biuppgifter i nästan varje skoltext;
// ort, klubb och lokal händelse kan vara själva ämnet.
const MASKERAS = new Set(['EXAKT ÅLDER', 'FÖRÄLDERS ARBETE']);

// Platshållaren säger inte VAD som togs bort. "[ÅLDER]" vore att i
// platshållaren återge just den uppgift som skulle skyddas - läsaren vet
// då att det stod en ålder där, och kombinationen är delvis återställd.
const PLATSHALLARE = '[UPPGIFT BORTTAGEN]';
```

Kvar blir en text som går att bedöma, men där kombinationen inte längre är komplett. Det är inte anonymisering — sådan finns inte för fri text — utan en mätbar försvagning av möjligheten att peka ut eleven. Tröskeln står som en egen namngiven konstant just för att den ska kunna ändras när verkliga elevtexter visat var den hör hemma.

**Uppgifter som blir personuppgifter genom att höra till en person**

Den andra motorn löser ett näraliggande men skilt problem. En utredande text som refererar en intervju kan innehålla meningar som var för sig är rena sakuppgifter:

> *”Deltagaren, 17 år, pendlar med 06.47-bussen från en mindre ort norr om Uppsala.”*

Ett klockslag är inte en personuppgift. En kompassriktning är inte en personuppgift. Men när de står i en mening som handlar om en bestämd människa blir de det, och tillsammans pekar de ut vilken buss en namngiven ungdom stiger på varje morgon.

Motorn maskerar därför inte klockslag och avstånd i allmänhet. Den letar först upp de personer grundskyddet redan funnit, och binder sedan uppgifter till dem inom samma meningskedja. Kedjan bryts vid varje ny referens, så att uppgifter inte vandrar över till fel person. Följande knyts på det sättet: ålder, födelsedag, datum, bostad, våningsplan, avstånd från hemmet, arbetsplats och arbetstider, anhörigs arbetsplats, förening och ungdomslag, resa och avgångstid, ort och väderstreck, samt initialer.

Motprovet är lika viktigt som regeln. Samma ord utan en person i närheten står kvar: *”Skolan ligger ungefär 700 meter från stationen”*, *”Bussen går 06.47 varje morgon”*, *”Sverige har fler dialekter norr om Dalälven”*. Utan den avgränsningen hade varje utredande text om kollektivtrafik eller geografi plockats isär. Både reglerna och motproven mäts i `test/run-personkoppling.js`, 55 kontroller.

**Uppgifter enligt artikel 9 som uttrycks utan kategoriordet**

Samma svaghet fanns i artikel 9-skyddet. Regeln i 5.2 kräver en bärare och ett verb — ”jag är muslim”, ”jag har adhd”. Men ingen behöver skriva så:

| FORMULERING | VAD SOM AVSLÖJAS |
|---|---|
| ”Efter min cancerdiagnos halkade jag efter.” | Hälsa |
| ”Jag tar Concerta varje morgon.” | Hälsa, via preparatet |
| ”Min psykolog säger att jag ska skriva dagbok.” | Hälsa, via vårdkontakten |
| ”På fredagar går jag till moskén.” | Religiös uppfattning, via handlingen |
| ”Jag går på möten med ungdomsförbundet.” | Politisk åsikt |
| ”Mamma har varit sjukskriven för depression.” | Hälsa, om en annan person |

Tre nya vägar in täcker dem: ägda substantiv (”min diagnos”, ”efter operationen”), läkemedel vid namn, och praktiker — men praktikerna endast i meningar som handlar om skribenten själv. Utan det sista villkoret skulle ”Moskén i Fittja är en av landets största” strykas ur en text som inte handlar om skribenten alls.

**Personuppgifter om andra än eleven**

Namnet på en annan människa är en personuppgift om den människan. Listan över relationer omfattade tidigare bara de närmaste banden, och därför stod ”min lillasyster Elvira”, ”min bonuspappa Marcus” och ”min tränare Kalle” kvar med namn. Den täcker nu syskon i sammansatta former, styv-, bonus- och fosterfamilj, partner, grannar, tränare, chefer och kontaktpersoner — alltså just de relationer en elev namnger när texten blir personlig.

