Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Christian Burkert, Jonathan Balack, Hannes Federrath

PrivacyDates: A Framework for More Privacy-Preserving Timestamp Data Types

GI Sicherheit 2022, 7. April

Agenda

Goal:
Present a **Framework** that makes it easier to **avoid full-precision timestamps** in app data models.

- Why necessary

Agenda

Goal:
Present a **Framework** that makes it easier to **avoid full-precision timestamps** in app data models.

- Why necessary
- Design of our Framework

Agenda

Goal:
Present a **Framework** that makes it easier to **avoid full-precision timestamps** in app data models.

- Why necessary
- Design of our Framework
- Implementation of our Framework

Agenda

Goal:
Present a **Framework** that makes it easier to **avoid full-precision timestamps** in app data models.

- Why necessary
- Design of our Framework
- Implementation of our Framework
- Cost of application

Setting the scene: How would you design that app?

Example: Design of a **chat app**.

User Stories:

1. *Users* can **join a Channel** and become a Channel Member.

Setting the scene: How would you design that app?

Example: Design of a **chat app**.
User Stories:

1. *Users* can **join a Channel** and become a Channel Member.
2. *Channel Members* can **post messages** to a Channel.

Setting the scene: How would you design that app?

Example: Design of a **chat app**.
User Stories:

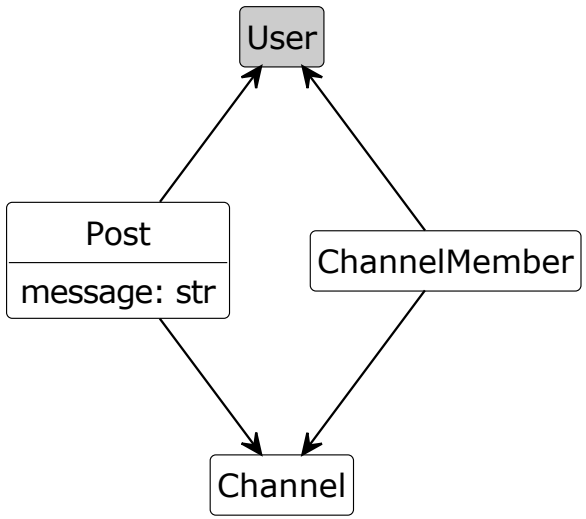
1. *Users* can **join a Channel** and become a Channel Member.
2. *Channel Members* can **post messages** to a Channel.
3. *Channel Members* can **read posts** in a Channel.

Setting the scene: How would you design that app?

Example: Design of a **chat app**.

User Stories:

1. *Users* can **join a Channel** and become a Channel Member.
2. *Channel Members* can **post messages** to a Channel.
3. *Channel Members* can **read posts** in a Channel.

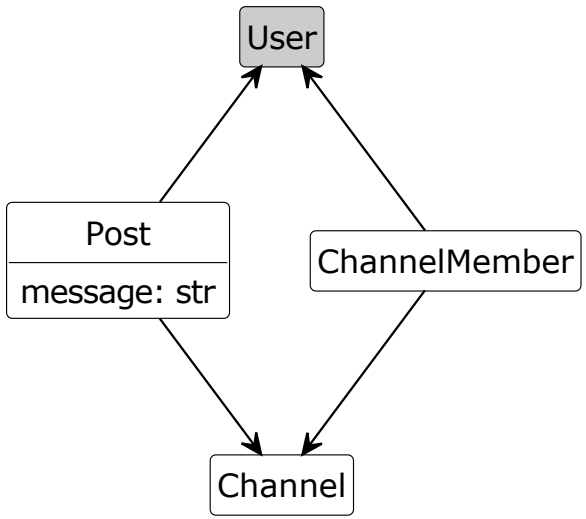


Setting the scene: How would you design that app?

Example: Design of a **chat app**.

