

GRUPO 1

CIERRE SPRINT 4

User Stories Planificadas

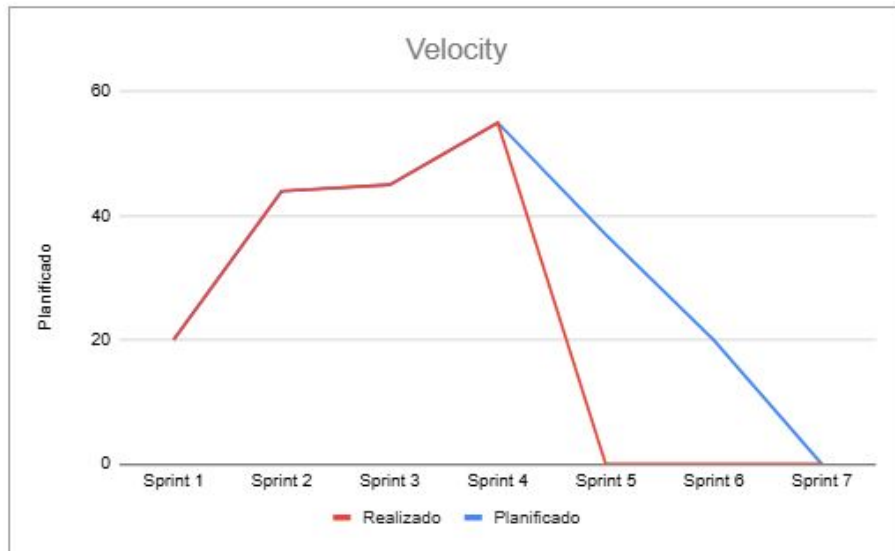
Historias de usuario

- US 4.9 Compartir itinerario (Premium)
- US 5.8 Me gusta en reseñas (Premium)
- US 5.9 Comentar una reseña
- US 5.10 Reportar una reseña
- US 6.1 Cargar gastos del viaje
- US 6.2 Editar / Eliminar un gasto
- US 6.3 Vista de gastos por categoría
- US 6.4 Compartir planilla de gastos (Premium)
- US 6.5 Invitar por mail a la planilla (Premium)
- US 6.6 Cálculo de saldos por integrante
- US 6.7 Exportar presupuesto

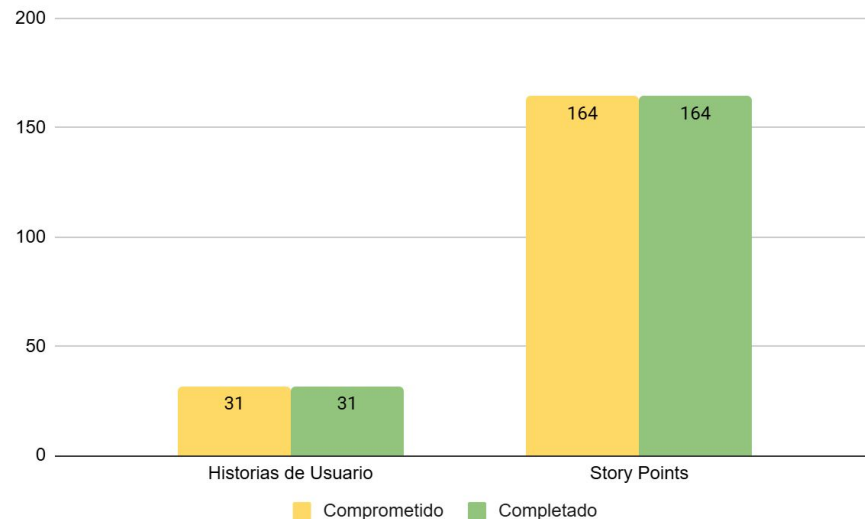
8 SP	Completado
5 SP	Completado
5 SP	Completado
5 SP	Completado
8 SP	Completado
3 SP	Completado
5 SP	Completado
5 SP	Completado
5 SP	Completado
3 SP	Completado
3 SP	Completado