Reglerna krävde länge att bäraren stod i **första person** — ”min syster heter Rosa”. En utredande text skriver inte så. Den skriver ”hans mamma Helena Bergqvist arbetar som tandläkare”, och där stod namnet kvar. Hela genren utredande text om andra människor saknade alltså skydd. Släktnamn i tredje person täcks nu på samma sätt som i första.

**Ett halvt maskerat namn är sämre än inget**

Vid ett prov den 8 september 2026 lämnade programmet ifrån sig `Helena [NAMN-11]` och `Daniel [NAMN-8]`: efternamnet ersatt, förnamnet kvar. Orsaken var att ersättningen skedde ord för ord, och att efternamnen råkade vara igenkända från annat håll i texten medan förnamnen inte var det.

Detta är värre än att inte maskera alls. Texten *ser* skyddad ut, både för läraren och för den som granskar, medan förnamnet ligger kvar tillsammans med allt annat i meningen. Ett namn som känns igen ersätts därför numera som **ett enda span**, i en operation, aldrig token för token. Kravet är prövat med ett eget mönster som letar efter just en kvarstående versal följd av en platshållare.

**Vad detta inte löser**

Ett regelbaserat system kan inte avgöra alla former av indirekt identifierbarhet. En tillräckligt ovanlig livshändelse, beskriven utan något av de mönster som beskrivs ovan, kan fortfarande göra en elev igenkännbar för den som redan känner henne eller honom. Lärio gör inte anspråk på motsatsen. Kontrollerna beskrivna här minskar risken och gör den mätbar; de tar inte bort den.

## 5.4 Fail-closed-spärr i nätverkslagret

Avidentifiering som utförs av anroparen kan glömmas när en ny kodväg tillkommer. Lärio placerar därför kontrollen **i själva nätverksanropet**, i den funktion som konstruerar den utgående begäran. All text som ska lämna processen passerar denna rad.

src/services/geminiClient.js

```
async function generateFromParts(parts, generationConfig = {}) {
  // SPÄRREN. Den sitter här, vid själva anropet, och inte hos den som
  // anropar. Skälet är att en ny kodväg då inte kan råka glömma den.
  //
  // Kastar den görs inget anrop alls. Fail-closed: hellre ett uteblivet
  // svar än en personuppgift som lämnar datorn.
  for (const del of parts) {
    if (del && typeof del.text === 'string') {
      assertClean(del.text, 'Gemini');
    }
  }
```

Funktionen `assertClean` genomsöker texten efter kvarvarande strukturerade identifikatorer. Vid träff kastas ett fel och **inget anrop utförs**. Beteendet är avsiktligt fail-closed: ett uteblivet svar är en olägenhet, en läckt personuppgift är en incident.

Spärren ovan granskar de textdelar anroparen lämnar in. Den svarar däremot inte på frågan om det *finns* textdelar som aldrig lämnades in — ett nytt fält i den utgående begäran, tillagt av en senare ändring, hade passerat oupptäckt. Sedan version 2.9.0 granskas därför själva den färdiga begäran, strukturellt: kroppen tolkas tillbaka till sina beståndsdelar, och varje sträng i den måste vara en sträng som redan har granskats. Är den inte det görs inget anrop.

src/services/geminiClient.js

```
// Granskningen sker STRUKTURELLT, inte på den serialiserade texten.
//
// Ett tidigare försök läste kroppen som en enda sträng. Det fungerade
// inte: JSON-kodningen gör om radbrytningar till tecknen  och n, och
// meningsgränserna försvinner. Regler som bygger på meningar blev då
// meningslösa, och spärren larmade på rätt texter.
//
// Kroppen tolkas därför tillbaka till sina beståndsdelar, och varje
// sträng jämförs mot mängden redan granskade strängar. Bilddata undantas
// - den är inte text och kan inte granskas som text.
```

Skillnaden mot `assertClean` är att den här kontrollen inte letar efter personuppgifter alls. Den letar efter *ogranskad text*, och behöver därför inte veta hur en personuppgift ser ut för att stoppa den. Fem kontroller i `test/run-lackage.js` mäter detta, bland annat genom att lägga till ett nytt fält på toppnivå och kontrollera att anropet uteblir.