User Stories:

1. *Users* can **join a Channel** and become a Channel Member.
2. *Channel Members* can **post messages** to a Channel.
3. *Channel Members* can **read posts** in a Channel.
4. *Channel Members* receive a visual indicator of what was **posted since** they **last viewed** a Channel

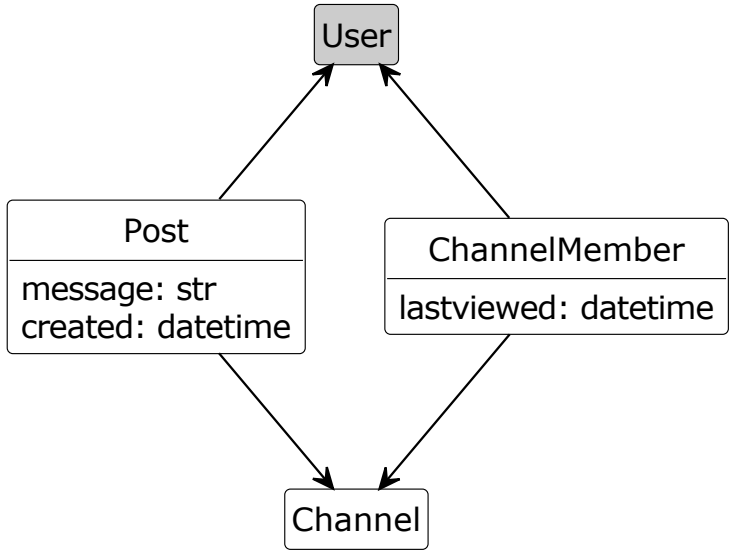


Setting the scene: How would you design that app?

Example: Design of a **chat app**.

User Stories:

1. *Users* can **join a Channel** and become a Channel Member.
2. *Channel Members* can **post messages** to a Channel.
3. *Channel Members* can **read posts** in a Channel.
4. *Channel Members* receive a visual indicator of what was **posted since** they **last viewed** a Channel

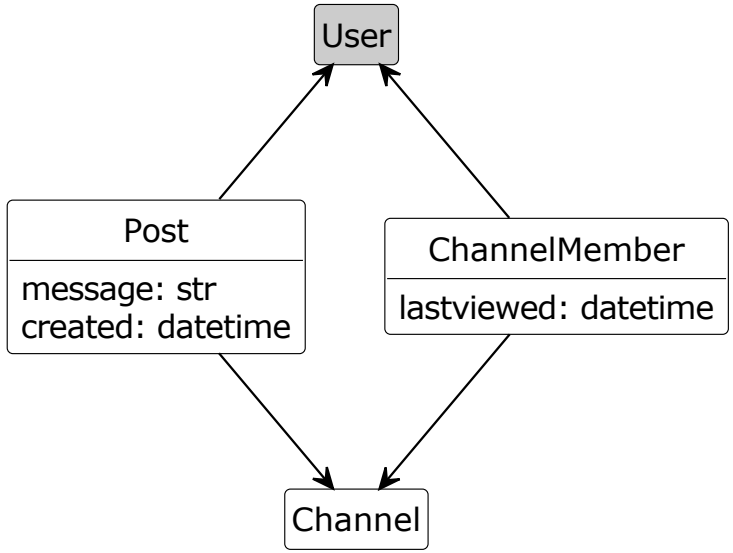


Setting the scene: How would you design that app?

Example: Design of a **chat app**.

User Stories:

1. *Users* can **join a Channel** and become a Channel Member.
2. *Channel Members* can **post messages** to a Channel.
3. *Channel Members* can **read posts** in a Channel.
4. *Channel Members* receive a visual indicator of what was **posted since** they **last viewed** a Channel
5. *Channel Members* can **see when** a post was **created**