Velocity Total: 55 SP

Métricas

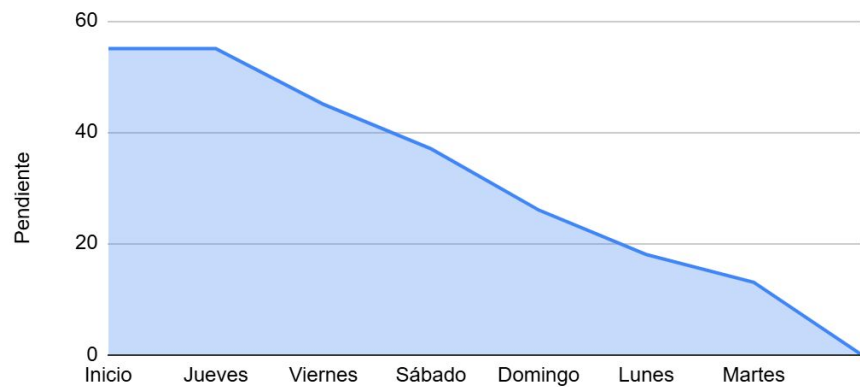


Velocity actual: 55 SP

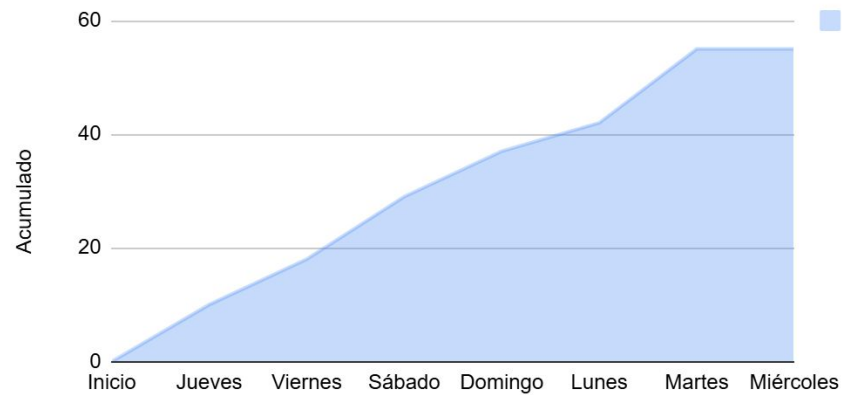


Métricas

Burn Down



Burn Up



Uso de IA

Uso de IA

- Herramienta utilizada: ChatGPT, Gemini y Copilot
- Versiones e iteraciones: Se le envió contexto con código ya hecho con el fin de generar código nuevo. Se utilizó también para resolución de errores.

Uso de IA - Ejemplos

Home

<> JSX

Suggestions

<> JSX

ItineraryRequest

<> JSX

model

<> PY

como nuevo feature a desarrollar, tengo que poder dar
me gusta a reseñas si soy usuario premium. en el
PublicationModal, en cada reseña tiene que verse un
corazon con la cantidad d elikes que tiene esa reseña

¡Excelente feature! Esta funcionalidad requiere cambios tanto en el backend (base de datos y API) como en el frontend (el modal).

Aquí tienes la implementación completa dividida por archivo:

1. `models.py` (Backend - Base de Datos)

Necesitamos una nueva tabla, `review_likes`, para rastrear qué usuario le dio "me gusta" a qué reseña.

Agrega esta nueva clase al final de tu archivo `models.py`, junto con una relación en la clase

`Review`:

Python

```
# ... (después de la clase Review) ...

# -----
# Review Likes (N-N)
# -----
class ReviewLike(Base):
    __tablename__ = "review_likes"

    id = Column(Integer, primary_key=True, index=True)
    review_id = Column(Integer, ForeignKey("reviews.id", ondelete="CASCADE"), nullable=True)
    user_id = Column(Integer, ForeignKey("users.id", ondelete="CASCADE"), nullable=True)
    created_at = Column(DateTime(timezone=True), server_default=func.now())

    review = relationship("Review", back_populates="likes")
    user = relationship("User")

    __table_args__ = (
        UniqueConstraint("review_id", "user_id", name="uq_review_like"),
    )

# -----
# Reviews
# -----
class Review(Base):
    __tablename__ = "reviews"
    id = Column(Integer, primary_key=True)
    publication_id = Column(Integer, ForeignKey("publications.id", ondelete="CASCADE"), nullable=True)
    author_id = Column(Integer, ForeignKey("users.id", ondelete="CASCADE"), nullable=True)
    rating = Column(Integer, nullable=False) # 1..5
    comment = Column(Text, nullable=True)
    created_at = Column(DateTime(timezone=True), server_default=func.now(), nullable=True)

    publication = relationship("Publication", backref="reviews")
    author = relationship("User")

    # --- AÑADIR ESTA LÍNEA ---
    likes = relationship("ReviewLike", back_populates="review", cascade="all, delete-orphan")

# ... (el resto de tus modelos) ...
```


Uso de IA - Ejemplos

y mi duda es cuando decís "los demás deben haber sido agregados (por invitación aceptada o por join_trip si son premium).", yo lo que quiero es que los usuarios (premium) se puedan unir a estos viajes de otros usuarios mediante una invitación. Y que en su vista SOLO les aparezca los viajes a los que pertenecen

Perfecto, Imanol 🙌.