## 5.5 Filnamn och metadata

Inlämnade filer bär ofta elevens namn eller personnummer i filnamnet. Filnamnet visas i lärarens gränssnitt, eftersom läraren måste kunna avgöra vilken återkoppling som hör till vilken elev, men det förs aldrig vidare till bedömningen.

`src/routes/feedback.js`

```
// FILNAMNET VISAS PÅ SKÄRMEN, MEN LÄMNAR ALDRIG DATORN.
//
// Skälet till att det gick att ändra: filnamnet går aldrig in i
// prompten. Bedömningen får bara den pseudonymiserade texten,
// aldrig namnet på filen.
```

## 5.6 Dolda personuppgifter i filen: metadata

Ett dokument bär personuppgifter på ställen där ingen text står. De påverkas inte av avidentifieringen i avsnitt 5.1, eftersom de inte ingår i texten, och de följer med genom nästan varje bearbetning.

| VAR | VAD |
|---|---|
| Word, dokumentegenskaper `docProps/core.xml` | Vem som skapade filen och vem som senast sparade den, hämtat från Windows-kontot i elevens egen installation. Därtill titel, ämne, beskrivning, nyckelord och kategori, där elever regelbundet skriver sitt namn. |
| Word, programegenskaper `docProps/app.xml` | Företag, vilket i skolsammanhang är skolans namn, samt ansvarig person. |
| Word, egna egenskaper `docProps/custom.xml` | Fält som skolan eller mallen lagt in. Innehållet är okänt på förhand. |
| Word, spårade ändringar och kommentarer | Varje markering bär författarens namn, initialer och i förekommande fall det Microsoft-konto ändringen gjordes från. |
| Word, mallens sökväg `word/settings.xml` | Sökvägen innehåller nästan alltid ett användarnamn: `C:\Users\fornamn.efternamn\…` |
| Word, förhandsbild `docProps/thumbnail.jpeg` | En bild av dokumentets första sida, alltså vanligen namnraden högst upp. |
| PDF, dokumentegenskaper | Författare, titel, ämne, nyckelord, samt XMP-metadata, som är en andra uppsättning egenskaper lagrad separat i filen. |

**Två skilda frågor, med två skilda svar**

Metadata reser åt två håll, och de behandlas olika. Skillnaden är avgörande och besvaras var för sig nedan.

| FRÅGAN | SVARET |
|---|---|
| **Följer metadata med i anropet?** Alltså till Google. | **Nej — den kan inte göra det.** Dokumentet skickas aldrig. Programmet läser ut texten ur filen och skickar enbart den, pseudonymiserad. Metadata ingår inte i texten och har därför ingen väg ut. |
| **Följer metadata med i filen läraren får tillbaka?** | **Nej — fälten töms.** Här krävs en aktiv rensning, eftersom filen är densamma som eleven lämnade in. |

Det första svaret är **strukturellt**, inte en städning: det finns ingen kodväg som bifogar originalfilen till ett anrop. Ett strukturellt skydd är samtidigt det lättaste att förlora av misstag — en framtida kodväg som skickar med filen vore enkel att skriva och omöjlig att upptäcka utan ett prov. Därför finns `test/run-metadata-stannar.js` : det bygger ett riktigt Word-dokument med elevens namn i dc:creator, i cp:lastModifiedBy, i dokumentets titel, i företagsfältet, i kommentarernas författarattribut och i mallens sökväg, kör det genom den skarpa vägen och genomsöker varje byte som lämnade processen.

Provet kontrollerar först att uppgifterna *verkligen står i filen* — annars vore det grönt av att dokumentet var tomt — och att elevens text *faktiskt skickades*, så att det inte kan bli grönt genom att ingenting sändes alls.

Rensningen av utfilen sker i en gemensam modul som både Word- och PDF-behandlingen anropar, av samma skäl som spärren i avsnitt 5.4 ligger i nätverkslagret: två separata städningar för två format glider isär, och den ena får en ny regel medan den andra inte gör det.

`src/services/metadataRensning.js`

```
// Namnrymden skrivs ut framför attributet, och den är INTE alltid "w".