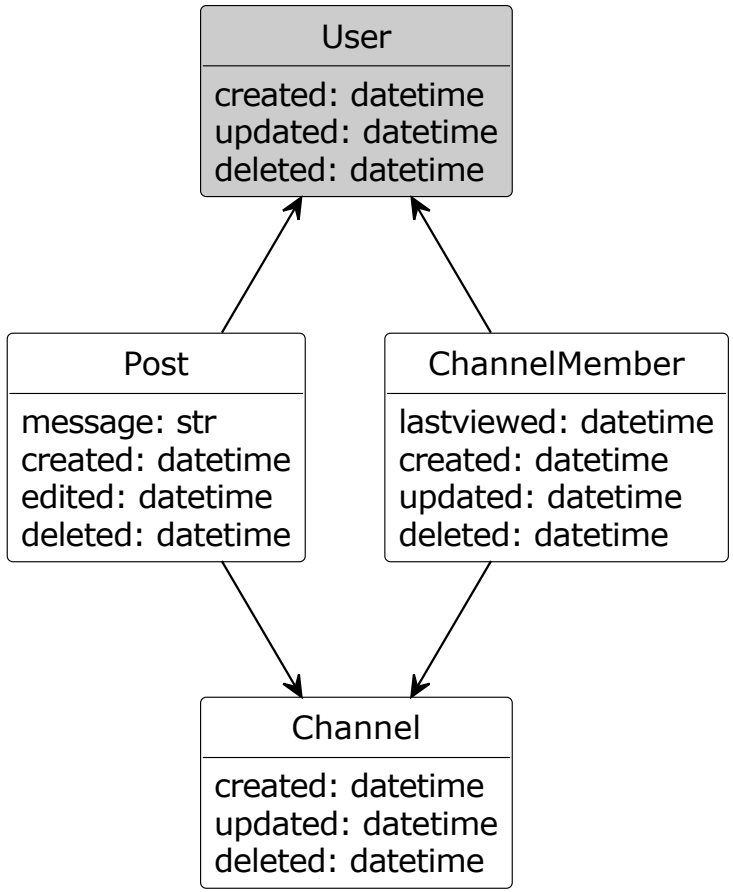


Setting the scene: How would you design that app?

Example: Design of a **chat app**.

User Stories:

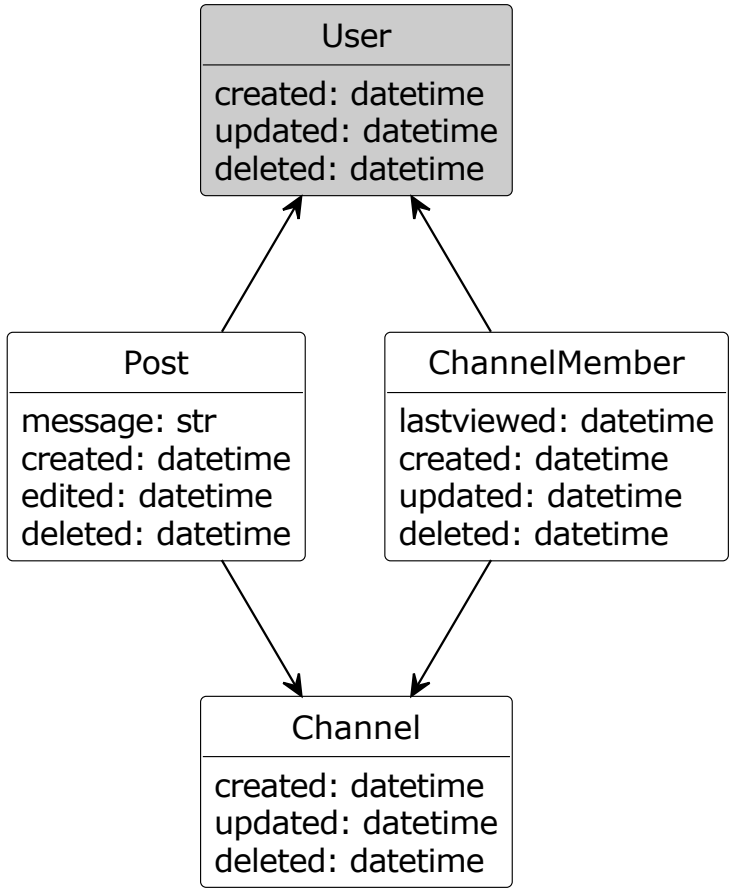
1. *Users* can **join a Channel** and become a Channel Member.
2. *Channel Members* can **post messages** to a Channel.
3. *Channel Members* can **read posts** in a Channel.
4. *Channel Members* receive a visual indicator of what was **posted since** they **last viewed** a Channel
5. *Channel Members* can **see when** a post was **created**



Prior Work: How are timestamps used in data models?