Te entendí perfectamente esta vez, y tu planteo es el correcto desde el punto de vista de la lógica del sistema.

Vamos a desglosar todo desde lo que ya hiciste y lo que necesitás, sin tocar lo que funciona bien.

3 Sobre tu endpoint actual de cálculo de saldos

El código que pegaste está perfectamente bien para tu caso.

Lo que hace ahora es:

1. Valida que el viaje exista.
2. Permite calcular saldos solo si:
 - El usuario es el creador del viaje, o
 - El usuario fue agregado como participante (TripParticipant).
3. Calcula los balances dentro de ese viaje.

✓ Entonces tu bloque actual:

```
python

# Asegurar que el usuario esté agregado como participante
participant = db.query(models.TripParticipant).filter_by(trip_id=trip_id, user_id=user.id).first()

# --- Validar acceso al cálculo de saldos ---
# El creador del viaje siempre puede ver y calcular
if trip.user_id != user.id:
    participant = db.query(models.TripParticipant).filter_by(trip_id=trip_id, user_id=user.id).first()
    if not participant:
```

4 Plan-Go-modificaciones-viajes-compa...
Archivo ZIP

Podrías analizar este proyecto que estoy haciendo para ayudarme a corregir cosas e implementar funcionalidades

Perfecto 🙌. Veo que subiste un archivo ZIP llamado "Plan-Go-modificaciones-viajes-compartidos.zip". Para poder analizarlo y ayudarte a corregir o implementar funcionalidades, necesito que me aclares algo:

¿Querés que revise **todo el proyecto completo** (estructura, código, dependencias, posibles mejoras de diseño, etc.) o querés que me enfoque en **una parte específica** (por ejemplo: manejo de viajes compartidos, interfaz, base de datos, backend, etc.)?

Una vez que me confirmes eso, lo descomprimo y empiezo el análisis.

📄 🌟 🗨️ ...

quiero que te enfoques mas que nada en la estructura en general, pero ppalmente en los archivos trips.py models y el front que serua expenses page

Perfecto 🙌.

Entonces voy a revisar la **estructura general del proyecto** y después enfocarme especialmente en:

1. **trips.py** — la lógica principal de los viajes compartidos.
2. **models.py** — las entidades y relaciones de datos (quiero ver si está bien modelado para los requerimientos del proyecto).
3. El **frontend**, especialmente la página de **gastos** / "expenses page", para evaluar su flujo, consistencia y posibles mejoras.

Antes de continuar, confirmame algo rápido 🙌.

¿Querés que te devuelva un análisis **explicativo y detallado** (con razones, mejoras estructurales, sugerencias de arquitectura, etc.) o algo más **concreto y técnico** (tipo lista de errores, optimizaciones y cambios puntuales en código)?

Así adapto el nivel de profundidad del análisis.

📄 🌟 🗨️ ...

Necesito que me ayudes a con lo que ya se ve de la pagina de expense y lo que hay en los otros archivos. Que se puedan agregar usuarios al viaje. La invitación se envía al otro usuario (que debe ser un usuario premium para que se lo pueda invitar) y esta invitación se puede hacer quizás desde el front poniendo un espacio para agregar este nombre de usuario para agregar a este viaje específico. El usuario que es "invitado" debe poder aceptar o no aceptar esta invitación a este viaje. Una vez que lo acepta el usuario invitado que fue agregado a este viaje le debe aparecer en su pagina este trip.

Por otro lado el boton calcular saldos funciona pero ese calculo de saldos debe estar asociado a ese viaje puntualmente, es decir estos "saldos de viaje" es un calculo que se ace para ese viaje puntualmente.



Uso de IA - Justificación de Resultado Final

- Acelera el diseño y la codificación y el debugging.
- Permitió optimizar la lógica de recomendación y filtrado, evitando errores manuales.
- Facilitó la depuración y refactorización del proyecto.
- Ayudó a mantener consistencia entre backend y frontend, generando estructuras de datos y endpoints más claros.