// Kommentarer i document.xml använder w:author, men personlistan i
// people.xml använder w15:author, och Microsoft lägger till nya prefix
// med varje Word-version.
//
// En lista med prefix hade missat nästa. Därför matchas vilket prefix som
// helst: allt som slutar på :author, :initials eller :userId är ett namn
// eller en inloggning, oavsett vem som hittade på namnrymden.
.replace(/\s[A-Za-z0-9]+:author="[^"]*"/g, (m) => `${m.split('=')[0]}=""`)
```

**Två krav som står mot varandra**

Rensningen får inte skada dokumentet. Elevens typsnitt, styckeindelning, kursiveringar och rubriker är sådant bedömningen delvis vilar på, och ett dokument som kommer tillbaka omformaterat är ett förstört underlag.

Därför töms fälten i stället för att delarna tas bort. En `.docx` är en zip där varje del är utpekad; försvinner en del som något pekar på öppnar Word filen som skadad. Ett tomt fält skyddar lika väl som en borttagen fil.

Provet `test/run-metadata.js` mäter båda kraven samtidigt: att namnet är borta ur varje del av filen, och att dokumentets innehåll i övrigt är *byte för byte oförändrat* bortsett från författarattributen.

Metadata har aldrig ingått i det som överförs till mottagaren. Textutvinningen läser dokumentets brödtext; dokumentegenskaper ingår inte i den. Rensningen ovan skyddar därför inte överföringen utan *filen läraren får tillbaka och kan komma att spara eller vidarebefordra*.

## 5.7 Anropets konstruktion

Autentiseringsnyckeln överförs i avsett autentiseringshuvud och inte som frågeparameter i adressen. Frågesträngar loggas rutinmässigt av mellanliggande proxyservrar, sparas i webbläsarhistorik och följer med i felrapporter och hänvisningsadresser. Att lägga nyckeln i ett huvud tar inte bort varje sådan risk — huvuden kan loggas av proxyservrar och felsökningsverktyg — men det undanröjer den vanligaste vägen ut, den där uppgiften hamnar i en adress som sparas vidare.

`src/services/geminiClient.js`

```
// Nyckeln skickas i ett huvud, inte som frågetecken-parameter i URL:en.
// Query-strängar hamnar i proxyloggar, i webbläsarhistorik och i
// felrapporter hos mellanliggande servrar; ett huvud gör det inte.
headers: {
  'Content-Type': 'application/json',
  'x-goog-api-key': apiKey,
},
```

## 5.8 Skydd av autentiseringsuppgifter

Lärarens API-nyckel krypteras med Windows inbyggda nyckelhantering (DPAPI) via Electrons `safeStorage`. Krypteringen är bunden till användarkontot på den specifika datorn. En kopierad datamapp — på ett USB-minne, i en

molnmapp, i en säkerhetskopia — innehåller därmed en nyckel som inte går att använda någon annanstans.

src/hemlighet.js

```
// Electrons safeStorage använder Windows DPAPI. Krypteringen är låst
// till DITT Windows-konto på DEN HÄR datorn: filen blir oanvändbar
// överallt annars.
//
// VAD DET INTE SKYDDAR MOT, sagt rakt ut: skadlig kod som körs som du
// får nyckeln uppläst automatiskt, eftersom det är precis vad DPAPI är
// byggt för.
```

## 5.9 Inga tredjepartsresurser i gränssnittet

Gränssnittet hämtar ingenting från internet. Typsnitt, format och skript ligger i programfilen. Detta är av betydelse: inbäddade webbtypsnitt överför användarens IP-adress till leverantören vid varje sidvisning, vilket underkänts i tysk domstol (Landgericht München I, 20 januari 2022, mål 3 O 17493/20).