Case Studies

Source code analysis and UI inspection of Mattermost[a]



[a] Burkert and Federrath (2019): 'Towards Minimising Timestamp Usage In Application Software'

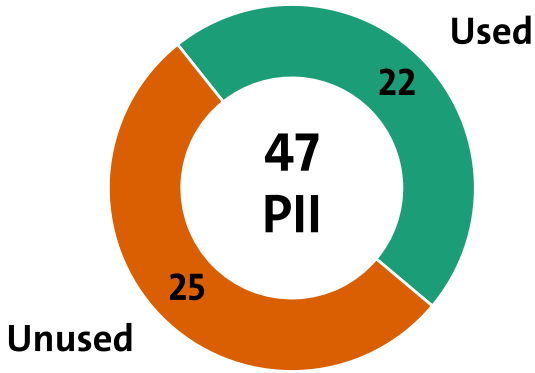
Prior Work: How are timestamps used in data models?

Case Studies

Source code analysis and UI inspection of Mattermost[a]

Findings

- Timestamps are excessively included in data models
  - >50% no programmatic use
  - most not visible on UI either



[a] Burkert and Federrath (2019): 'Towards Minimising Timestamp Usage In Application Software'

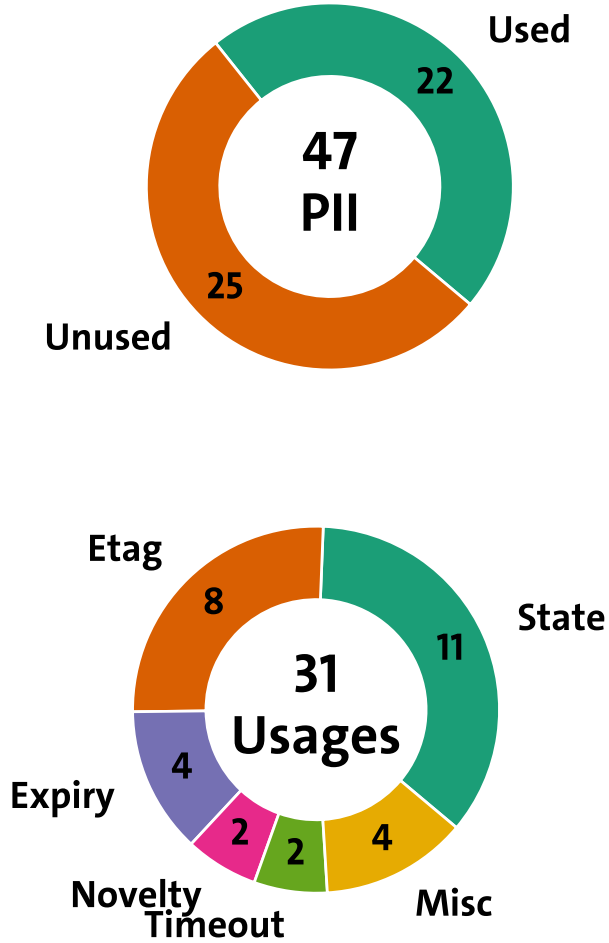
Prior Work: How are timestamps used in data models?

Case Studies

Source code analysis and UI inspection of Mattermost[a]

Findings

- Timestamps are excessively included in data models
  - >50% no programmatic use
  - most not visible on UI either
- Many timestamp use cases allow for less sensitive **alternatives**