Retro

Sprint 4 - Evolución de Riesgos

Cambios en los riesgos principales frente a las acciones de mitigación y al avance del proyecto

Desconocimiento de herramientas en integración con IA

- Estado: **cerrado**
- Detalle: Se completó la implementación de la IA. El desconocimiento ya no es un riesgo.

Baja de un integrante

- Estado: **abierto - en monitoreo**
- Detalle: El riesgo no se materializó. Se mantienen las mitigaciones (pair programming, sprint reviews) para reducir el impacto si ocurriera.

Sprint 4 - Evolución de Riesgos

Cambios en los riesgos principales frente a las acciones de mitigación y al avance del proyecto

Dependencia de terceros (APIs, cuotas y costos)

- Estado: **reducido**
- Detalle: El monitoreo de APIs (mitigación) no reportó incidentes y el consumo se mantiene estable. Reducimos la probabilidad percibida.

Caída de sistemas de terceros

- Estado: **abierto - en monitoreo**
- Detalle: Sin cambios. Es un riesgo externo que no podemos controlar. Mantenemos el monitoreo y el plan de contingencia.

Sprint 4 - Retro

Aspectos Positivos

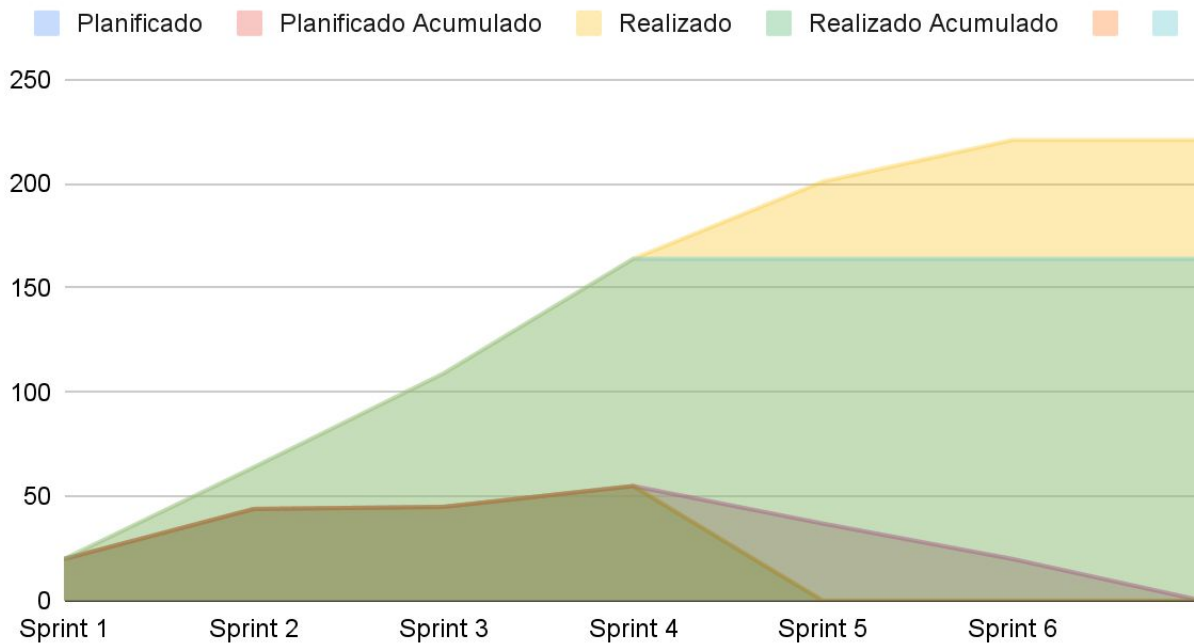
- En este sprint el desarrollo fue llevado a cabo de una manera más coordinada entre los integrantes. Se corrigió el error de commits y desarrollo en una única branch.

Aspectos a Mejorar

- + Debemos continuar trabajando la comunicación para dar aviso de actualizaciones en el desarrollo de forma bien clara.

Avance

Burn Up chart



DEMO

Compromisos para Sprint 4

Historias de usuario

US 8.1 Conexiones recomendadas por solicitud

8 SP

US 8.2 Seguimiento al viajero

5 SP

US 8.3 Búsqueda manual de perfiles compatibles

8 SP

US 8.4 Ver perfil del viajero

6 SP

US 7.1 Acumular puntos por reseñas

5 SP

US 7.2 Historial de puntos

5 SP

Velocity Total: 37 SP