Förbudet upprätthålls i två lager. Ett automatiskt prov läser samtliga vyer och faller om en extern resurs tillkommer. Därutöver sätter programmet en innehållssäkerhetspolicy som stoppar samma sak i webbläsarmotorn:

server.js

```
res.setHeader('Content-Security-Policy', [
  "default-src 'self'",
  "script-src 'self' 'unsafe-inline'",
  "style-src 'self' 'unsafe-inline'",
  "font-src 'self'",
  "img-src 'self' data: blob:",
  "connect-src 'self'",
  "object-src 'none'",
  "frame-ancestors 'none'",
  "base-uri 'self'",
  "form-action 'self'",
].join('; '));
```

## 5.10 Loggning utan innehåll

Programmet för en logg över utgående anrop. Loggen registrerar tidpunkt, mottagare, ändamål, antal tecken som skickades och hur många personuppgifter som avlägsnades. **Textinnehåll registreras aldrig.**

src/egressLog.js

```
// VAD SOM SPARAS: tidpunkt, mottagare, ändamål, hur många tecken som
// skickades och hur många personuppgifter som togs bort - aldrig
// innehållet självt.
function notera({ mottagare, andamal, tecken, borttaget, bilder, modell })
```

Loggen kan visas i programmet. Den utgör underlag för huvudmannens registerföring enligt artikel 30.

### 5.11 Programintegritet

Den distribuerade programfilen är signerad med kontrollsummor per fil (*ASAR integrity*) och försedd med Electron Fuses som stänger av felsökningsingångar. Ett manipulerat program vägrar starta.

`package.json`

```
"electronFuses": {
  "enableEmbeddedAsarIntegrityValidation": true,
  "onlyLoadAppFromAsar": true,
  "enableNodeCliInspectArguments": false,
  "enableNodeOptionsEnvironmentVariable": false,
  "runAsNode": false
}
```

---

## 6 Lagring och gallring

| VAD | VAR | HUR LÄNGE |
|---|---|---|
| Rättade dokument | `Lärio-data/db.json` | Gallras automatiskt efter det antal dagar läraren angett. Förval 30 dagar. |
| Bedömningsomgångar | Endast arbetsminnet | Försvinner när programmet stängs. Lagring på disk är **avstängd som förval** och kräver ett aktivt val. |
| Arkiverade rättningar | `Lärio-data/arkiv.json` | Tills läraren själv raderar. Läggs dit endast genom ett uttryckligt val. |
| Uppgifter och källor | `Lärio-data/assignments.json` | Tills läraren själv raderar. Innehåller lärarens material, inte elevens. |
| Utgående logg | `Lärio-data/utgaende-logg.json` | Kan rensas av läraren. Innehåller inget textinnehåll. |

**Radering är slutgiltig**

När läraren raderar en arkivmapp avlägsnas innehållet helt, inklusive samtliga säkerhetskopior. Åtgärden kräver att ordet `RADERA` skrivs för hand, och kontrollen utförs i programmets lokala backend-process och inte enbart i gränssnittet — en knapp som låses upp i webbläsarlagret utgör ingen spärr. (Lärio har ingen serverkomponent; backend-processen kör på lärarens egen dator.)

Motivet är rättsligt: en radering som lämnar kvar innehållet i en säkerhetskopia uppfyller inte artikel 17. En begäran om radering från en registrerad ska kunna verkställas fullständigt.

## 7 Mottagare och tredjelandsöverföring

Den avidentifierade texten överförs till Google för språkanalys. Detta är programmets enda mottagare. Överföringen sker till `generativelanguage.googleapis.com`.

Huvudmannen bör beakta att Google är etablerat i tredjeland och att överföringen därmed omfattas av kapitel V. Det bör samtidigt vägas in att det överförda materialet är avidentifierat enligt avsnitt 5.1–5.2, och att materialet därigenom har ett väsentligt lägre skyddsvärde än en identifierbar elevtext.

Frågan om huruvida avidentifierat material över huvud taget utgör personuppgifter i den mening som avses i artikel 4.1 är beroende av omständigheterna i det enskilda fallet och avgörs av den personuppgiftsansvarige.

## 8 Inbyggt dataskydd (artikel 25)

1. **Dataminimering.** Endast den avidentifierade texten och lärarens egen uppgiftsinstruktion överförs. Filnamn, metadata och lärarens egna anteckningar följer inte med.
2. **Dataskydd som förval.** Lagring av bedömningar är avstängd tills läraren aktivt slår på den. Uppdateringskontrollen kan stängas av. Ingenting samlas in om användaren.