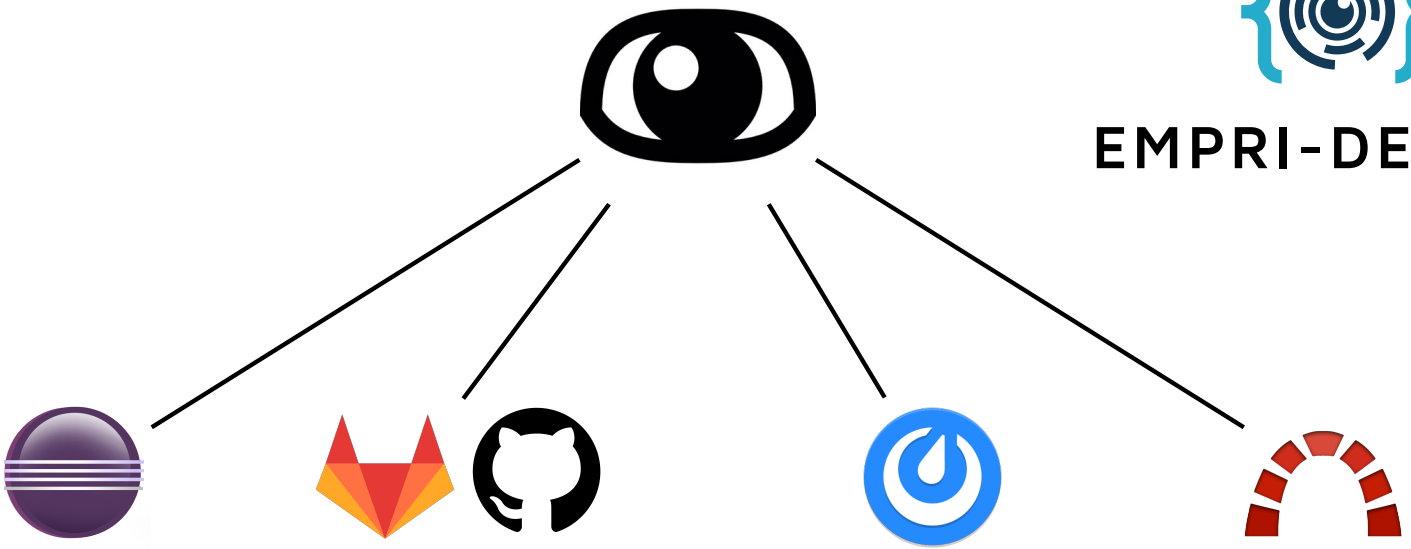


[a]Burkert and Federrath (2019): 'Towards Minimising Timestamp Usage In Application Software'

Motivation: Why necessary?

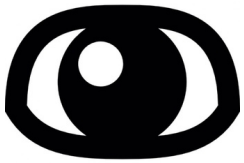
EMPRI-DEVOPS

Motivation: Why necessary?



EMPRI-DEVOPS

Attacker Model: Curious Employer / Supervisor / Colleague



Motivation Monitor and assess performance and well-behaviour of employees

Capabilities

- Unrestricted access to all persisted application data (databases, filesystems)
- Cannot extend the capabilities of applications

Design Process

Approach:

1. Searched literature for privacy patterns on data minimisation applicable for timestamps
2. Derived a design with timestamp alternatives
3. Validated Design in case study

Design Process

Approach:

1. Searched literature for privacy patterns on data minimisation applicable for timestamps
2. Derived a design with timestamp alternatives
3. Validated Design in case study

Identified Alternatives:

1. timestamp with statically reduced precision
   $\rightarrow$ **Rough Date**

Design Process

Approach:

1. Searched literature for privacy patterns on data minimisation applicable for timestamps
2. Derived a design with timestamp alternatives
3. Validated Design in case study

Identified Alternatives:

1. timestamp with statically reduced precision
   $\rightarrow$ **Rough Date**
2. timestamp with dynamically reduced precision over time
   $\rightarrow$ **Vanishing Date**

Design Process

Approach:

1. Searched literature for privacy patterns on data minimisation applicable for timestamps
2. Derived a design with timestamp alternatives
3. Validated Design in case study

Identified Alternatives:

1. timestamp with statically reduced precision
   $\rightarrow$ **Rough Date**
2. timestamp with dynamically reduced precision over time
   $\rightarrow$ **Vanishing Date**
3. pseudo-timestamp that is just a context-aware counter
   $\rightarrow$ **Ordering Date**

Design Validation

Can we accommodate all timestamp use with those 3 alternatives?

ToE Taiga[1] (project management app)

Method

1. Located all timestamp uses in code (model/API exposure) by source code and documentation inspection
2. Classified semantics and purpose

Model Field	Occurrences	Used in Models	Exposed via API
DateTimeField	170	48	41
DateField	15	3	3
TimeField	5	0	0

Table: Usage of date-related Django model fields in Taiga's back-end.

[1] https://www.taiga.io

Design Validation

Can we accommodate all timestamp use with those 3 alternatives?

ToE Taiga[1] (project management app)

Method
1. Located all timestamp uses in code (model/API exposure) by source code and documentation inspection
2. Classified semantics and purpose

Revision Combined type date + ordering

Model Field	Occurrences	Used in Models	Exposed via API
DateTimeField	170	48	41
DateField	15	3	3
TimeField	5	0	0

Table: Usage of date-related Django model fields in Taiga's back-end.