3. **Lagringsminimering.** Rättade dokument gallras automatiskt. Bedömningsomgångar hålls som förval endast i arbetsminnet.
4. **Ändamålsbegränsning.** Programmet har en enda utgående mottagare och ett enda ändamål för överföringen: språklig analys av den avidentifierade texten.

## 9 Verifierbarhet

Påståendena ovan är inte enbart beskrivningar. De upprätthålls av automatiska prov som körs vid varje ändring i koden och som mäter *vad som faktiskt lämnar processen*, inte vad koden uppges göra.

| PROV | VAD DET MÄTER |
|---|---|
| `test/run-lackage.js` | Startar hela programmet, skickar trettio elevtexter genom den väg en lärare använder, och genomsöker varje byte som lämnade processen. Texterna innehåller namn, personnummer, e-post, telefon, adress, skola, klass, samtliga kategorier enligt artikel 9, samma kategorier uttryckta utan kategoriordet, personuppgifter om andra än eleven, och kvasi-identifierare. Filerna är namngivna efter eleven. Trettio mönster får inte förekomma i utgående data; motprov säkerställer att sakinnehållet *faktiskt skickades*, så att provet inte kan bli grönt genom att ingenting sändes. Provet granskar dessutom den färdiga begäran strukturellt och kräver att varje sträng i kroppen är en redan granskad sträng. Sextio kontroller. |
| `test/run-metadata-stannar.js` | Att dold metadata inte lämnar datorn vid anropet: ett riktigt Word-dokument med elevens namn i sex metadatafält körs genom den skarpa vägen, och varje byte som lämnade processen genomsöks. Två motprov säkerställer att uppgifterna fanns i filen och att texten faktiskt skickades. |
| `test/run-filnamn-stannar.js` | Att filnamnet aldrig når prompten eller nätverket. Nätverkslagret byts ut och den utgående begäran granskas. |
| `test/run-arkivet.js` | Att lärarens egna namn på klasser och elever aldrig lämnar datorn, och att radering avlägsnar innehållet från hela datamappen, inklusive säkerhetskopior. |
| `test/run-pii.js` , `test/run-pii-bank.js` | Avidentifieringens träffsäkerhet mot en bank av konstruerade fall, mätt i **två** tal i stället för godkänt eller underkänt: *täckning* — hur många planterade uppgifter som togs bort — och *precision* — hur många rena texter som lämnades helt orörda. Vid senaste körningen 67 av 67 respektive 18 av 18. |
| `test/run-indirekt-identifiering.js` | Indirekt identifierbarhet: att artikel 9-uppgifter uttryckta utan kategoriordet, personuppgifter om andra än eleven och kvasi-identifierare avlägsnas — att ämnestexter står kvar orörda — att den andra spärren fångar det den första missar — och att skyddet inte stoppar programmets egna prompter. |
| `test/run-artikel-9.js` | Att var och en av de nio kategorierna i artikel 9 finns kvar i koden och faktiskt avlägsnas — och, med lika många kontroller, att samma ord *står kvar* när de används som ämne, så att en text om adhd, islam eller hbtq fortfarande går att bedöma. |
| `test/run-angrepp.js` | Att en uppgift som skrivits på ett annat sätt fångas ändå: fullbreddssiffror, homoglyfer, osynliga tecken, mjuka bindestreck, isärspärrad text, blandade alfabet. Provet kräver också att den avidentifierade texten fortfarande är elevens egen text tecken för tecken — en normalisering som skrev om texten vore ett fel, inte ett skydd. 121 kontroller. |
| `test/run-utokad.js` | Att grundskyddet är ett golv. En motor som kastar fel, tar för lång tid, returnerar skräp eller försöker stryka en av grundskyddets träffar får |

| | |
|---|---|
| | inte försämra resultatet; den kopplas bort ensam och de övriga står kvar. 34 kontroller. |
| `test/run-personkoppling.js` | Att uppgifter knutna till en person avlägsnas — och, med motprov för varje regel, att samma ord står kvar när ingen person finns i närheten. 55 kontroller. |
| `test/run-dataskydd.js` | Strukturella krav på koden: att inga kodvägar kringgår avidentifieringen. |
| `test/run-inget-utifran.js` | Att gränssnittet inte hämtar någon resurs från internet, och att innehållssäkerhetspolicyn är på plats. |

**Motprovets betydelse**

Ett prov som endast kontrollerar att personuppgifter saknas i utgående data kan bli grönt av fel skäl: om inget anrop görs saknas allt. Proven innehåller därför kontroller som kräver att sakinnehållet *ska* finnas i den utgående begäran. Ett grönt resultat betyder alltså att texten skickades, och att inget personligt följde med den.