[1] https://www.taiga.io

Rough Date

A timestamp with static precision reduction.

API:

```
created_at = RoughDateField(hours=1)
```

Example:

original	2021-11-08 15:17
rough date (1h)	2021-11-08 15:00

Implementation:

- easy to adopt
- compatible with original `DateTimeField`

Vanishing Date

A timestamp with dynamically reduced precision over time.

API:

```
created_at = VanishingDateField(
  policy=make_policy([
    Precision(hours=1),
    Precision(days=1, after_hours=3),
    Precision(months=1, after_days=7),
  ])
)
```

Example:

Step	Current Time	Stored Date	Next Due Date
Creation/1st Red.	2021-11-08 15:17	2021-11-08 15:00	2021-11-08 18:00

Vanishing Date

A timestamp with dynamically reduced precision over time.

API:

```
created_at = VanishingDateField(
  policy=make_policy([
    Precision(hours=1),
    Precision(days=1, after_hours=3),
    Precision(months=1, after_days=7),
  ])
)
```

Example:

Step	Current Time	Stored Date	Next Due Date
Creation/1st Red.	2021-11-08 15:17	2021-11-08 15:00	2021-11-08 18:00
2nd Reduction	2021-11-08 18:01	2021-11-08 00:00	2021-11-15 00:00

Vanishing Date

A timestamp with dynamically reduced precision over time.

API:

```
created_at = VanishingDateField(
  policy=make_policy([
    Precision(hours=1),
    Precision(days=1, after_hours=3),
    Precision(months=1, after_days=7),
  ])
)
```

Example:

Step	Current Time	Stored Date	Next Due Date
Creation/1st Red.	2021-11-08 15:17	2021-11-08 15:00	2021-11-08 18:00
2nd Reduction	2021-11-08 18:01	2021-11-08 00:00	2021-11-15 00:00
3rd Reduction	2021-11-15 00:03	2021-11-01 00:00	-

Vanishing Date: Implementation

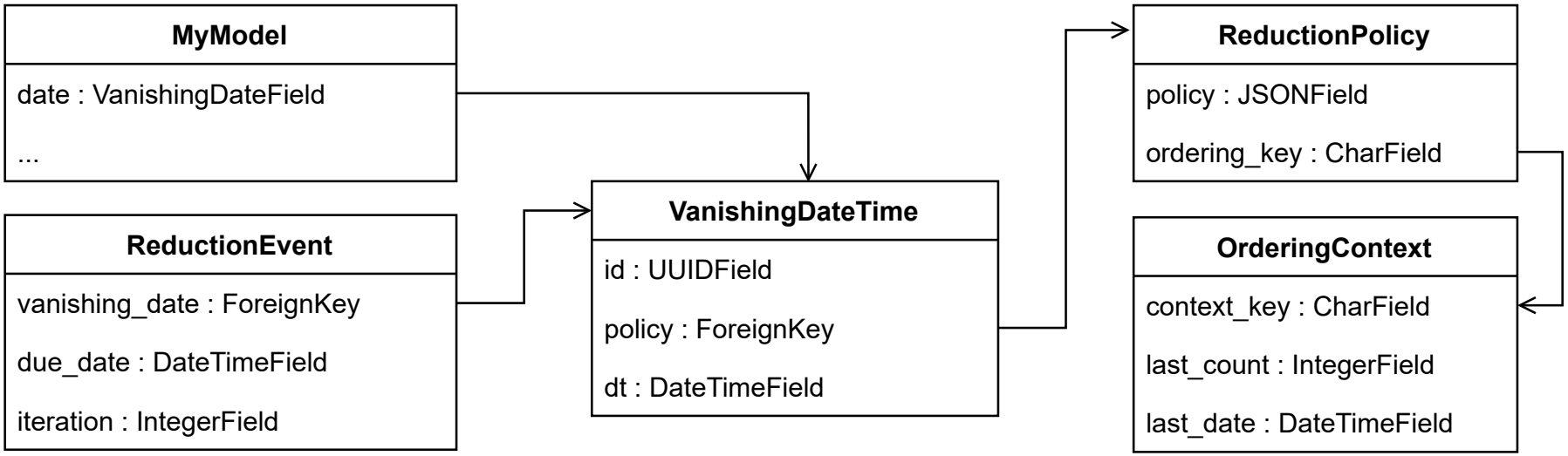


Figure: Architecture of vanishing date auxiliary models

- UUIDs instead of AutoField as primary keys to avoid insertion chronology leaks
  But: databases may still maintain insertion order
- Transparent policy reuse to preserve storage
- Models with vanishing date are identified via a mixin