## 10 Vad skyddet inte omfattar

En redogörelse som endast beskriver styrkor är av begränsat värde för en granskare. Följande begränsningar redovisas därför uttryckligen.

1. **Avidentifieringen är regelbaserad och inte fullständig.** Ett namn skrivet med gemener mitt i en mening, utan sammanhang som avslöjar att det är ett namn, kan undgå upptäckt. Programmet visar därför läraren vad som identifierats innan texten skickas.
2. **Bilder granskas inte.** En bild är pixlar; att läsa ut personuppgifter ur den skulle kräva just det nätverksanrop som ska skyddas. Bildfunktionen är därför begränsad till lärarens egen bedömningsmatris, och läraren varnas innan bilden skickas.
3. **Kryptering av nyckeln skyddar inte mot skadlig kod.** Windows DPAPI dekrypterar automatiskt för den inloggade användaren. Skyddet avser kopierade datamappar, inte en kompromitterad dator.
4. **Arkivet innehåller de namn läraren själv skriver.** Funktionen finns för att läraren ska kunna följa en elev över en termin. Dessa namn lagras lokalt och lämnar aldrig datorn, men de utgör personuppgifter och omfattas av huvudmannens ansvar.
5. **Tröskeln för kombinerade uppgifter är en avvägning, inte ett bevis.** Tre samtidiga kategorier är valt, inte uträknat. Två uppgifter beskriver stora grupper; vid tre börjar mängden krympa snabbt. Var gränsen faktiskt går beror på skolans storlek och ortens. Siffran står som en egen namngiven konstant i koden just för att den ska gå att ändra, och en huvudman som bedömer att den ligger fel bör säga till.
6. **Skyddet mäts mot konstruerade fall.** Provbanken innehåller texter skrivna för att pröva reglerna. Den kan därför inte utesluta att en verklig elevtext formulerar sig på ett sätt banken inte innehåller. Måtten — täckning och precision — är avsedda att göra risken jämförbar över tid, inte att påstå att den är noll. Nya missar som upptäcks förs in i banken innan de rättas, så att en rättning som slutar fungera märks.

7. **Utökade motorer kan vara frånkopplade.** På en dator med knappa resurser kan en utökad motor kopplas bort under körning. Texten skyddas då av grundskyddet ensamt, vilket är svagare i fråga om kombinerade och personknutna uppgifter. Läget redovisas för läraren och döljs inte.
8. **Programfilen är ännu inte kodsignerad.** Windows visar därför en varning vid första start. Detta påverkar inte behandlingen av personuppgifter, men redovisas för fullständighetens skull.

## 11 De registrerades rättigheter

Eftersom allt elevmaterial finns på lärarens dator och inte hos leverantören, kan huvudmannen verkställa samtliga rättigheter enligt kapitel III utan medverkan från tredje part.

| RÄTTIGHET | VERKSTÄLLS GENOM |
|---|---|
| Tillgång (art. 15) | Historiken och arkivet i programmet; nedladdning av allt material som fil. |
| Rättelse (art. 16) | Läraren redigerar eller ersätter bedömningen direkt. |
| Radering (art. 17) | Radering i programmet avlägsnar innehållet fullständigt, inklusive säkerhetskopior. |
| Dataportabilitet (art. 20) | Nedladdning som zip med läsbara textfiler. |
| Registerföring (art. 30) | Den utgående loggen visar tidpunkt, mottagare och ändamål för varje överföring. |

Detta dokument beskriver programvarans tekniska konstruktion. Det utgör inte juridisk rådgivning och ersätter inte den personuppgiftsansvariges egen bedömning.

Källkodsreferenserna avser den version som anges i programmets nedre högra hörn. Beskrivningen uppdateras när konstruktionen ändras.