Ordering Date

Not actually a timestamp but a sequence number with context isolation

API:

```
class MyModel(models.Model):
    created_at = OrderingDateField(hashed=True)

def do_something():
    my_instance = MyModel(
        created="my-context-key",
    )
    # ...
```

Implementation:

- Context keys identify the sequence number context
- Auxiliary model keeps track of context state
- Optional hashing for sensitive context keys

Combined Type: Vanishing and Ordering

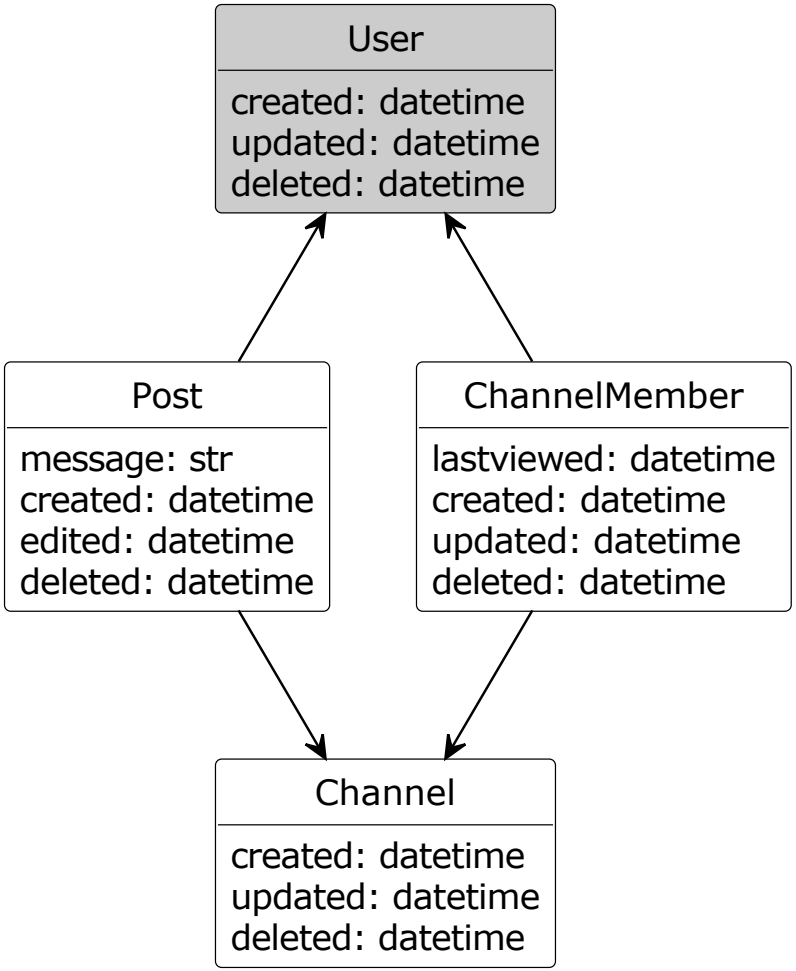
Idea: Use sub-second part to hold context counter

Original	**Vanishing** 1. Iteration [5 sec]	**Vanishing+Order** 1. Iteration [5 sec]	**Vanishing+Order** 2. Iteration [30 sec]
12:20:11:673320	12:20:10:000000	12:20:10:000000	12:20:00:000000
12:20:14:313406	12:20:10:000000	12:20:10:000001	12:20:00:000001
12:20:17:248323	12:20:15:000000	**12:20:15:000002**	**12:20:00:000002**
12:20:33:040852	12:20:30:000000	12:20:30:000000	12:20:30:000000
12:20:35:917632	12:20:35:000000	12:20:35:000001	12:20:30:000001

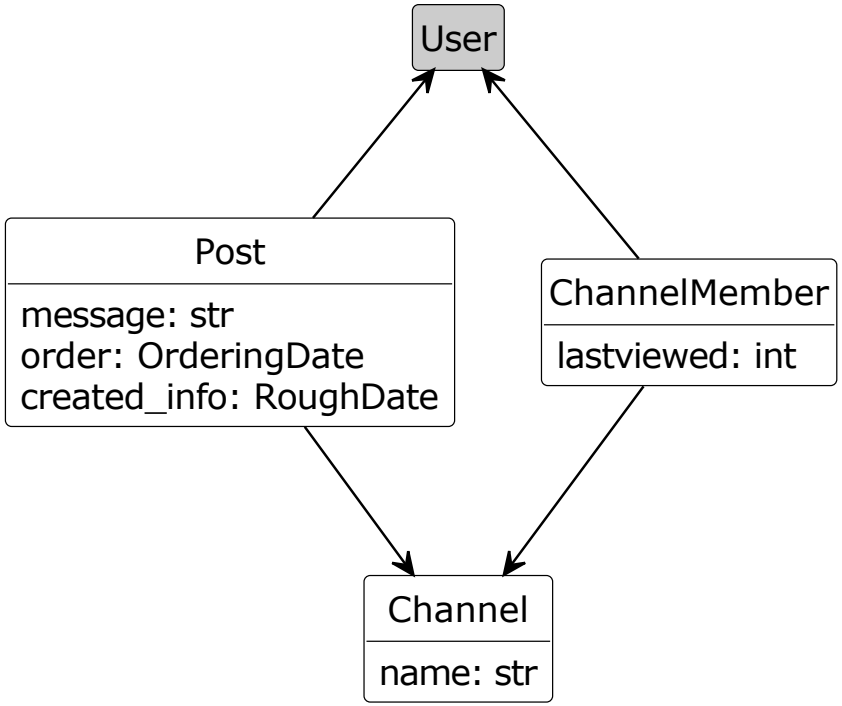
bold = counter reset

Application on Chat Example

Now



With Privacy Dates



- Channel name could be context key

Evaluation

- Practicality
  - Integrated in Taiga 6.0.7
  - Experimentally tested functioning
  - Implementation effort as expected

Evaluation

- Practicality
  - Integrated in Taiga 6.0.7
  - Experimentally tested functioning
  - Implementation effort as expected
- Storage Cost

Type	**Size [bytes]**	**Factor of** `DateTimeField`
`RoughDateField`	8	1.0
`VanishingDateField`	142	17.75
`OrderingDateField`	4	0.5

Limitations & Future Work

Limitations:

- Chronology Leaks trough DB insertion order possible

Future Work:

- Investigate insertion order leak for various DBMS

Limitations & Future Work

Limitations:

- Chronology Leaks trough DB insertion order possible
- Usability not evaluated

Future Work:

- Investigate insertion order leak for various DBMS
- Evaluate usability in developer study

Limitations & Future Work

Limitations:

- Chronology Leaks trough DB insertion order possible
- Usability not evaluated

Future Work:

- Investigate insertion order leak for various DBMS
- Evaluate usability in developer study
- Reduce storage cost for vanishing date

Summary: Findings & Contribution

Presented a **Framework** that makes it easier to **avoid full-precision timestamps** in app data models.

- Designed timestamp alternatives based on prior work
- Validates design on Taiga
- Implemented framework for Django (open source on GitHub)
- Evaluated framework practicality and cost

https://github.com/EMPRI-DEVOPS/django-privacydates

Summary: Findings & Contribution

Presented a **Framework** that makes it easier to **avoid full-precision timestamps** in app data models.

- Designed timestamp alternatives based on prior work
- Validates design on Taiga
- Implemented framework for Django (open source on GitHub)
- Evaluated framework practicality and cost
- Learnt: DBMS needs to play along

https://github.com/EMPRI-DEVOPS/django-privacydates

Contact

Christian Burkert

Tel. +49 40 42883-2406
Mail christian.burkert@uni-hamburg.de

I'd be happy to hear from you!

OpenPGP Fingerprint:

`9B97 CC4B 5FF4 7BA3 EF7B 1966 A5FB 6E0B 41AC CDFB`